國 立 交 通 大 學

應用數學系
碩 士 論 文

弦環式網路之全體對全體私有訊息傳送問題之研究

On the All-to-All Personalized Exchange Problem

in Chordal Ring Networks

研 究 生：曾慧棻

指導教授：陳秋媛　教授

中 華 民 國 九 十 九 年 一 月

弦環式網路之全體對全體私有訊息傳送問題之研究

On the All-to-All Personalized Exchange Problem in Chordal Ring Networks

研 究 生：曾慧棻　　　　Student：Hui-Fen Tseng

指導教授：陳秋媛　　　　Advisor：Chiuyuan Chen

國 立 交 通 大 學

應 用 數 學 系

碩 士 論 文

A Thesis

Submitted to Department of Applied Mathematics

College of Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Applied Mathematics

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年一月

弦環式網路之全體對全體私有訊息傳送問題之研究

研究生：曾慧萘　　　　　　　　　　　　指導老師：陳秋媛　教授

國 立 交 通 大 學

應 用 數 學 系

摘　要

弦環式網路中的邊可分為「弦邊」和「環邊」。在文獻[5]和文獻[7]中，Masuyama 等學者提出了兩個弦環式網路的全體對全體通訊演算法，其中的第一個演算法（為方便，稱之演算法 A）是一個全體對全體私有訊息傳送演算法，它使用於網路中沒有任何壞點時，第二個演算法（為方便，稱之演算法 B）是一個全體對全體廣播演算法，當網路中有一至二個壞點時，它仍可使用。文獻[5]和文獻[7]證明了：對弦環式網路 $CR(N,w)$ 而言，演算法 A 需要 $\sum_{i=1}^{\frac{N}{2}} i$ 單位時間，其中 N 表示節點數，w 表示弦長。然而，我們發現演算法 A 只使用弦環式網路中的環邊來傳送訊息，演算法 B 只在網路中有壞點時才會使用弦邊來傳送訊息。文獻[5]和文獻[7]的演算法浪費了大量的網路硬體，因為只要沒有壞點就不會使用弦邊。在這篇論文中，我們利用弦邊來使弦環式網路的全體對全體私有訊息傳送更快速。我們首先提出一個利用弦邊的演算法來執行弦環式網路的全體對全體私有訊息傳送；我們以實際數據來證明我們的演算法比演算法 A 花費較少的單位時間，因此改進了演算法 A。此外，我們也提出了一個針對 $w=3$ 的弦環式網路的全體對全體私有訊息傳送演算法，我們證明了此演算法比演算法 A 花費少 50%以上的單位時間。最後，我們闡明了演算法 B 中的一些不清楚或不正確的地方。

關鍵詞：弦環式網路、路由、容錯、全體對全體通訊、全體對全體廣播、全體對全體私有訊息傳送。

中 華 民 國 九 十 九 年 一 月

On the All-to-All Personalized Exchange Problem in Chordal Ring Networks

Student: Hui-Fen Tseng Advisor: Chiuyuan Chen

Department of Applied Mathematics
National Chiao Tung University
Hsinchu, Taiwan 30050

Abstract

In [5, 7], Masuyama et al. proposed two all-to-all communication algorithms for chordal ring networks of degree 3. The first algorithm (call it Algorithm A) is an all-to-all personalized exchange algorithm and it is used when there is no fault. The second algorithm (call it Algorithm B) is an all-to-all broadcast algorithm and it can tolerate one or two faults. In [5, 7], it has been proven that Algorithm A takes $\sum_{i=1}^{\frac{N}{2}} i$ time units to fulfill an all-to-all personalized exchange in a chordal ring network $CR(N, w)$, where N is the number of nodes and w is the chord length of the chordal ring network. However, we observe that Algorithm A only utilizes ring-links to fulfill an all-to-all communication and Algorithm B utilizes chordal-links only when there are faults. Since all of the chordal-links are not used in any all-to-all communication when there is no fault, a huge amount of hardware is wasted. In this thesis, we will use chordal-links to facilitate an all-to-all personalized exchange. In particular, we propose an all-to-all personalized exchange algorithm that works for all chordal ring networks. We will show that our algorithm uses less time units to fulfill an all-to-all personalized exchange and hence improves Algorithm A. We also provide an all-to-all personalized exchange algorithm that works only for chordal ring networks with $w = 3$ and clarify some unclear parts and correct some incorrect parts in Algorithm B.

Keywords: chordal ring network, routing, fault tolerance, all-to-all communication, all-to-all personalized exchange.

誌 謝

打從高雄北上新竹念書，老實說對我而言是種很大的挑戰，因為在台灣交大應數離散組是數一數二掛著離散界的大龍頭，想當然爾一定有很多強者在這邊，所以我其實是抱著忐忑不安的心進入這個大環境,但是讓我很意外的是系上的教授不僅在教學上有淵博的學識，而且對學生很親切，除了課業上給予支持外，也會關心學生生活的細節，讓我深刻的體會到這個環境其實是很令人放心。感謝所有我修過課的教授，你們有條不紊的講解方式，讓我很快的進入學習的殿堂，加入這個大家庭。我修過最多門課的教授，傅恆霖教授，老師上課很風趣，教學中也會把一些他的人生經驗及想法傳達給我們,讓我們學習不忘吸取老師其他的寶貴經驗；麥可教授，雖然老師是英文授課，英文不好的我，上起課來著時費力，但是在課後時間老師總是會努力的用其他講解方式讓我們理解課堂內容,諸如此類，交大應數的老師都很棒，總把學生擺在前線去關心。

再之是我的指導教授—陳秋媛教授，雖然在修課其間剛好錯開了老闆開的課，但是在我的研究生活中，老師就像一個好媽媽,除了研究上給了我們很多意見，以及方向外，對於我們生活一切，老師也都會關心，讓在異地求學的我倍感溫馨，謝謝老師這兩年的指導以及愛護。

次之就是感謝我在碩班認識的所有同學，油油、阿華田、小張、簡科科、阿毅、哲皓、昱丞、神…細數不完的同學，謝謝你們在我研究生活增添很多色彩及歡樂；以及同師門的國元學長、鈺傑學長、pollow、志文、子鴻、天兵、人拍、士慶，及學弟們，感謝你們對我的照顧，以及在我看到爆炸的 paper 時指點我一些方向和建議。

最後特別感謝我最親愛的家人，一路上走來我只有一句話要對你們說，有你們真好，謝謝你們，希望我能成為你們的驕傲。

曾慧棻

謹誌于交通大學
2010 年 1 月

Contents

List of Figures

List of Tables

1 Introduction

Processors in a parallel and distributed processing system often need to communicate with other processors. The communication among these processors could be *one-to-one*, *one-to-many*, or *all-to-all*. All-to-all communication can be further classified into *all-to-all broadcast* and *all-to-all personalized exchange*. In all-to-all broadcast, each processor sends the same message to all other processors; while in all-to-all personalized exchange, each processor sends a specific message to every other processor.

All-to-all personalized exchange occurs in many important applications (for example, matrix transposition and fast Fourier transform (FFT)) in parallel and distributed computing. The all-to-all personalized exchange problem has been extensively studied for hypercubes, meshes, and tori; see [9] for details.

In [9], Yang and Wang considered fulfilling all-to-all personalized exchange on multi-stage interconnection networks. The purpose of this thesis is to consider fulfilling all-to-all personalized exchange in chordal ring networks. Graph terminologies and notations in this thesis are standard; see [3] and [8] except as indicated. All the graphs considered in this thesis are assumed simple. For convenience, vertices of a graph are also called nodes.

A *chordal ring network* $CR(N, w)$, where N is a positive even integer and w is a positive odd integer and $w \leq N/2$, is a graph with N nodes $0, 1, \ldots, N-1$ and $3N/2$ links of the forms:

$$\begin{array}{lll} \text{ring-links:} & (i, (i+1) \bmod N), & i=0,1,2,\ldots,N-1, \\ \text{chordal-links:} & (i, (i+w) \bmod N), & i=1,3,5,\ldots,N-1. \end{array}$$

See Figure 1 for an example. Chordal ring networks were first proposed by Arden and Lee [1] and these networks can be viewed as adding chords to a ring network (a cycle).

In this thesis, each node in a chordal ring network has exactly three neighbors; some researchers call such a 3-regular chordal ring network a *chordal ring network of degree 3*. In this thesis, we consider chordal ring networks of degree 3. Do notice that some researchers considered chordal ring networks of degree 4 or degree 6 (in other words, each vertex has four or six neighbors); see [2]. Note that communication algorithms can

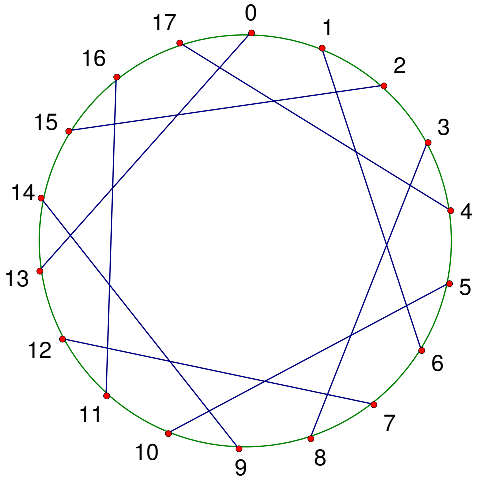

Figure 1: $CR(18, 5)$: the chordal ring network with $N = 18$ and $w = 5$.

be implemented in a *one-port model* or a *multi-port model*. In a one-port model, only one incident link of a node can be used; however, in a multi-port model, more than one incident link of a node can be used simultaneously. In this thesis, we assume a multi-port model since a multi-port model can be simulated by a one-port model; this is also the model used in [5]. Further, the links in a chordal ring network can be either unidirectional or bidirectional. In this thesis, we follow the convention used in [5] that all the links in a chordal ring network are *bidirectional*. It is assumed that there are at most two faults in an all-to-all communication. We also assume that it takes one time unit to transmit a message from a node to an adjacent node.

In [5, 7], Masuyama et al. proposed two all-to-all personalized exchange algorithms for chordal ring networks of degree 3; for convenience, call these two algorithms Algorithm A and Algorithm B. Algorithm A is used when there is no fault. Algorithm B is used when at least one fault occurs. See also [6] for optical routing algorithms for chordal ring of degree 4. In this thesis, we consider the all-to-all personalized exchange problem in degree 3 chordal ring networks. Therefore, in the remaining discussions, unless other specified, a chordal ring network means a degree 3 chordal ring network.

The motivation of this thesis is as follows. In [5, 7], it has been proven that $\frac{N}{2}$ time units can fulfill an all-to-all broadcast in a chordal ring network and $\sum_{i=1}^{\frac{N}{2}} i$ time units can fulfill an all-to-all personalized exchange in a chordal ring network. However, we observe that Algorithm A only utilizes ring-links to fulfill an all-to-all communication and chordal-links are used only when there are faults. Since all of the chordal-links are not used in any all-to-all communication when there is no fault, a huge amount of hardware is wasted. In this thesis, we will use chordal-links to facilitate an all-to-all personalized exchange. In particular, we propose an all-to-all personalized exchange algorithm that works for all chordal ring networks; we will call it a *general* algorithm. We will use experimental results to show that our general algorithm uses less time units to fulfill an all-to-all personalized exchange and hence improves the previous algorithms in [5, 7]. We also provide an all-to-all personalized exchange algorithm that works only for chordal ring networks with $w = 3$; we will show that this algorithm makes a huge improvement for the previous algorithms in [5, 7].

When there is only one fault, Algorithm B has several unclear parts. In this thesis, we will clarify these unclear parts. When there are two faults, we will show that Algorithm B actually does not fulfill an all-to-all broadcast. In this thesis, we will point out why Algorithm B does not fulfill an all-to-all broadcast.

This thesis is organized as follows. Section 2 gives some preliminaries. Section 3 gives an algorithm for fulfilling all-to-all personalized exchange in a chordal ring network; this algorithm works for all chordal ring networks $CR(N, w)$. Section 4 gives an algorithm for fulfilling all-to-all personalized exchange in chordal ring networks $CR(N, w)$ with $w = 3$. In Section 5, we will point out the unclear parts and incorrect parts of Algorithm B, and we also correct Algorithm B. The concluding remarks are given in the last section.

2 Preliminaries

Recall that a chordal ring network $CR(N, w)$ contains ring-links and chordal-links. We will follow the following definitions given in [5, 7]: $R(i)$, $R(i)^{start}$, $R(i)^{end}$, and $\overline{R(1)}^f$.

(1) Ring $R(1)$ is a loop which is composed of one chordal-link (labeled by (1) in Figure 2), w ring-links bypassed by the chordal-links, and $w+1$ nodes.

(2) Ring $R(2)$ is a loop which is composed of two chordal-links which have only one common ring-link bypassed by the two consecutive chordal-links (labeled by (1) and (2) in Figure 2), and $2w$ nodes, but does not include the common bypassed ring-link.

(3) Ring $R(i)$ is a loop which is composed of ring $R(i-1)$ exclusive of a ring-link bypassed by the i-th additional new chordal-link (labeled by (i) in Figure 2), the additional new chordal-link, and $w-1$ ring-links and nodes bypassed by the additional new chordal-link.

(4) Two nodes labels $R(i)^{start}$ and $R(i)^{end}$ on each loop $R(i)$ are special marks that are used to find the loop $R(i)$. See Figure 2 for an example of $R(i)^{start}$ and $R(i)^{end}$.

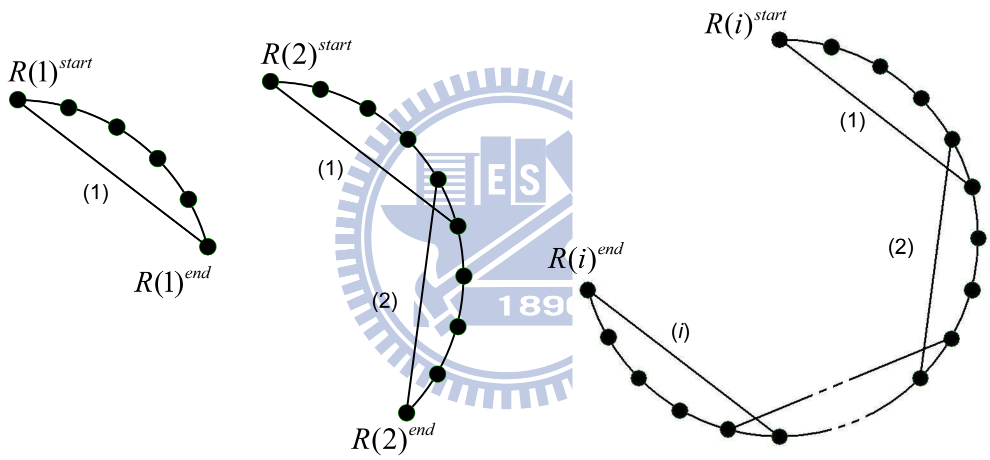

Figure 2: $R(1)$, $R(2)$, and $R(i)$ in the $w = 5$ case.

(5) Given a faulty node f, find a ring $R(1)$ such that f is in the *middle* of $R(1)$ and denote this $R(1)$ by $R(1)^f$. Make the longest non-faulty loop which is composed of $R(1)^{start}$, $R(1)^{end}$, and all nodes except nodes on $R(1)^f$, and denote this loop by $\overline{R(1)}^f$. For example, in Figure 3, suppose the node depicted in color black is the faulty node. Then $R(1)^f$ is the loop $1, 2, 3, 4, 5, 6, 1$ and $\overline{R(1)}^f$ is the loop $1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 0, 1$.

In the following, the notations h, k, and k_p are used very often. We now define them. The notations h and k_p denote the numbers of consecutive ring-links started from a given

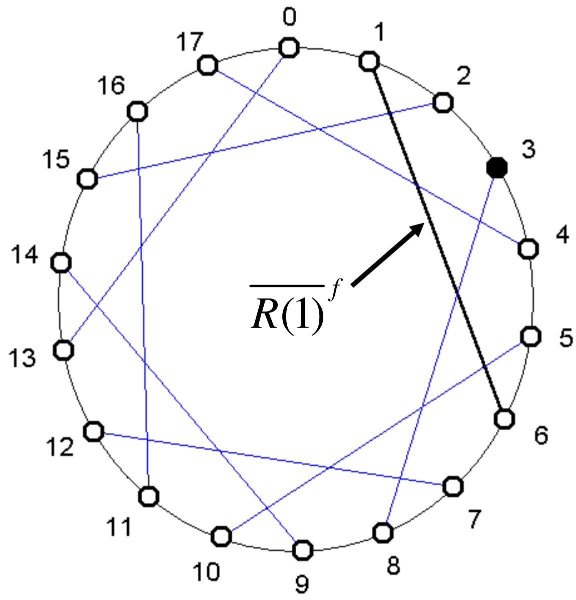

Figure 3: $\overline{R(1)}^f$.

source node and used by our algorithm. Further, k denotes the number of consecutive (ring-link, chordal-link) pairs used by our algorithm. Notice that h and k_p require that the ring-links start from a given source node. Also notice that k requires that a total of k (ring-link, chordal-link) pairs are used and in each pair of (ring-link, chordal-link), a ring-link is used before a chordal-link.

3 Main result–a general all-to-all personalized exchange algorithm

Recall that Masuyama et al. [5, 7] proposed an all-to-all personalized exchange algorithms for chordal ring network when there is no fault and we call it Algorithm A. This algorithm takes $\sum_{i=1}^{\frac{N}{2}} i$ time units and this algorithm does not utilize any chordal-links. In this section, we will propose an all-to-all personalized exchange algorithm that utilizes both ring-links and chordal-links; since our algorithm works for all chordal ring networks, we also say that it is a general algorithm.

In the remaining discussion, a *processor* means a node in a given chordal ring network. Moreover, we assume that the indices of processors are taken modulo N; for example, processor P_{i+h+1} is processor $P_{(i+h+1) \bmod N}$ and processor $P_{i+N-h-1}$ is processor $P_{(i+N-h-1) \bmod N}$. Our algorithm, Algorithm 1, is an algorithm for fulfilling an all-to-all personalized exchange in chordal ring networks. Algorithm 1 is divided into two parts: part 1 handles the $h \geq w$ case and part 2, the $h < w$ case. For convenience, in Algorithm 1, the following terminologies are used:

route 1: a route via k (ring-link,chordal-link) pairs and k_p ring-links;

route 2: a route via k_p ring-links and k (ring-link, chordal-link) pairs;

route 3: a route via k (ring-link,chordal-link) pairs, k_p ring-links, and 1 ring-link;

route 4: a route via k_p ring-links, k (ring-link,chordal-link) pairs, and 1 ring-link;

route 5: a route via k (ring-link,chordal-link) pairs, 1 ring-link, and 1 chordal-link;

route 6: a route via 1 chordal-link, k (ring-link,chordal-link) pairs, and 1 ring-link.

Algorithm 1 works as follows. All the processors $P_0, P_1, \ldots, P_{N-1}$ transmit their messages simultaneously. Each P_i first computes a value h and transmits a personalized message to each processor which is within a distance h from P_i. After the above process, $P_{i+1}, P_{i+2}, \ldots, P_{i+h}$ and also P_{i-1}, P_{i-2}, $\ldots$, P_{i-h} have received a personalized message from P_i; only P_{i+h+1}, P_{i+h+2}, $\ldots$, $P_{i+N-h-1}$ have not received a personalized message from P_i yet. Then, according to whether $h \geq w$ or not, Algorithm 1 is divided into two parts.

Part 1 of Algorithm 1 (the $h \geq w$ case) works as follows. Each P_i computes the values k and k_p (note that k_p is always less than or equal to w). According to the parity of h, the $h \geq w$ case actually consists of two subcases.

- Suppose h is even.

  ◇ Suppose $k_p < w$.

  If i is even (resp., odd), then P_i will use route 1 (resp., route 2) to arrive P_{i+h+1} and use route 2 (resp., route 1) to arrive $P_{i+N-h-1}$. Once P_i can arrive P_{i+h+1}, it can transmit, via ring-links, a personalized message to P_{i+h+1}, P_{i+h+2}, and so on; simultaneously, once P_i can arrive $P_{i+N-h-1}$, it can transmit, via ring-links, a personalized message to $P_{i+N-h-1}$, $P_{i+N-h-2}$, and so on.

◇ Suppose $k_p = w$.

If i is even (resp., odd), then P_i will use route 5 (resp., route 6) to arrive P_{i+h+2} and use route 6 (resp., route 5) to arrive $P_{i+N-h-2}$. Once P_i can arrive P_{i+h+2}, it can transmit, via a ring-link, a personalized message to P_{i+h+1}; simultaneously, once P_i can arrive $P_{i+N-h-2}$, it can transmit, via a ring-link, a personalized message to $P_{i+N-h-1}$. Also, once P_i can arrive P_{i+h+2}, it can transmit, via ring-links, a personalized message to P_{i+h+2}, P_{i+h+3}, and so on; simultaneously, once P_i can arrive $P_{i+N-h-2}$, it can transmit, via ring-links, a personalized message to $P_{i+N-h-2}$, $P_{i+N-h-3}$, and so on.

- Suppose h is odd.

◇ Suppose $k_p < w$.

If i is even (resp., odd), then P_i will use route 3 (resp., route 4) to arrive P_{i+h+1} and use route 4 (resp., route 3) to arrive $P_{i+N-h-1}$. Once P_i can arrive P_{i+h+1}, it can transmit, via ring-links, a personalized message to P_{i+h+1}, P_{i+h+2}, and so on; simultaneously, once P_i can arrive $P_{i+N-h-1}$, it can transmit, via ring-links, a personalized message to $P_{i+N-h-1}$, $P_{i+N-h-2}$, and so on.

◇ Suppose $k_p = w$.

If i is even (resp., odd), then P_i will use route 5 (resp., route 6) to arrive P_{i+h+1} and use route 6 (resp., route 5) to arrive $P_{i+N-h-1}$. Once P_i can arrive P_{i+h+1}, it can transmit, via ring-links, a personalized message to P_{i+h+1}, P_{i+h+2}, and so on; simultaneously, once P_i can arrive $P_{i+N-h-1}$, it can transmit, via ring-links, a personalized message to $P_{i+N-h-1}$, $P_{i+N-h-2}$, and so on.

Part 2 of Algorithm 1 (the $h < w$ case) works as follows. In a chordal ring network, $N \geq 2w$ always holds. According to whether $N = 2w$, the $h < w$ case is divided into two subcases.

- Suppose $N = 2w$.

  - ◇ Suppose i is even.

    Then P_i will use a chordal-link to arrive P_{i-w} and use 1 ring-link plus 1 chordal-link to arrive P_{i+1+w} (i.e., P_i arrives P_{i+1+w} via P_{i+1}). Once P_i can arrive P_{i-w}, it can transmit, via ring-links, a personalized message to $P_{i-w}, P_{i-w-1}, \ldots, P_{i+h+1}$; simultaneously, once P_i can arrive P_{i+1+w}, it can transmit, via ring-links, a personalized message to $P_{i+1+w}, P_{i+1+w+1}, \ldots, P_{i+N-h-1}$.

  - ◇ Suppose i is odd.

    Then P_i will use a chordal-link to arrive P_{i+w} and use 1 ring-link plus 1 chordal-link to arrive P_{i-1-w} (i.e., P_i arrives P_{i-1-w} via P_{i-1}). Once P_i can arrive P_{i+w}, it can transmit, via ring-links, a personalized message to $P_{i+w}, P_{i+w+1}, \ldots, P_{i+N-h-1}$; simultaneously, once P_i can arrive P_{i-1-w}, it can transmit, via ring-links, a personalized message to $P_{i-1-w}, P_{i-1-w-1}, \ldots, P_{i+h+1}$.

- Suppose $N \geq 2w + 2$.

  - ◇ Suppose i is even.

    Then P_i will use a chordal-link to arrive P_{i-w} and use 1 ring-link plus 1 chordal-link to arrive P_{i+1+w} (i.e., P_i arrives P_{i+1+w} via P_{i+1}). Once P_i can arrive P_{i-w}, it can transmit, via ring-links, a personalized message to $P_{i-w}, P_{i-w+1}, \ldots, P_{i+N-h-1}$ (in a clockwise fashion) and also to P_{i-w-1}, P_{i-w-2}, and so on (in a counterclockwise fashion); simultaneously, once P_i can arrive P_{i+1+w}, it can transmit, via ring-links, a personalized message to $P_{i+1+w}, P_{i+1+w-1}, \ldots, P_{i+h+1}$ (in a counterclockwise fashion) and also to $P_{i+1+w+1}$, $P_{i+1+w+2}$, and so on (in a clockwise fashion).

  - ◇ Suppose i is odd.

    Then P_i will use a chordal-link to arrive P_{i+w} and use 1 ring-link plus 1 chordal-link to arrive P_{i-1-w} (i.e., P_i arrives P_{i-1-w} via P_{i-1}). Once P_i can arrive P_{i+w}, it can transmit, via ring-links, a personalized message to $P_{i+w}, P_{i+w-1}, \ldots,$

P_{i+h+1} (in a counterclockwise fashion) and also to P_{i+w+1}, P_{i+w+2}, and so on (in a clockwise fashion); simultaneously, once P_i can arrive P_{i-1-w}, it can transmit, via ring-links, a personalized message to $P_{i-1-w}, P_{i-1-w+1}, \ldots, P_{i+N-h-1}$ (in a clockwise fashion) and also to $P_{i-1-w-1}$, $P_{i-1-w-2}$, and so on (in a counterclockwise fashion).

The following is the outline of Algorithm 1.

$$\text{Algorithm 1:}\begin{cases} \text{part 1: the } h \geq w \text{ case} \begin{cases} h \text{ is even} \begin{cases} k_p < w \ (\textbf{case 1a}) \\ k_p = w \ (\textbf{case 1b}) \end{cases} \\ h \text{ is odd} \begin{cases} k_p < w \ (\textbf{case 1c}) \\ k_p = w \ (\textbf{case 1d}) \end{cases} \end{cases} \\ \text{part 2: the } h < w \text{ case} \begin{cases} N = 2w & (\textbf{case 2a}) \\ N \geq 2w + 2 & (\textbf{case 2b}) \end{cases} \end{cases}$$

Now we are ready to present Algorithm 1.

Algorithm 1 All-to-all personalized exchange algorithm for chordal ring networks.

Input: A chordal ring network $CR(N, w)$.

Output: All-to-all personalized exchange of $CR(N, w)$.

for each P_i $(0 \leq i \leq N-1)$ **do in parallel**
  $h \leftarrow \lfloor \frac{33}{96} N \rfloor$;
  for $j = 1$ to h **do**
    P_i prepares personalize messages for P_{i+j} and P_{i-j};
    P_i transmits personalize messages to P_{i+j} and P_{i-j} via ring-links;
  endfor
  if $h < w$ **then** goto part 2; **endif**
  part 1:
  $k \leftarrow \lfloor \frac{h+1}{w+1} \rfloor$;
  $k_p \leftarrow h + 1 - k(w+1)$;
  $a \leftarrow h + 1$;
  $b \leftarrow N - h - 1$;
  if h is even **then**
    if $k_p < w$ **then**
      for $j = 0$ to $\lceil \frac{N-2h-3}{2} \rceil$ **do**
        if i is even **then**
          P_i transmits personalize messages to P_{i+j+a} via route 1 and j ring-links
          and to P_{i-j+b} via route 2 and j ring-links;
        else $/*$ i is odd $*/$
          P_i transmits personalize messages to P_{i+j+a} via route 2 and j ring-links
          and to P_{i-j+b} via route 1 and j ring-links;
        endif
      endfor
    else $/*$ $k_p = w$ $*/$
      P_i transmits personalize messages to P_{i+a+1} via route 5 and 1 ring-links
      to P_{i+a} and to P_{i+b-1} via route 6 and 1 ring-links to P_{i+b};
      for $j = 0$ to $\lceil \frac{N-2h-5}{2} \rceil$ **do**
        if i is even **then**
          P_i transmits personalize messages to $P_{i+j+a+1}$ via route 5 and j ring-links
          and to $P_{i-j+b-1}$ via route 6 and j ring-links;
        else $/*$ i is odd $*/$
          P_i transmits personalize messages to $P_{i+j+a+1}$ via route 6 and j ring-links
          and to $P_{i-j+b-1}$ via route 5 and j ring-links;
        endif
      endfor
    endif
  else $/*$ h is odd $*/$
    if $k_p < w$ **then**
      for $j = 0$ to $\lceil \frac{N-2h-3}{2} \rceil$ **do**
        if i is even **then**
          P_i transmits personalize messages to P_{i+j+a} via route 3 and j ring-links
          and to P_{i-j+b} via route 4 and j ring-links;
        else
          P_i transmits personalize messages to P_{i+j+a} via route 4 and j ring-links
          and to P_{i-j+b} via route 3 and j ring-links;

endif
endfor
endif
else /∗ $k_p = w$ ∗/
for $j = 0$ to $\lceil \frac{N-2h-3}{2} \rceil$ **do**
if i is even **then**
P_i transmits personalize messages to P_{i+j+a} via route 5 and j ring-links
and to P_{i-j+b} via route 6 and j ring-links;
else /∗ i is odd ∗/
P_i transmits personalize messages to P_{i+j+a} via route 6 and j ring-links
and to P_{i-j+b} via route 5 and j ring-links;
endif
endfor
endif
stop the algorithm
part 2:
$c \leftarrow i - w$; $d \leftarrow i + w$;
if $N = 2w$ **then**;
for $j = 0$ to $\lceil \frac{N-2h-3}{2} \rceil$ **do**
if i is even **then**
P_i transmits personalize messages to P_{c-j} via 1 chordal-link and j ring-links,
and to P_{1+d+j} via P_{i+1}, 1 chordal-link, and j ring-links;
else /* i is odd */
P_i transmits personalize messages to P_{d+j} via 1 chordal-link and j ring-links,
and to P_{c-1-j} via P_{i-1}, 1 chordal-link, and j ring-links;
endif
endfor
else /* $N \geq 2w + 2$ */
for $j = 0$ to $w - h + 1$ **do**
if i is even **then**
P_i transmits personalize messages to P_{c+j} via 1 chordal-link and j ring-links,
and to P_{1+d-j} via P_{i+1}, 1 chordal-link, and j ring-links;
else /* i is odd */
P_i transmits personalize messages to P_{d-j} via 1 chordal-link and j ring-links,
and to P_{c-1+j} via P_{i-1}, 1 chordal-link, and j ring-links;
endif
endfor
for $j = 0$ to $\frac{N-2w-2}{2}$ **do**
if i is even **then**
P_i transmits personalize messages to P_{c-j} via 1 chordal-link and j ring-links,
and to P_{1+d+j} via P_{i+1}, 1 chordal-link, and j ring-links;
else /* i is odd */
P_i transmits personalize messages to P_{d+j} via 1 chordal-link and j ring-links,
and to P_{c-1-j} via P_{i-1}, 1 chordal-link, and j ring-links;
endif
endfor
endif
end for

Before going further, we give two remarks of Algorithm 1.

Remark 1. When a personalized message is sent from P_i to P_j, we try to utilize as many (ring-link, chordal-link) pairs as possible; k is the maximum number of such (ring-link, chordal-link) pairs.

Remark 2. When a personalized message is sent from P_i to P_j, k (ring-link, chordal-link) pairs may not make P_i arrive P_j; k_p ring-links are used to make P_i arrive P_j.

We now give six examples for Algorithm 1.

Example 1. Consider $CR(24, 5)$. Then $h = 8$. Since $h \geq w$, part 1 is executed and $k = 1$, $k_p = 3$. So case 1a occurs. Figure 4(a) shows the general situation of case 1a with P_0 being the source of personalized messages. Figure 4(b) shows the situation of $CR(24, 5)$ with P_0 being the source of personalized messages. Figure 4(b) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent the k (ring-link,chordal-link) pairs, and the lines depicted as color grey represent the k_p ring-links. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_8$ and also to $P_{23}, P_{22}, \ldots, P_{16}$. After executing lines 15 to 23, P_0 utilizes route 1 to arrive P_9 and route 2 to arrive P_{15}; so P_0 transmits personalized messages to $P_9, P_{10}, P_{11}, P_{12}$ and also to P_{15}, P_{14}, P_{13}, P_{12}.

Example 2. Consider $CR(20, 3)$. Then $h = 6$. Since $h \geq w$, part 1 is executed and $k = 1$. So case 1b occurs. Figure 5(a) shows the general situation of case 1b with P_0 being the source of personalized messages. Figure 5(b) shows the situation of $CR(20, 3)$ with P_0 being the source of personalized messages. Figure 5(b) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent the k (ring-link,chordal-link) pairs, the lines depicted as color grey represent the additional one ring-link, and the lines depicted as dashed represent the w. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_6$, and also to $P_{19}, P_{18}, \ldots, P_{14}$.

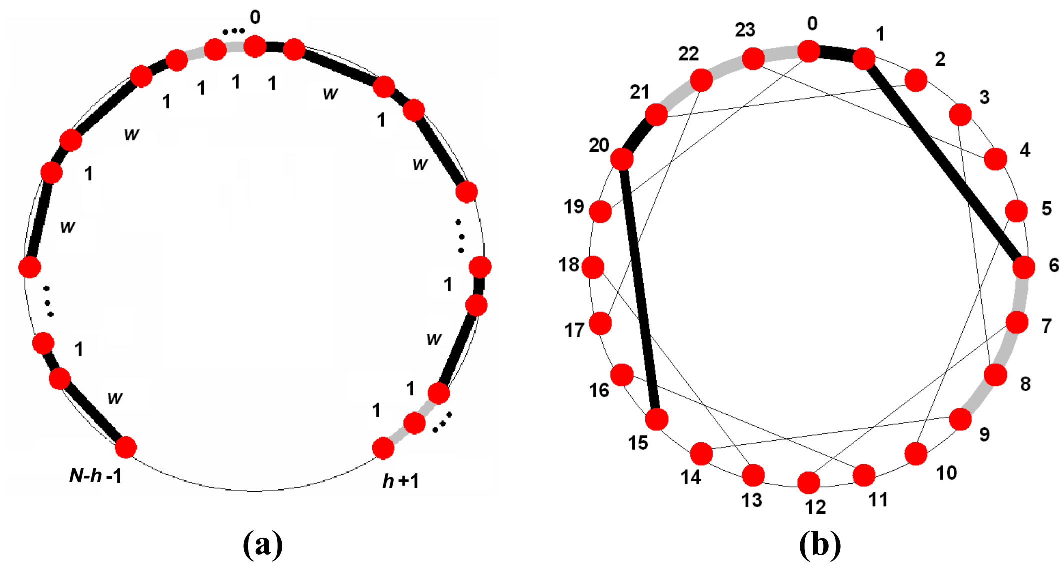

Figure 4: (a) The general situation of case 1a. (b) The situation of case 1a for $CR(24, 5)$.

After executing lines 25 to 35, P_0 utilizes route 5 to arrive P_8 and route 6 to arrive P_{12}; so P_0 transmits personalized messages to P_8, P_9, P_{10}, and also to P_{12}, P_{11}, P_{10}. Finally P_0 transmits personalized messages to P_7, P_{13}.

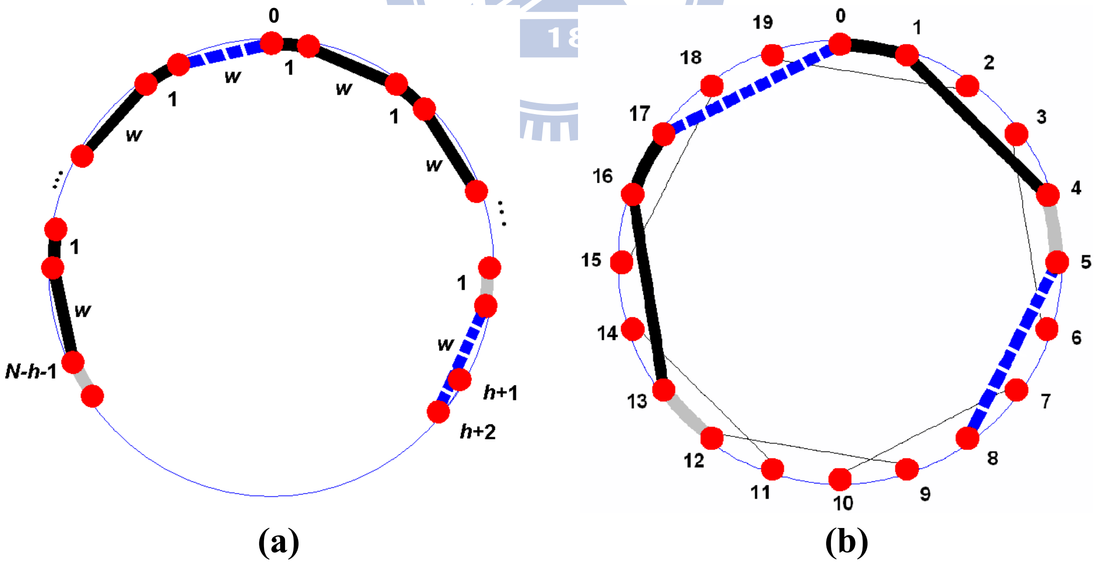

Figure 5: (a) The general situation of case 1b. (b) The situation of case 1b for $CR(20, 3)$.

Example 3. Consider $CR(22, 5)$. Then $h = 7$. Since $h \geq w$, part 1 is executed and $k = 1$, $k_p = 1$. So case 1c occurs. Figure 6(a) shows the situation of $CR(22, 5)$

with P_0 being the source of personalized messages. Figure 6(a) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent the k (ring-link,chordal-link) pairs, the line depicted as color grey represent the k_p ring-links, and last the line depicted as color grey represent additional ring-link. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_7$, and also to $P_{21}, P_{20}, \ldots, P_{15}$. After executing lines 38 to 48, P_0 utilizes route 3 to arrive P_8 and route 4 to arrive P_{14}; so P_0 transmits personalized messages to P_9, P_{10}, P_{11}, and also to P_{13}, P_{12}, P_{11}.

Example 4. Consider $CR(22, 3)$. Then $h = 7$. Since $h \geq w$, part 1 is executed and $k = 1$. So case 1d occurs. Figure 6(b) shows the situation of $CR(22, 3)$ with P_0 being the source of personalized messages. Figure 6(b) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent the k (ring-link,chordal-link) pairs, the lines depicted as color grey represent the additional one ring-link, and the lines depicted as dashed represent the w. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_7$, and also to $P_{21}, P_{20}, \ldots, P_{15}$. After executing lines 49 to 59, P_0 utilizes route 5 to arrive P_8 and route 6 to arrive P_{14}; so P_0 transmits personalized messages to P_9, P_{10}, P_{11}, and also to P_{13}, P_{12}, P_{11}.

Example 5. Consider $CR(18, 9)$. Then $h = 6$. Since $h < w$, part 2 is executed. So case 2a occurs. Figure 7(a) shows the situation of $CR(18, 9)$ with P_0 being the source of personalized messages. Figure 7(a) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent P_0 will arrive to two main processors, P_9 and P_{10}. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_6$, and also to $P_{17}, P_{16}, \ldots, P_{12}$. After executing lines 64 to 72, P_0 utilizes a chordal-link to arrive P_9 and and use 1 ring-link plus 1 chordal-link to arrive P_{10}. so P_0 transmits personalized messages to P_9, P_8, P_7; and also to P_{10}, P_{11}.

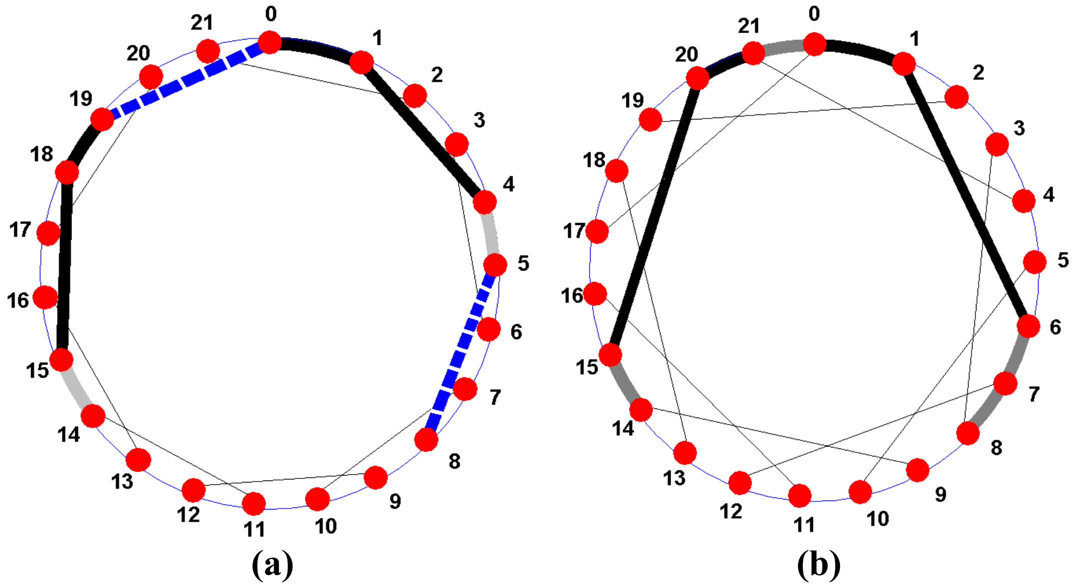

Figure 6: (a) The situation of case 1c for $CR(22, 5)$. (b) Case 1d for $CR(22, 3)$.

Example 6. Consider $CR(20, 7)$. Then $h = 6$. Since $h < w$, part 2 is executed. So case 2b occurs. Figure 7(b) shows the situation of $CR(18, 9)$ with P_0 being the source of personalized messages. Figure 7(b) shows that by using Algorithm 1, P_0 can transmit personalized messages to all other processors. In particular, the lines depicted as color black represent P_0 will arrive to two main processors, P_8 and P_{13}. Thus after executing lines 3 to 6, P_0 transmits personalized messages to $P_1, P_2, \ldots, P_6$, and also to $P_{19}, P_{18}, \ldots, P_{14}$. After executing lines 74 to 91, P_0 utilizes a chordal-link to arrive P_8 and and use 1 ring-link plus 1 chordal-link to arrive P_{13}. so P_0 transmits personalized messages to P_7, P_9, P_{10}; and also to $P_{13}, P_{12}, P_{11}, P_{10}$.

We now prove the correctness of Algorithm 1.

Theorem 1. *Algorithm 1 fulfills an all-to-all personalized exchange for $CR(N, w)$.*

Proof. It suffices to prove that each processor receives a personalized message from P_i, where $0 \le i \le N - 1$. Let P_j be a processor of $CR(N, w)$ such that $j \ne i$. If $j \in \{i+1, i+2, \ldots, i+h, i-1, i-2, \ldots, i+N-h\}$, then P_j receives a personalized message from P_i after lines 3 to 6 are executed. For the $h \ge w$ case, if $j \in \{i+h+1, i+h+2, \ldots, i+N-h-1\}$, then P_j receives a personalized message from P_i after part 1 of

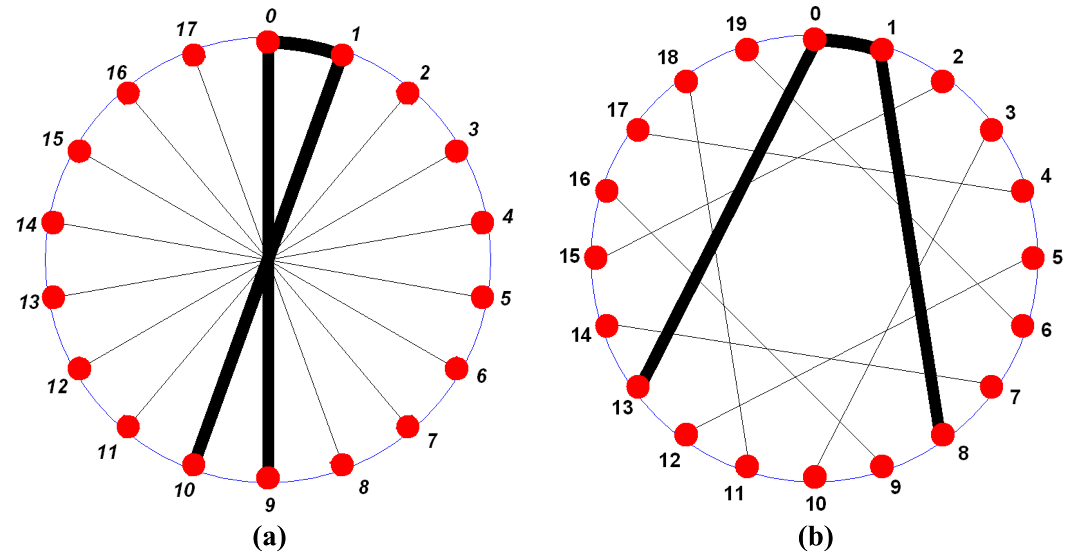

Figure 7: (a) The situation of case 2a for $CR(18, 9)$. (b) Case 2b for $CR(20, 7)$.

Algorithm 1 is executed; these personalized messages are from routes 1, 2, 3, or 4. On the other hand, if $h < w$, then part 2 of Algorithm 1 is executed. Let P_j be a processor of $CR(N, w)$ such that $j \neq i$. If $j \in \{i+1, i+2, \ldots, i+h, i-1, i-2, \ldots, i+N-h\}$, then P_j receives a personalized message from P_i after lines 3 to 6 are executed. If $N = 2w$ and $j \in \{i+h+1, i+h+2, \ldots, i+N-h-1\}$, then P_j receives a personalized message from P_i after lines 64 to 72 are executed. If $N \geq 2w+2$ and $j \in \{i+h+1, i+h+2, \ldots, i+N-h-1\}$, then P_j receives a personalized message from P_i after lines 74 to 91 are executed. ∎

We now analyze the time complexity of Algorithm 1.

Theorem 2. *For $CR(N, w)$, Algorithm 1 takes t time units, where*

$$t = \begin{cases} \frac{h(h+1)}{2} + \frac{(h'+1)(8k+4k_p+h'-1)}{8} & \text{if case 1a occurs,} \\ \frac{h(h+1)}{2} + \frac{(h'-1)(8k+h'+5)}{8} + 2k + 3 & \text{if case 1b occurs,} \\ \frac{h(h+1)}{2} + \frac{(h'-1)(8k+4k_p+h'-1)}{8} & \text{if case 1c occurs,} \\ \frac{h(h+1)}{2} + \frac{(h'+1)(8k+4k_p+h'+3)}{8} & \text{if case 1d occurs,} \\ \frac{h(h+1)}{2} + \frac{(N-w-h)(N-w-h+1)}{2} & \text{if case 2a occurs,} \\ \frac{h(h+1)}{2} + \frac{(w-h+4)(w-h+1)}{2} + \frac{(N-2w+8)(N-2w-2)}{8} & \text{if case 2b occurs,} \end{cases}$$

in which $h' = N - 1 - 2h$.

Proof. By Algorithm 1, each P_i first computes a value h and transmits a personalized message to each node which is within a distance h from i. This process spends $\sum_{i=1}^{h} i$ time units. After the above process, $h' = N-1-2h$ processors have not receive a personalized message from P_i. If P_i sends messages to the remaining h' processors by using case 1a (lines 15 to 23), then

$$t = \sum_{i=1}^{h} i + \left(\sum_{i=0}^{\frac{h'-1}{2}} (2k + k_p + i) \right) = \frac{h(h+1)}{2} + \frac{(h'+1)(8k+4k_p+h'-1)}{8}.$$

If P_i sends messages to the remaining h' processors by using case 1b (lines 25 to 35), then

$$t = \sum_{i=1}^{h} i + \left(\sum_{i=0}^{\frac{h'-3}{2}} (2k + 2 + i) \right) + (2k+3) = \frac{h(h+1)}{2} + \frac{(h'-1)(8k+h'+5)}{8} + 2k + 3.$$

If P_i sends messages to the remaining h' processors by using case 1b (lines 38 to 48), then

$$t = \sum_{i=1}^{h} i + \left(\sum_{i=0}^{\frac{h'-1}{2}} (2k + k_p + 1 + i) \right) = \frac{h(h+1)}{2} + \frac{(h'-1)(8k+4k_p+h'-1)}{8}.$$

If P_i sends messages to the remaining h' processors by using case 1b (lines 49 to 59), then

$$t = \sum_{i=1}^{h} i + \sum_{i=0}^{\frac{h'-1}{2}} (2k + 2 + i) = \frac{h(h+1)}{2} + \frac{(h'+1)(8k+4k_p+h'+3)}{8}.$$

If P_i sends messages to the remaining h' processors by using case 1b (lines 64 to 72), then

$$t = \sum_{i=1}^{h} i + \sum_{i=1}^{N-w-h} i = \frac{h(h+1)}{2} + \frac{(N-w-h)(N-w-h+1)}{2}.$$

If P_i sends messages to the remaining h' processors by using case 1b (lines 74 to 91), then

$$t = \sum_{i=1}^{h} i + \left(\sum_{i=1}^{w-h+1} (i+1) \right) + \left(\sum_{i=1}^{\frac{N-2w-2}{2}} (2+i) \right) = \frac{h(h+1)}{2} + \frac{(w-h+4)(w-h+1)}{2} + \frac{(N-2w+8)(N-2w-2)}{8}.$$

∎

Before ending this section, we compare our algorithm with Algorithm A. Recall that Algorithm A takes $\sum_{i=1}^{\frac{N}{2}} i$ time units. On the other hand, our algorithm takes only t time units, where t is given in Theorem 2. Therefore, for a given $CR(N, w)$, the percentage of improvements of our algorithm is

$$\frac{\left(\sum_{i=1}^{\frac{N}{2}} i \right) - t}{\sum_{i=1}^{\frac{N}{2}} i}. \tag{1}$$

In Appendix A, we list the percentage of improvements of our algorithm for $N = 12, 14, \ldots, 2010$ (a total of 1000 N's and a total of 252000 chordal ring networks). In this appendix, N and w are the parameters of $CR(N, w)$, t_{our} is the number of time units used by Algorithm 1, t_A is the number of time units used by Algorithm A, and % is the percentage of improvements of Algorithm 1, which is calculated by (1). See Figure 8 for the curve and Figure 9 for the pie chart of the percentage of improvements of Algorithm 1.

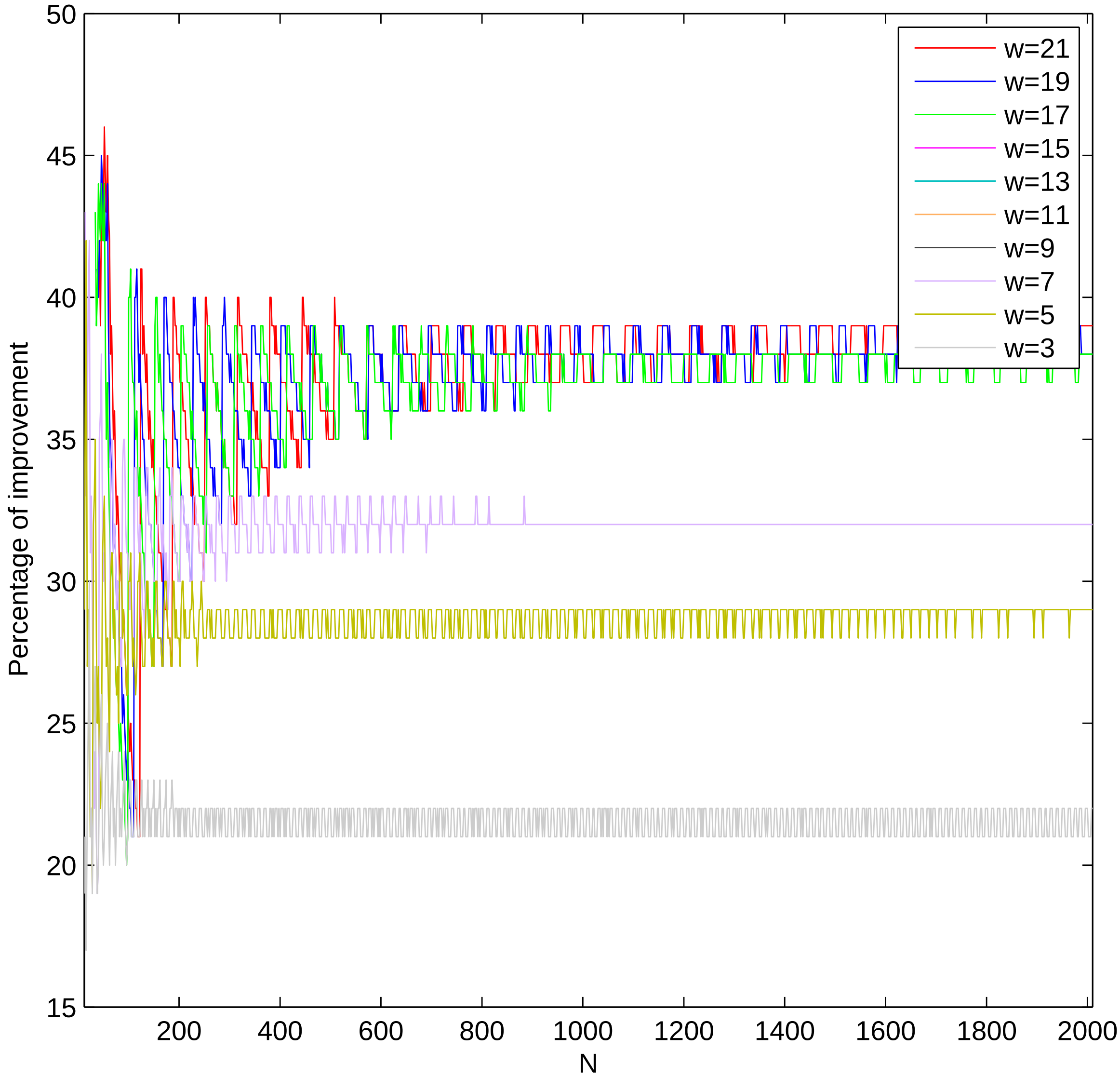

Figure 8: The percentage of improvements of Algorithm 1 for $N = 12, 14, \ldots, 2010$ (a total of 1000 N's) and $w = 3, 5, \ldots, 21$ (a total of 10 w's).

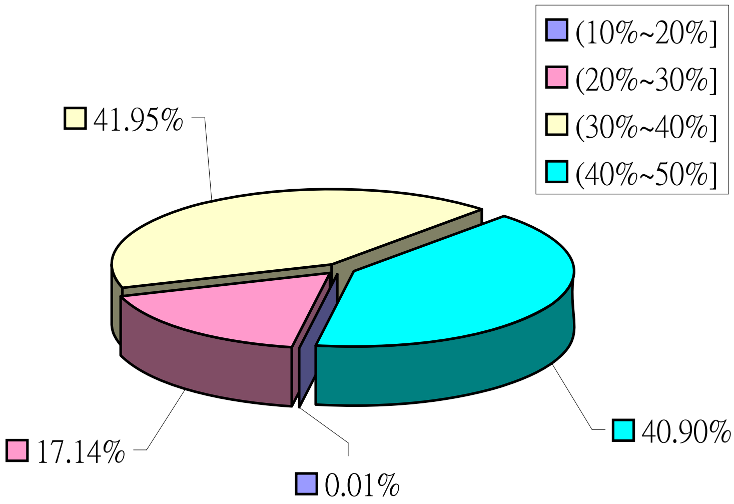

Figure 9: The pie chart of the percentage of improvements of Algorithm 1 for $N = 12, 14, \ldots, 2010$ (a total of 1000 N's) and $w = 3, 5, \ldots, 21$ (a total of 10 w's).

According to Figure 8, we find that the percentage of improvements of Algorithm 1 becomes better as w increases. Moreover, for each w, the curve of $CR(N, w)$ oscillates very seriously when when N varies from 12 to 200, and the curve has its period of oscillation when $N > 200$. Finally, the curve becomes smooth and approaches to a fixed value.

According to Figure 9, among the 252000 chordal ring networks, 40.90% of them get an improvement of at least 40%; 41.95% of them get an improvement of at least 30% and less than 40%; about 41.95% of them get an improvement of at least 30% but less than 40%; 17.14% of them get an improvement of at least 20% but less than 30%; and 0.01% of them get an improvement of at least 10% but less than 20%.

4 A special all-to-all personalized exchange algorithm for $CR(N, w)$ with $w = 3$

From Figure 8, we know that Algorithm 1 has the lowest percentage of improvements when $w = 3$. Therefore, in this section, we will design an all-to-all personalized exchange algorithm for chordal ring networks $CR(N, w)$ with $N \geq 10$ and $w = 3$. We find that when $N \geq 10$, $CR(N, 3)$ has a very nice structure and we make use of this nice structure to design a all-to-all personalized exchange algorithm, called Algorithm 2, such that the percentage of improvements of this algorithm is at least 50%.

To describe Algorithm 2, we define three types of links. Recall that we assume that the indices of processors are taken modulo N.

Definition : The links of $CR(N,3)$ is divided into three categories.

$$\begin{array}{lll} \text{a-link:} & (i,i+1), & i\!=\!0,2,4,\ldots,N-2, \\ & (i,i-1), & i\!=\!1,3,5,\ldots,N-1, \\ \text{b-link:} & (i,i-1), & i\!=\!0,2,4,\ldots,N-2, \\ & (i,i+1), & i\!=\!1,3,5,\ldots,N-1, \\ \text{c-link:} & (i,i-3), & i\!=\!0,2,4,\ldots,N-2, \\ & (i,i+3), & i\!=\!1,3,5,\ldots,N-1. \end{array}$$

We now give the structure of $CR(N,3)$. We observe that $CR(N,3)$ has only two possible structures, depending on whether N is divisible by 4. If $4 \mid N$, we call the structure *structure 1* and if $4 \nmid N$, we call the structure *structure 2*. For convenience, we use n to denote the number of layers which have exactly 4 nodes in structures 1 or 2 and we use $\mathcal{L}$ to denote the set of processors that appear in these n layers. See Figures 10 and 11; in these figures, a, b and c represent a-link, b-link and c-link, respectively.

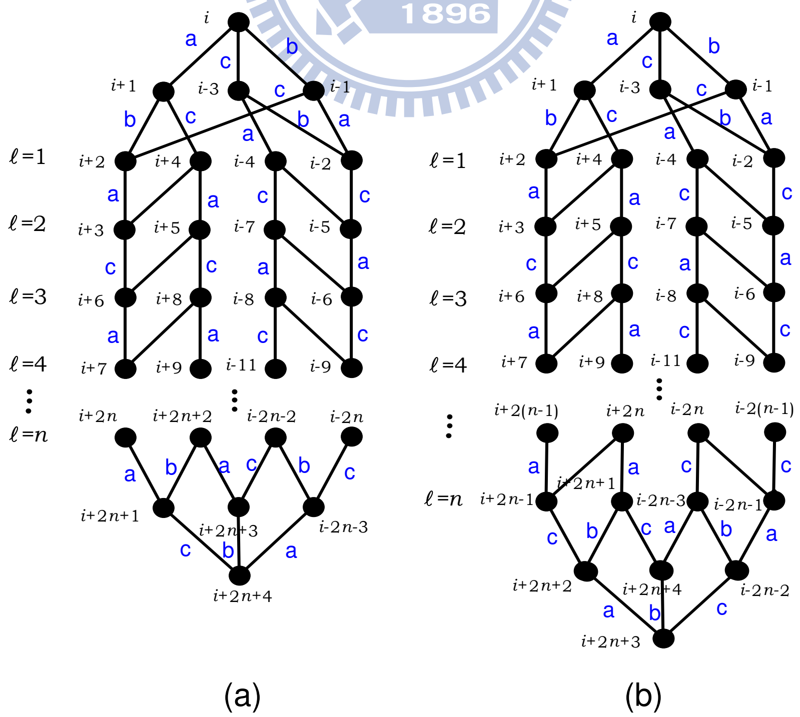

Figure 10: Structure 1 of $CR(N,3)$; here $4 \mid N$. (a) n is odd. (b) n is even.

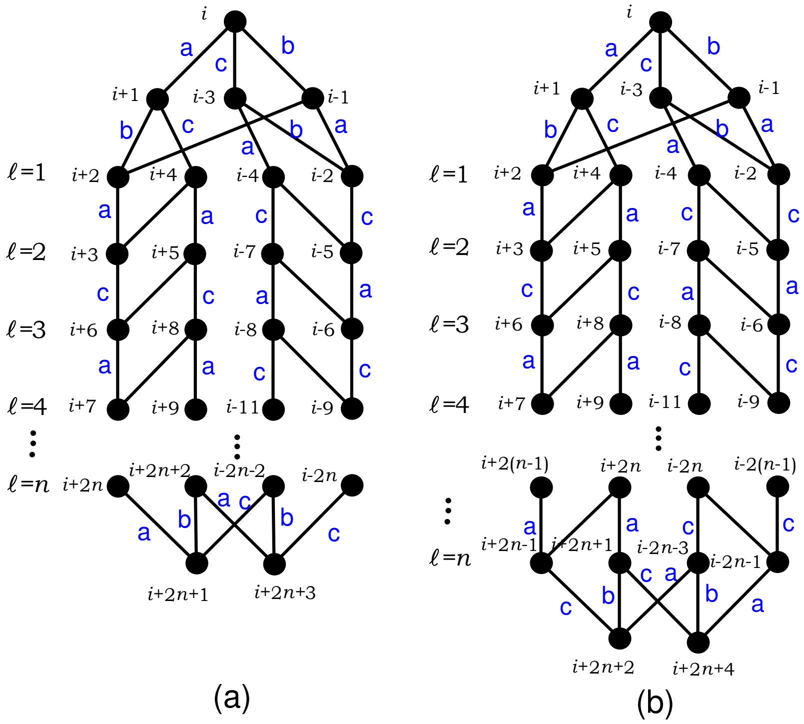

Figure 11: Structure 2 of $CR(N, 3)$; here $4 \nmid N$. (a) n is odd. (b) n is even.

It is not difficult to see from Figures 10 and 11 that each processor P_i may have several shortest paths to the remaining processors. We therefore give rules to restrict each P_i to have exactly one path from itself to each of the remaining processors. The rules will be stored in a table, called the *routing table*; see Table 1. Table 1 is the routing table for $CR(N, 3)$ with P_i as the source of all personalized messages. In Table 1, the numbers $1, 2, 3, \ldots$ represent the time unit; the notation "$j; k$" means that a personalized message for P_j is sent out from P_i and the next personalized message sent out from P_i through the same link is after k time units; the notation "$-$" means that P_i does not send out a personalized message by itself but P_i may help with transmitting another processor's personalized message.

Our all-to-all personalized exchange algorithm for chordal ring networks $CR(N, w)$ with $w = 3$ will use as its routing table to send out the messages; see Algorithm 2.

Table 1: Routing table for P_i when i is even; if i is odd, then replace a-link by b-link and replace b-link by a-link.

(i) $CR(N,3)$ is of structure 1 and n is odd

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			t_{p+1}		
a	$i+1$	$i+4$	–	–	$i+5$	–	–	–	–	$i+8$	–	–	–	–	–	–		–	$i+2n+3$	–	
b	$i-1$	$i+2$	$i-2;1$	–	$i+3$	$i-5;3$	–	–	–	$i+6$	$i-6;5$	–	–	–	–	–	...	$i+2n+4$	$i+2n+1$	$i-2n-3$	...
c	$i-3$	$i-4$	–	–	$i-7$	–	–	–	–	$i-8$	–	–	–	–	–	–		–	–	–	

(ii) $CR(N,3)$ is of structure 1 and n is even

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			t_{p+1}		
a	$i+1$	$i+4$	–	–	$i+5$	–	–	–	–	$i+8$	–	–	–	–	–	–		–	$i+2n+4$	–	
b	$i-1$	$i+2$	$i-2;1$	–	$i+3$	$i-5;3$	–	–	–	$i+6$	$i-6;5$	–	–	–	–	–	...	$i+2n+3$	$i+2n+2$	$i-2n-2$	...
c	$i-3$	$i-4$	–	–	$i-7$	–	–	–	–	$i-8$	–	–	–	–	–	–		–	–	–	

(iii) $CR(N,3)$ is of structure 2 and n is odd

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			$t_{p+1}-2$		
a	$i+1$	$i+4$	–	–	$i+5$	–	–	–	–	$i+8$	–	–	–	–	–	–		–	–	–	
b	$i-1$	$i+2$	$i-2;1$	–	$i+3$	$i-5;3$	–	–	–	$i+6$	$i-6;5$	–	–	–	–	–	...	–	$i+2n+1$	$i+2n+3$	...
c	$i-3$	$i-4$	–	–	$i-7$	–	–	–	–	$i-8$	–	–	–	–	–	–		–	–	–	

(iv) $CR(N,3)$ is of structure 2 and n is even

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			$t_{p+1}-2$		
a	$i+1$	$i+4$	–	–	$i+5$	–	–	–	–	$i+8$	–	–	–	–	–	–		–	–	–	
b	$i-1$	$i+2$	$i-2;1$	–	$i+3$	$i-5;3$	–	–	–	$i+6$	$i-6;5$	–	–	–	–	–	...	–	$i+2n+2$	$i+2n+4$	...
c	$i-3$	$i-4$	–	–	$i-7$	–	–	–	–	$i-8$	–	–	–	–	–	–		–	–	–	

Table 2: A table used to construct Table 1. Note that "final*" will be explained later.

	1	$t_1 = 2$			$t_2 = 5$			
a	a	Line 2			Line 2			
b	b	Line 1	Line 4	...	Line 1	Line 4	...	final*
c	c	Line 3			Line 3			

Table 1 is constructed due to the following observations. Form Figures 10 and 11, the processors in $\mathcal{L}$ can be divided into four groups (each group consists of processors that lie in the same vertical line) and we call them *Line 1*, *Line 2*, *Line 3* and *Line 4*. More precisely, Line 1 consists of $P_{i+2}, P_{i+3}, P_{i+6}$, and so on; Line 2 consists of $P_{i+4}, P_{i+5}, P_{i+8}$, and so on; Line 3 consists of $P_{i-4}, P_{i-7}, P_{i-8}$ and so on; and Line 4 consists of $P_{i-2}, P_{i-5}, P_{i-6}$, and so on. Notice that P_i has only three links; hence P_i can only send personalized messages to processors in three of the four lines.

Table 1 is constructed according to Table 2, which suggests processors on which lines of the four lines are selected. More precisely, in time unit 1, P_i transmits personalized

---

Algorithm 2 All-to-all personalized exchange algorithm for $CR(N,3)$, where $N \geq 10$.

Input: A chordal ring network $CR(N,3)$, where $N \geq 10$.
Output: All-to-all personalized exchange of $CR(N,3)$.

1: **for** P_i $(0 \leq i \leq N-1)$ **do in parallel**
2: P_i transmits personalize messages according to its routing table.
3: **end for**

---

messages to P_{i+1}, P_{i-1} and P_{i-3}. Started from time unit 2, P_i transmits personalized messages according to Table 2. Table 2 says that at time unit 2, P_i transmits personalized messages to a processor on Line 2, a processor on Line 1 and a processor on Line 3; at the next time unit, P_i transmits a personalized message to a processor on Line 4; at time unit 5, P_i transmits personalized messages to a processor on Line 2, a processor on Line 1 and a processor on Line 3; at the next time unit, P_i transmits a personalized message to a processor on Line 4. In general, at time unit t_p, P_i transmits personalized messages to a processor on Line 2, a processor on Line 1 and a processor on Line 3, and at the next time unit, P_i transmits a personalized message to a processor on Line 4. Here $t_p = t_{p-1} + w_p$, where $w_p = 2 \times (p-1) + 1$, $t_0 = 1$, and $1 \leq p \leq n$.

We assume that when a personalized message is sent out, a routing header is attached with the message and is used for routing the message. For example, $b(ac)^2a$ means that the routing path is via a b-link, then 2 pairs of (a-link,c-link), and 1 a-link. When a message is sent out from a processor, the first character of the routing header will be used in selecting the link used for sending the message; this character will be discarded then. Table 3 gives the routing headers for $CR(N,3)$ with P_i as the source of all personalized messages.

From Table 3, we know that there are only eleven possible forms for the routing headers. We now summarize them in Table 4 and give a binary notation for each of them.

Table 3: Routing header for P_i when i is even; if i is odd, then replace a-link by b-link and replace b-link by a-link.

(i) $CR(N,3)$ is of structure 1 and n is odd

	1	$t_1 = 2$	3	4	$t_2 = 5$				t_{p+1}		
a	a	$(ac)^m$	–	–	$(ac)^m a$	–		–	$(ac)^m a$	–	
b	b	$b(ca)^{m-1}c$	$b(ac)^{m-1}a$	–	$b(ca)^m$	$b(ac)^m$	...	$b(ca)^m c$	$b(ca)^m$	$b(ac)^m$	...
c	c	$(ca)^m$	–	–	$(ca)^m c$	–		–	–	–	

(ii) $CR(N,3)$ is of structure 1 and n is even

	1	$t_1 = 2$	3	4	$t_2 = 5$				t_{p+1}		
a	a	$(ac)^m$	–	–	$(ac)^m a$	–		–	$(ac)^m$	–	
b	b	$b(ca)^{m-1}c$	$b(ac)^{m-1}a$	–	$b(ca)^m$	$b(ac)^m$	...	$b(ca)^m$	$b(ca)^{m-1}c$	$b(ac)^{m-1}c$	...
c	c	$(ca)^m$	–	–	$(ca)^m c$	–		–	–	–	

(iii) $CR(N,3)$ is of structure 2 and n is odd

	1	$t_1 = 2$	3	4	$t_2 = 5$				$t_{p+1} - 2$		
a	a	$(ac)^m$	–	–	$(ac)^m a$	–		–	–	–	
b	b	$b(ca)^{m-1}c$	$b(ac)^{m-1}a$	–	$b(ca)^m$	$b(ac)^m$	...	–	$b(ca)^m$	$b(ac)^m$	...
c	c	$(ca)^m$	–	–	$(ca)^m c$	–		–	–	–	

(iv) $CR(N,3)$ is of structure 2 and n is even

	1	$t_1 = 2$	3	4	$t_2 = 5$				$t_{p+1} - 2$		
a	a	$(ac)^m$	–	–	$(ac)^m a$	–		–	–	–	
b	b	$b(ca)^{m-1}c$	$b(ac)^{m-1}a$	–	$b(ca)^m$	$b(ac)^m$	...	–	$b(ca)^{m-1}c$	$b(ac)^{m-1}a$	...
c	c	$(ca)^m$	–	–	$(ca)^m c$	–		–	–	–	

Table 4: Eleven forms of the routing headers.

a	b	c	$b(ca)^{m-1}c$	$b(ca)^m$	$(ac)^m$	$(ac)^m a$	$(ca)^m$	$(ca)^m c$	$b(ac)^{m-1}a$	$b(ac)^m$

We now explain "final*" in Table 2; it means the last personalized messages sent out from P_i; the destinations of these messages are given in Table 1. More precisely, when $CR(N,3)$ is of structure 1, the destinations of these messages are P_{i+2n+4}, P_{i+2n+3}, P_{i+2n+1} and P_{i-2n-3} if n is odd; the destinations are P_{i+2n+3}, P_{i+2n+4}, P_{i+2n+2} and P_{i-2n-2} if n is even. When $CR(N,3)$ is of structure 2, the destinations of these messages are P_{i+2n+1} and P_{i+2n+3} if n is odd; the destinations are P_{i+2n+2} and P_{i+2n+4} if n is even.

Table 5: The routing table with respected to P_0 of $CR(20,3)$. Note that the destinations of the last personalized messages sent out from P_0 are P_{10}, P_9, P_7 and P_{11}.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a	1	4	–	–	5	–	–	–	–	8	–	–	–	–	–	–	9	–	–	–	–	–	–	–	–
b	19	2	18;1	–	3	15;3	–	–	–	6	14;5	–	–	–	–	10	7	11	–	–	–	–	–	–	–
c	17	16	–	–	13	–	–	–	–	12	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Example. Consider $CR(20,3)$. It is of structure 1 and we illustrate its structure in Figure 12. Table 5 is the routing table with P_0 being the source node. According to Table 5, at time unit 1, P_0 sends personalized messages to P_1 via its a-link, to P_{19} via

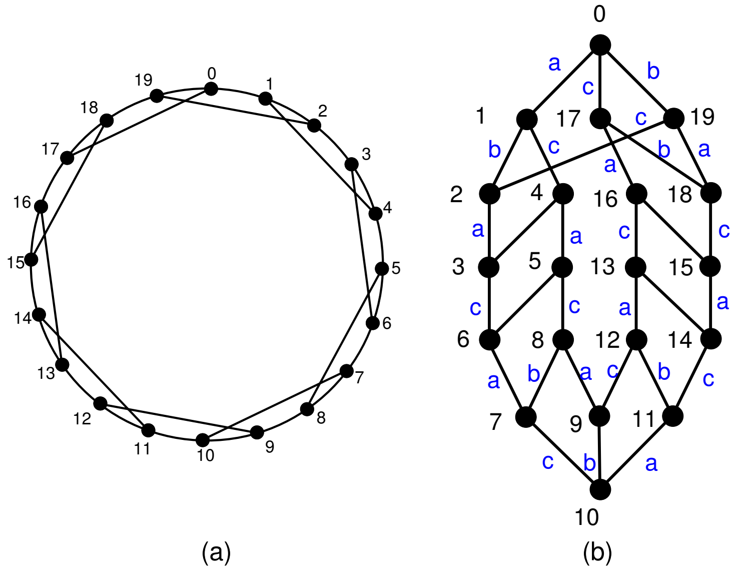

Figure 12: (a) The chordal ring network $CR(20,3)$. (b) $CR(20,3)$ is of structure 1.

its b-link and to P_{17} via its c-link. At time unit 2, P_0 sends personalized messages to P_4 via its a-link, to P_2 via its b-link and to P_{16} via its c-link. At time unit 3, P_0 only sends a personalized message out to P_{18} via its b-link; at the same time unit, P_0 helps with transmitting personalized messages of other processors to their destinations. Similarly, at time unit 5, P_0 sends personalized messages to P_5 via its a-link, to P_3 via its b-link and to P_{13} via its c-link. At time unit 6, P_0 only sends a personalized message out to P_{15} via its b-link; at the same time, P_0 helps with transmitting personalized messages of other processors to their destinations.

Let n be the number of layers in $CR(N,3)$, where $N \geq 10$. By Figures 10 and Figure 11, we know that if $CR(N,3)$ is of structure 1, then $N = 4n+8$ and $n \in \{1,2,\ldots\}$; on the other hand, if $CR(N,3)$ is of structure 2, then $N = 4n+6$ and $n \in \{1,2,\ldots\}$. Hence

$$n = \begin{cases} \frac{N-8}{4} & \text{if } CR(N,3) \text{ is of structure 1,} \\ \frac{N-6}{4} & \text{if } CR(N,3) \text{ is of structure 2.} \end{cases} \tag{2}$$

Let diam denote the diameter of $CR(N,w)$. By Figures 10 and Figure 11,

$$\text{diam} = \begin{cases} n+3 & \text{if } CR(N,3) \text{ is of structure 1,} \\ n+2 & \text{if } CR(N,3) \text{ is of structure 2.} \end{cases}$$

We now prove the correctness of Algorithm 2.

Theorem 3. *Algorithm 2 fulfills an all-to-all personalized exchange for $CR(N,3)$, where $N \geq 10$.*

Proof. Suppose $CR(N,3)$ has n layers and it is of structure 1. Then $N = 4n+8$. It suffices to prove that each P_i sends out $N-1$ personalized messages out. By Table 1, each P_i sends out 3 personalized messages out at time unit 1. After that, each P_i sends out $4n$ personalized messages to the processors in $\mathcal{L}$ (this can be observed from Table 2), and finally, each P_i sends out 4 personalized messages out according to "final*". From the above, each P_i sends out $3+4n+4 = N-1$ personalized messages out. Since the routing headers of these $N-1$ messages are obtained from the structure of $CR(N,3)$, these messages can use the routing headers to arrive their destinations. The case that $CR(N,3)$ is of structure 2 can be proven similarly. ∎.

We now analyze the number of time units used by Algorithm 2.

Theorem 4. *Algorithm 2 uses $\frac{N^2}{16}$ time units if the given $CR(N,3)$ is of structure 1 and $\frac{N^2-4}{16}$ time units if the given $CR(N,3)$ is of structure 2.*

Proof. By Figures 10 and 11, it is not difficult to see that there is only one path from P_i to each processor on Lines 2 or 3 and this unique path utilizes only a-links and c-links. Our routing table restricts that P_i arrives processors on Line 1 by using a b-Link followed by a c-link and a series of a-links and c-links. Our routing table also restricts that P_i arrives processors on Line 4 by using a b-Link followed by an a-link and a series of c-links and a-links. Notice that only the b-link of P_i will be used and the b-links of other processors will not be used. Thus the number of b-links used by our algorithm is much less than the numbers of a-links and c-links. Therefore the number of time units used by our algorithm is dominated by the number of pairs of (a-link,c-link). Thus

$$\begin{aligned}
\text{total time units of structure 1} &= \Big(\sum_{i=1}^{n+1} i\Big) - 1 + \Big(\sum_{j=1}^{n+2} j\Big) + 1 = \sum_{i=1}^{n+1} i + \sum_{j=1}^{n+2} j \\
&= \sum_{i=1}^{n+1} i + \Big(\sum_{j=1}^{n+1} j\Big) + (n+2) = 2 \times \Big(\sum_{i=1}^{n+1} i\Big) + (n+2) \\
&= (n+2)^2 \overset{(2)}{=} \frac{N^2}{16}, \\
\text{total time units of structure 2} &= \Big(\sum_{i=1}^{n+1} i\Big) - 1 + \Big(\sum_{j=1}^{n+1} j\Big) + 1 = \sum_{i=1}^{n+1} i + \sum_{j=1}^{n+1} j \\
&= 2 \times \sum_{i=1}^{n+1} i = (n+1)(n+2) \overset{(2)}{=} \frac{N^2-4}{16}.
\end{aligned}$$

∎

We now give the impact of Algorithm 2.

Corollary 5. *The percentage of improvements of Algorithm 2 is more than 50%.*

Proof. Recall that the number of time units used by Algorithm A is $\sum_{i=1}^{\frac{N}{2}} i$, which is $\frac{N^2}{8} + \frac{N}{4}$. By (1), the percentage of improvements of Algorithm 2 is at least

$$\frac{\frac{N^2}{8} + \frac{N}{4} - \frac{N^2}{16}}{\frac{N^2}{8} + \frac{N}{4}} = \frac{N+4}{2N+4} > \frac{1}{2}.$$

Thus we have this corollary. ∎

To conclude this section, we summarize the routing headers of all the processors in Table 6 when the given chordal ring network is of structure 1 and when the source is P_i for i even.

5 About Algorithm B–unclear parts and incorrect parts

In [5, 7], Masuyama et al. proposed an all-to-all broadcast algorithm for chordal ring networks $CR(N, w)$ and we call it Algorithm B. The purpose of this section is to clarify some unclear parts and correct some incorrect parts in Algorithm B. Note that Algorithm B can handle a single fault or double faults. Thus this section consists of two

Table 6: The routing header for each processor (except P_{i+1}, P_{i-1}, and P_{i-3}, which will use only one a-link or b-link or c-link) when $CR(N,3)$ is of structure 1, P_i is the source and i is even.

$m\diagdown$ forms	$b(ca)^{m-1}c$	$b(ca)^m$	$(ac)^m$	$(ac)^m a$	$(ca)^m$	$(ca)^m c$	$b(ac)^{m-1}a$	$b(ac)^m$
1	$i+2$	$i+3$	$i+4$	$i+5$	$i-4$	$i-7$	$i-2$	$i-5$
2	$i+6$	$i+7$	$i+8$	$i+9$	$i-8$	$i-11$	$i-6$	$i-9$
3	$i+10$	$i+11$	$i+12$	$i+13$	$i-12$	$i-15$	$i-10$	$i-13$
4	$i+14$	$i+15$	$i+16$	$i+17$	$i-16$	$i-19$	$i-14$	$i-17$
...								
n (odd)	$i+2n$	$i+2n+1$	$i+2n+2$	$i+2n+3$	$i-2n-2$		$i-2n$	$i-2n-3$
$n+1$		$i+2n+4$						
...								
n (even)	$i+2(n-1)$	$i+2n-1$	$i+2n$	$i+2n+1$	$i-2n$	$i-2n-3$	$i-2(n-1)$	$i-2n-1$
$n+1$	$i+2n+2$	$i+2n+3$	$i+2n+4$				$i-2n-2$	

subsections: in subsection 5.1, the single-fault part of Algorithm B is discussed and in subsection 5.2, the double-fault part of Algorithm B is discussed.

5.1 About the single-fault part of Algorithm B

Notice that all the processors on $\overline{R(1)}^f$ are non-faulty. When two messages arrive at a processor on $\overline{R(1)}^f$ at the same time unit, Algorithm B only sends out the message from the larger-numbered processor; Algorithm B does not say anything about how to handle the message from the smaller-numbered processor. This is an unclear part of Algorithm B. Moreover it takes $N-(w-1)+\lceil (w-1)/2 \rceil$ time units. We now give a new all-to-all broadcast algorithm for the single-fault case. Our algorithm consists of two phases.

- In phase 1, all the processors on $\overline{R(1)}^f$ broadcast their messages.

  More precisely, in phase 1, all the processors on $\overline{R(1)}^f$ broadcast their messages to processors on $\overline{R(1)}^f$ and to non-faulty processors on $R(1)^f$ as follows. Each processor on $\overline{R(1)}^f$ transmits its own message to processors on $\overline{R(1)}^f$ by using ring-links and to its adjacent processor on $R(1)^f$ by using chordal-link at the same time. If processor P_j on $\overline{R(1)}^f$ has a non-faulty adjacent processor P_k on $R(1)^f$ through a chordal-link, then P_j transmits the message received from the larger-numbered adjacent processor to P_k; this takes only 1 time unit. As for the message received from the smaller-numbered adjacent processor, P_j utilizes $R(1)^{start}$ (or $R(1)^{end}$) to transmit the message to P_k; this takes $\frac{w-1}{2}$ time units. From the above, phase 1 takes at most $\frac{N-(w-1)}{2} + \frac{w-1}{2}$ time units.

- In phase 2, all the non-faulty processors on $R(1)^f$ broadcast their messages.

  More precisely, in phase 2, all the non-faulty processors on $R(1)^f$ broadcast their messages to processors on $\overline{R(1)}^f$ and to non-faulty processors on $R(1)^f$ as follows. Each non-faulty processor on $R(1)^f$ transmits its own message to processors on $R(1)^f$ by using ring-links and to its adjacent processor on $\overline{R(1)}^f$ by using chordal-link at the same time. Note that transmitting through the chordal-link is done only once and after the message is sent to the adjacent processor on $\overline{R(1)}^f$, this message will be transmitted to all the processors on $\overline{R(1)}^f$ by using ring-links. From the above, phase 2 takes at most $\frac{N-(w-1)}{2} + \frac{w-1}{2}$ time units.

Our new algorithm uses at most N time units to fulfill an all-to-all broadcast on $CR(N, w)$.

5.2 About the double-fault part of Algorithm B

When there are two faults, Algorithm B also has some unclear or incorrect parts. For convenience, let f_1 and f_2 be the two faulty processors and let $d(f_1, f_2)$ be the distance between f_1 and f_2. There are two cases.

Case 1: $1 \leq d(f_1, f_2) \leq w$.

Algorithm B says that without losing generality, we can assume that there is an $R(1)$ containing both f_1 and f_2. This is incorrect. Take Figure 3 for an example. If $f_1 = P_0$ and $f_2 = P_5$, then there is *no* $R(1)$ containing both f_1 and f_2. Even if there is an $R(1)$ containing both f_1 and f_2, Algorithm B still has problems. In the following, we will assume that there is an $R(1)$ containing both f_1 and f_2.

1: When $f_2 = f_1 + 1$, it is unclear how an all-to-all broadcast is fulfilled by Algorithm B. The reason is that when $f_1 + 1 = f_2$, Algorithm B considers the two consecutive faulty processors as a single faulty processor and uses the single-fault part to fulfill the all-to-all broadcast. Since the single-fault part is unclear and actually incomplete (see subsection 5.1), Algorithm B is incorrect. Moreover, it takes $1 + N - (w + 1) +$

$\lceil w-2 \rceil = N-2$ time units. If our algorithm in subsection 5.1 is used instead of Algorithm B, then it takes $(\frac{N-(w-1)}{2} + \lfloor \frac{w-2}{2} \rfloor) + (\frac{N-(w-1)}{2} + \lfloor \frac{w-2}{2} \rfloor) = N-2$ time units to fulfill an all-to-all broadcast.

2: When $f_2 = f_1 + 2$, it is unclear how an all-to-all broadcast is fulfilled by Algorithm B in the mentioned $\max\{N-3, N-(w-1)+1\}$ time units. Again, take Figure 3 for an example. If $f_1 = P_2$ and $f_2 = P_4$, then P_3 can only use its chordal-link to transmit or receive messages. If only $\max\{N-3, N-(w-1)+1\}$ time units are used, then in Algorithm B, after P_3 transmits its own message to P_8, P_8 has to transmit this message to P_7 and P_9 (hence the ring-link between P_8 and P_9 must be used); however, P_7 has to transmit its own message to P_9 also via the ring-link between P_8 and P_9 and a conflicting between the messages occurs. For convenience, let P_k be the unique non-faulty processor between f_1 and f_2. We now fix the above problem. More precisely, we think about the double faults as a single fault by considering f_1, f_2 and P_k as a faulty processor; so we can apply our results for the single-fault part. It takes at most $N-2$ time units to fulfill all-to-all broadcast on the remaining $N-3$ processors. Notice that while these $N-3$ processors perform their all-to-all broadcast, they also broadcast to P_k. Otherwise, when the non-faulty processors on $R(1)$ broadcast their messages, P_k also broadcasts its message.

Case 2: $w+1 \leq d(f_1, f_2) \leq 2w+1$.

Algorithm B divides case 2 into two subcases: $d(f_1, f_2) = w+1$ and $w+1 < d(f_1, f_2) \leq 2w+1$. Algorithm B is correct; however, we find that there is no need to divide case 2 into two subcases. In the following, we propose a method that can handle the two subcases in a unified way. We will prove that when $d(f_1, f_2) = w+1$ and $w > 3$, our method uses less time units than Algorithm B. Since $d(f_1, f_2) \geq w+1$, f_1 and f_2 belong to two different, isolated $R(1)$'s. See Figure 13 for an illustration. For convenience, denote the two $R(1)$'s by R_a and R_b. Let R_a^{start} and R_a^{end} be the two special marks of R_a. Similarly, let R_b^{start} and R_b^{end} be the two special marks of R_b. Define a loop R_1 as follows. Let R_1 be the loop which is composed of the chordal-link of R_a, the processors and ring-links between R_a^{end}

and R_b^{start}, the chordal-link of R_b, the processors and ring-links between R_b^{end} and R_a^{start}. R_1 can be handled as phase 1 in the method we proposed for the single-fault part; R_a and R_b can be handled as phase 2 in the method we proposed for the single-fault part.

It is said in [5, 7] that Algorithm B takes $N-3$ time units when $d(f_1, f_2) = w+1$. As for our method, handling R_1 takes $\frac{N-2w+2}{2} + \frac{w-1}{2}$ time units and handling R_a and R_b takes $\frac{N-2w+2}{2} + \frac{w-1}{2}$ time units. Thus our method takes a total of $N-w+1$ time units, which improves the $d(f_1, f_2) = w+1$ subcase of Algorithm B.

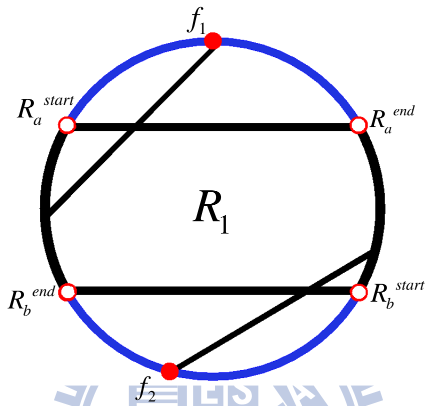

Figure 13: An illustration for case 2.

6 Conclusion

In [5, 7], Masuyama et al. proposed two all-to-all communication algorithms for chordal ring networks of degree 3 and we call them Algorithms A and B. Algorithm A is an all-to-all personalized exchange algorithm and it is used when there is no fault. Algorithm B is an all-to-all broadcast algorithm and it can tolerate one or two faults. A huge amount of hardware is wasted since Algorithm A only utilizes ring-links and Algorithm B utilizes chordal-links only when there are faults. In this thesis, we improve Algorithm A by utilizing chordal-links. We have tested 252000 chordal ring networks and found that among these chordal ring networks, 82.85% of them get an improvement of at least 30%; moreover, the percentage of improvements of our algorithm becomes better and better as w increases. We also provide an all-to-all personalized exchange algorithm that works only for chordal ring networks with $w = 3$ and we prove that the percentage of improvements of this special algorithm is at least 50%. There are some unclear parts and incorrect parts in Algorithm B; for example, it does not take into consideration that some broadcasting messages collide. In this thesis, we clarify the unclear parts and correct the incorrect parts in Algorithm B. In particular, we divide the transmitting process into phases to avoid collisions.

References

[1] B. W. Arden and H. Lee, Analysis of chordal ring network networks, IEEE Trans. Comput. C-30 (1981) 291-295.

[2] L. Barrière, Symmetry properties of chordal rings of degree 3, Discrete Applied Mathematics 129 (2003) 211-232.

[3] G. Chartrand and L. Lensniak, Graph and Digraphs, Wadsworth, Monterey, CA, 1981.

[4] S.L. Johnsson and C.-T. Ho, Optimum broadcasting and personalized communication in Hypercubes, IEEE Trans. Comput. 38 (9) (1989) 1249-1268.

[5] H. Masuyama, H. Taniguchi, and T. Miyoshi, All-to-All Personalized Communication Algorithms in Chordal Ring Networks, ICN. LNCS 2094, Springer-Verlag Berlin Heidlberg (2001) 877-889.

[6] L. Narayanan, J.Opatrny, and D. Sotteau, All-to-all optical routing in chordal ring networks of degree 4, Algorithmica 31 (2001) 155-178.

[7] H. Taniguchi, All-to-all broadcasting and personalized communication algorithms in chordal ring networks, Master Thesis, Info. and Know. Eng. Tottori Univ. (2001).

[8] D. B. West, Introduction to Graph Theory, 2nd ed. Prentice Hall, Upper Saddle River, NJ, 2001.

[9] Y. Yang, J. Wang, Optimal all-to-all personalized exchange in self-routable multistage networks, IEEE Trans. Parallel Distrib. Syst. 11 (3) (2000) 261-274.

A Appendix: the percentage of improvements

N	w	t_{our}	t_A	%
12	3	17	21	19
12	5	15	21	29
14	3	22	28	21
14	5	18	28	36
14	7	16	28	43
16	3	30	36	17
16	5	21	36	42
16	7	24	36	33
18	3	35	45	22
18	5	33	45	27
18	7	29	45	36
18	9	27	45	40
20	3	41	55	25
20	5	39	55	29
20	7	33	55	40
20	9	35	55	36
22	3	46	66	30
22	5	50	66	24
22	7	38	66	42
22	9	40	66	39
22	11	38	66	42
24	3	62	78	21
24	5	62	78	21
24	7	54	78	31
24	9	48	78	38
24	11	50	78	36
26	3	71	91	22
26	5	71	91	22
26	7	61	91	33
26	9	53	91	42
26	11	53	91	42
26	13	51	91	44
28	3	85	105	19
28	5	85	105	19
28	7	75	105	29
28	9	60	105	43
28	11	61	105	42
28	13	65	105	38
30	3	92	120	23
30	5	82	120	32
30	7	90	120	25
30	9	80	120	33
30	11	72	120	40
30	13	72	120	40
30	15	70	120	42
32	3	101	136	26
32	5	91	136	33
32	7	106	136	22
32	9	96	136	29
32	11	81	136	40
32	13	82	136	40
32	15	86	136	37
34	3	111	153	27
34	5	99	153	35
34	7	117	153	24
34	9	105	153	31
34	11	87	153	43
34	13	87	153	43
34	15	89	153	42
34	17	87	153	43
36	3	135	171	21
36	5	123	171	28
36	7	135	171	21
36	9	123	171	28
36	11	111	171	35
36	13	101	171	41
36	15	99	171	42
36	17	105	171	39
38	3	154	190	19
38	5	142	190	25
38	7	154	190	19
38	9	142	190	25
38	11	130	190	32
38	13	112	190	41
38	15	112	190	41
38	17	114	190	40
38	19	112	190	41
40	3	168	210	20
40	5	154	210	27
40	7	168	210	20
40	9	154	210	27
40	11	140	210	33
40	13	119	210	43
40	15	118	210	44
40	17	118	210	44
40	19	126	210	40
42	3	177	231	23
42	5	175	231	24
42	7	149	231	35
42	9	175	231	24
42	11	161	231	30
42	13	147	231	36

N	w	t_{our}	t_A	%
42	15	135	231	42
42	17	131	231	43
42	19	135	231	42
42	21	133	231	42
44	3	190	253	25
44	5	197	253	22
44	7	162	253	36
44	9	197	253	22
44	11	183	253	28
44	13	169	253	33
44	15	148	253	42
44	17	147	253	42
44	19	147	253	42
44	21	155	253	39
46	3	204	276	26
46	5	212	276	23
46	7	172	276	38
46	9	212	276	23
46	11	196	276	29
46	13	180	276	35
46	15	156	276	43
46	17	154	276	44
46	19	152	276	45
46	21	158	276	43
46	23	156	276	43
48	3	236	300	21
48	5	206	300	31
48	7	204	300	32
48	9	236	300	21
48	11	220	300	27
48	13	204	300	32
48	15	188	300	37
48	17	174	300	42
48	19	168	300	44
48	21	170	300	43
48	23	180	300	40
50	3	261	325	20
50	5	221	325	32
50	7	229	325	30
50	9	261	325	20
50	11	245	325	25
50	13	229	325	30
50	15	213	325	34
50	17	189	325	42
50	19	187	325	42
50	21	185	325	43
50	23	191	325	41
50	25	189	325	42
52	3	279	351	21
52	5	234	351	33
52	7	243	351	31
52	9	279	351	21
52	11	261	351	26
52	13	243	351	31
52	15	225	351	36
52	17	198	351	44
52	19	195	351	44
52	21	191	351	46
52	23	195	351	44
52	25	207	351	41
54	3	290	378	23
54	5	270	378	29
54	7	270	378	29
54	9	236	378	38
54	11	288	378	24
54	13	270	378	29
54	15	252	378	33
54	17	234	378	38
54	19	218	378	42
54	21	210	378	44
54	23	210	378	44
54	25	218	378	42
54	27	216	378	43
56	3	307	406	24
56	5	298	406	27
56	7	298	406	27
56	9	253	406	38
56	11	316	406	22
56	13	298	406	27
56	15	280	406	31
56	17	262	406	35
56	19	235	406	42
56	21	232	406	43
56	23	228	406	44
56	25	232	406	43
56	27	244	406	40
58	3	325	435	25
58	5	315	435	28
58	7	315	435	28
58	9	265	435	39
58	11	335	435	23
58	13	315	435	28

N	w	t_{our}	t_A	%
58	15	295	435	32
58	17	275	435	37
58	19	245	435	44
58	21	241	435	45
58	23	235	435	46
58	25	237	435	46
58	27	247	435	43
58	29	245	435	44
60	3	365	465	22
60	5	345	465	26
60	7	345	465	26
60	9	305	465	34
60	11	365	465	22
60	13	345	465	26
60	15	325	465	30
60	17	305	465	34
60	19	285	465	39
60	21	267	465	43
60	23	257	465	45
60	25	255	465	45
60	27	261	465	44
60	29	275	465	41
62	3	396	496	20
62	5	376	496	24
62	7	376	496	24
62	9	336	496	32
62	11	396	496	20
62	13	376	496	24
62	15	356	496	28
62	17	336	496	32
62	19	316	496	36
62	21	286	496	42
62	23	282	496	43
62	25	276	496	44
62	27	278	496	44
62	29	288	496	42
62	31	286	496	42
64	3	410	528	22
64	5	370	528	30
64	7	350	528	34
64	9	368	528	30
64	11	330	528	38
64	13	408	528	23
64	15	388	528	27
64	17	368	528	30
64	19	348	528	34
64	21	328	528	38
64	23	310	528	41
64	25	300	528	43
64	27	298	528	44
64	29	304	528	42
64	31	318	528	40
66	3	431	561	23
66	5	387	561	31
66	7	365	561	35
66	9	385	561	31
66	11	343	561	39
66	13	429	561	24
66	15	407	561	27
66	17	385	561	31
66	19	363	561	35
66	21	341	561	39
66	23	321	561	43
66	25	309	561	45
66	27	305	561	46
66	29	309	561	45
66	31	321	561	43
66	33	319	561	43
68	3	452	595	24
68	5	408	595	31
68	7	386	595	35
68	9	419	595	30
68	11	364	595	39
68	13	463	595	22
68	15	441	595	26
68	17	419	595	30
68	19	397	595	33
68	21	375	595	37
68	23	342	595	43
68	25	337	595	43
68	27	329	595	45
68	29	329	595	45
68	31	337	595	43
68	33	353	595	41
70	3	498	630	21
70	5	454	630	28
70	7	432	630	31
70	9	454	630	28
70	11	410	630	35
70	13	498	630	21
70	15	476	630	24
70	17	454	630	28

N	w	t_{our}	t_A	%
70	19	432	630	31
70	21	410	630	35
70	23	388	630	38
70	25	368	630	42
70	27	356	630	43
70	29	352	630	44
70	31	356	630	43
70	33	368	630	42
70	35	366	630	42
72	3	522	666	22
72	5	474	666	29
72	7	450	666	32
72	9	474	666	29
72	11	426	666	36
72	13	522	666	22
72	15	498	666	25
72	17	474	666	29
72	19	450	666	32
72	21	426	666	36
72	23	402	666	40
72	25	380	666	43
72	27	366	666	45
72	29	360	666	46
72	31	362	666	46
72	33	372	666	44
72	35	390	666	41
74	3	559	703	20
74	5	511	703	27
74	7	487	703	31
74	9	511	703	27
74	11	463	703	34
74	13	559	703	20
74	15	535	703	24
74	17	511	703	27
74	19	487	703	31
74	21	463	703	34
74	23	439	703	38
74	25	403	703	43
74	27	397	703	44
74	29	387	703	45
74	31	385	703	45
74	33	391	703	44
74	35	405	703	42
74	37	403	703	43
76	3	575	741	22
76	5	549	741	26
76	7	525	741	29
76	9	549	741	26
76	11	501	741	32
76	13	455	741	39
76	15	573	741	23
76	17	549	741	26
76	19	525	741	29
76	21	501	741	32
76	23	477	741	36
76	25	453	741	39
76	27	431	741	42
76	29	417	741	44
76	31	411	741	45
76	33	413	741	44
76	35	423	741	43
76	37	441	741	40
78	3	600	780	23
78	5	572	780	27
78	7	546	780	30
78	9	572	780	27
78	11	520	780	33
78	13	470	780	40
78	15	598	780	23
78	17	572	780	27
78	19	546	780	30
78	21	520	780	33
78	23	494	780	37
78	25	468	780	40
78	27	444	780	43
78	29	428	780	45
78	31	420	780	46
78	33	420	780	46
78	35	428	780	45
78	37	444	780	43
78	39	442	780	43
80	3	625	820	24
80	5	612	820	25
80	7	586	820	29
80	9	612	820	25
80	11	560	820	32
80	13	495	820	40
80	15	638	820	22
80	17	612	820	25
80	19	586	820	29
80	21	560	820	32
80	23	534	820	35

N	w	t_{our}	t_A	%
80	25	508	820	38
80	27	469	820	43
80	29	462	820	44
80	31	450	820	45
80	33	446	820	46
80	35	450	820	45
80	37	462	820	44
80	39	482	820	41
82	3	679	861	21
82	5	603	861	30
82	7	627	861	27
82	9	551	861	36
82	11	601	861	30
82	13	549	861	36
82	15	679	861	21
82	17	653	861	24
82	19	627	861	27
82	21	601	861	30
82	23	575	861	33
82	25	549	861	36
82	27	523	861	39
82	29	499	861	42
82	31	483	861	44
82	33	475	861	45
82	35	475	861	45
82	37	483	861	44
82	39	499	861	42
82	41	497	861	42
84	3	707	903	22
84	5	625	903	31
84	7	651	903	28
84	9	569	903	37
84	11	623	903	31
84	13	567	903	37
84	15	707	903	22
84	17	679	903	25
84	19	651	903	28
84	21	623	903	31
84	23	595	903	34
84	25	567	903	37
84	27	539	903	40
84	29	513	903	43
84	31	495	903	45
84	33	485	903	46
84	35	483	903	47
84	37	489	903	46
84	39	503	903	44
84	41	525	903	42
86	3	750	946	21
86	5	652	946	31
86	7	694	946	27
86	9	596	946	37
86	11	666	946	30
86	13	610	946	36
86	15	750	946	21
86	17	722	946	24
86	19	694	946	27
86	21	666	946	30
86	23	638	946	33
86	25	610	946	36
86	27	582	946	38
86	29	540	946	43
86	31	532	946	44
86	33	518	946	45
86	35	512	946	46
86	37	514	946	46
86	39	524	946	45
86	41	542	946	43
86	43	540	946	43
88	3	768	990	22
88	5	710	990	28
88	7	656	990	34
88	9	654	990	34
88	11	710	990	28
88	13	654	990	34
88	15	600	990	39
88	17	766	990	23
88	19	738	990	25
88	21	710	990	28
88	23	682	990	31
88	25	654	990	34
88	27	626	990	37
88	29	598	990	40
88	31	572	990	42
88	33	554	990	44
88	35	544	990	45
88	37	542	990	45
88	39	548	990	45
88	41	562	990	43
88	43	584	990	41
90	3	797	1035	23
90	5	735	1035	29

N	w	t_{our}	t_A	%
90	7	677	1035	35
90	9	675	1035	35
90	11	735	1035	29
90	13	675	1035	35
90	15	617	1035	40
90	17	795	1035	23
90	19	765	1035	26
90	21	735	1035	29
90	23	705	1035	32
90	25	675	1035	35
90	27	645	1035	38
90	29	615	1035	41
90	31	587	1035	43
90	33	567	1035	45
90	35	555	1035	46
90	37	551	1035	47
90	39	555	1035	46
90	41	567	1035	45
90	43	587	1035	43
90	45	585	1035	43
92	3	826	1081	24
92	5	781	1081	28
92	7	706	1081	35
92	9	721	1081	33
92	11	781	1081	28
92	13	721	1081	33
92	15	646	1081	40
92	17	841	1081	22
92	19	811	1081	25
92	21	781	1081	28
92	23	751	1081	31
92	25	721	1081	33
92	27	691	1081	36
92	29	661	1081	39
92	31	616	1081	43
92	33	607	1081	44
92	35	591	1081	45
92	37	583	1081	46
92	39	583	1081	46
92	41	591	1081	45
92	43	607	1081	44
92	45	631	1081	42
94	3	888	1128	21
94	5	828	1128	27
94	7	768	1128	32
94	9	768	1128	32
94	11	828	1128	27
94	13	768	1128	32
94	15	708	1128	37
94	17	888	1128	21
94	19	858	1128	24
94	21	828	1128	27
94	23	798	1128	29
94	25	768	1128	32
94	27	738	1128	35
94	29	708	1128	37
94	31	678	1128	40
94	33	650	1128	42
94	35	630	1128	44
94	37	618	1128	45
94	39	614	1128	46
94	41	618	1128	45
94	43	630	1128	44
94	45	650	1128	42
94	47	648	1128	43
96	3	936	1176	20
96	5	876	1176	26
96	7	816	1176	31
96	9	816	1176	31
96	11	876	1176	26
96	13	816	1176	31
96	15	756	1176	36
96	17	936	1176	20
96	19	906	1176	23
96	21	876	1176	26
96	23	846	1176	28
96	25	816	1176	31
96	27	786	1176	33
96	29	756	1176	36
96	31	726	1176	38
96	33	681	1176	42
96	35	672	1176	43
96	37	656	1176	44
96	39	648	1176	45
96	41	648	1176	45
96	43	656	1176	44
96	45	672	1176	43
96	47	696	1176	41
98	3	969	1225	21
98	5	905	1225	26
98	7	841	1225	31
98	9	841	1225	31

N	w	t_{our}	t_A	%
98	11	905	1225	26
98	13	841	1225	31
98	15	777	1225	37
98	17	969	1225	21
98	19	937	1225	24
98	21	905	1225	26
98	23	873	1225	29
98	25	841	1225	31
98	27	809	1225	34
98	29	777	1225	37
98	31	745	1225	39
98	33	697	1225	43
98	35	687	1225	44
98	37	669	1225	45
98	39	659	1225	46
98	41	657	1225	46
98	43	663	1225	46
98	45	677	1225	45
98	47	699	1225	43
98	49	697	1225	43
100	3	989	1275	22
100	5	893	1275	30
100	7	891	1275	30
100	9	891	1275	30
100	11	797	1275	37
100	13	891	1275	30
100	15	827	1275	35
100	17	765	1275	40
100	19	987	1275	23
100	21	955	1275	25
100	23	923	1275	28
100	25	891	1275	30
100	27	859	1275	33
100	29	827	1275	35
100	31	795	1275	38
100	33	763	1275	40
100	35	733	1275	43
100	37	711	1275	44
100	39	697	1275	45
100	41	691	1275	46
100	43	693	1275	46
100	45	703	1275	45
100	47	721	1275	43
100	49	747	1275	41
102	3	1022	1326	23
102	5	926	1326	30
102	7	942	1326	29
102	9	942	1326	29
102	11	830	1326	37
102	13	942	1326	29
102	15	878	1326	34
102	17	798	1326	40
102	19	1038	1326	22
102	21	1006	1326	24
102	23	974	1326	27
102	25	942	1326	29
102	27	910	1326	31
102	29	878	1326	34
102	31	846	1326	36
102	33	814	1326	39
102	35	766	1326	42
102	37	756	1326	43
102	39	738	1326	44
102	41	728	1326	45
102	43	726	1326	45
102	45	732	1326	45
102	47	746	1326	44
102	49	768	1326	42
102	51	766	1326	42
104	3	1055	1378	23
104	5	953	1378	31
104	7	970	1378	30
104	9	970	1378	30
104	11	851	1378	38
104	13	970	1378	30
104	15	902	1378	35
104	17	817	1378	41
104	19	1072	1378	22
104	21	1038	1378	25
104	23	1004	1378	27
104	25	970	1378	30
104	27	936	1378	32
104	29	902	1378	35
104	31	868	1378	37
104	33	834	1378	39
104	35	783	1378	43
104	37	772	1378	44
104	39	752	1378	45
104	41	740	1378	46
104	43	736	1378	47
104	45	740	1378	46
104	47	752	1378	45

N	w	t_{our}	t_A	%
104	49	772	1378	44
104	51	800	1378	42
106	3	1125	1431	21
106	5	1023	1431	29
106	7	1023	1431	29
106	9	1023	1431	29
106	11	921	1431	36
106	13	1023	1431	29
106	15	955	1431	33
106	17	887	1431	38
106	19	1125	1431	21
106	21	1091	1431	24
106	23	1057	1431	26
106	25	1023	1431	29
106	27	989	1431	31
106	29	955	1431	33
106	31	921	1431	36
106	33	887	1431	38
106	35	853	1431	40
106	37	821	1431	43
106	39	797	1431	44
106	41	781	1431	45
106	43	773	1431	46
106	45	773	1431	46
106	47	781	1431	45
106	49	797	1431	44
106	51	821	1431	43
106	53	819	1431	43
108	3	1179	1485	21
108	5	1077	1485	27
108	7	1077	1485	27
108	9	1077	1485	27
108	11	975	1485	34
108	13	1077	1485	27
108	15	1009	1485	32
108	17	941	1485	37
108	19	1179	1485	21
108	21	1145	1485	23
108	23	1111	1485	25
108	25	1077	1485	27
108	27	1043	1485	30
108	29	1009	1485	32
108	31	975	1485	34
108	33	941	1485	37
108	35	907	1485	39
108	37	856	1485	42
108	39	845	1485	43
108	41	825	1485	44
108	43	813	1485	45
108	45	809	1485	46
108	47	813	1485	45
108	49	825	1485	44
108	51	845	1485	43
108	53	873	1485	41
110	3	1216	1540	21
110	5	1108	1540	28
110	7	1108	1540	28
110	9	1108	1540	28
110	11	1000	1540	35
110	13	1108	1540	28
110	15	1036	1540	33
110	17	964	1540	37
110	19	1216	1540	21
110	21	1180	1540	23
110	23	1144	1540	26
110	25	1108	1540	28
110	27	1072	1540	30
110	29	1036	1540	33
110	31	1000	1540	35
110	33	964	1540	37
110	35	928	1540	40
110	37	874	1540	43
110	39	862	1540	44
110	41	840	1540	45
110	43	826	1540	46
110	45	820	1540	47
110	47	822	1540	47
110	49	832	1540	46
110	51	850	1540	45
110	53	876	1540	43
110	55	874	1540	43
112	3	1238	1596	22
112	5	1164	1596	27
112	7	1058	1596	34
112	9	1022	1596	36
112	11	1056	1596	34
112	13	1164	1596	27
112	15	1092	1596	32
112	17	1020	1596	36
112	19	950	1596	40
112	21	1236	1596	23
112	23	1200	1596	25

N	w	t_{our}	t_A	%
112	25	1164	1596	27
112	27	1128	1596	29
112	29	1092	1596	32
112	31	1056	1596	34
112	33	1020	1596	36
112	35	984	1596	38
112	37	948	1596	41
112	39	914	1596	43
112	41	888	1596	44
112	43	870	1596	45
112	45	860	1596	46
112	47	858	1596	46
112	49	864	1596	46
112	51	878	1596	45
112	53	900	1596	44
112	55	930	1596	42
114	3	1275	1653	23
114	5	1221	1653	26
114	7	1095	1653	34
114	9	1059	1653	36
114	11	1113	1653	33
114	13	1221	1653	26
114	15	1149	1653	30
114	17	1077	1653	35
114	19	987	1653	40
114	21	1293	1653	22
114	23	1257	1653	24
114	25	1221	1653	26
114	27	1185	1653	28
114	29	1149	1653	30
114	31	1113	1653	33
114	33	1077	1653	35
114	35	1041	1653	37
114	37	1005	1653	39
114	39	951	1653	42
114	41	939	1653	43
114	43	917	1653	45
114	45	903	1653	45
114	47	897	1653	46
114	49	899	1653	46
114	51	909	1653	45
114	53	927	1653	44
114	55	953	1653	42
114	57	951	1653	42
116	3	1312	1711	23
116	5	1255	1711	27
116	7	1122	1711	34
116	9	1084	1711	37
116	11	1141	1711	33
116	13	1255	1711	27
116	15	1179	1711	31
116	17	1103	1711	36
116	19	1008	1711	41
116	21	1331	1711	22
116	23	1293	1711	24
116	25	1255	1711	27
116	27	1217	1711	29
116	29	1179	1711	31
116	31	1141	1711	33
116	33	1103	1711	36
116	35	1065	1711	38
116	37	1027	1711	40
116	39	970	1711	43
116	41	957	1711	44
116	43	933	1711	45
116	45	917	1711	46
116	47	909	1711	47
116	49	909	1711	47
116	51	917	1711	46
116	53	933	1711	45
116	55	957	1711	44
116	57	989	1711	42
118	3	1390	1770	21
118	5	1240	1770	30
118	7	1200	1770	32
118	9	1162	1770	34
118	11	1200	1770	32
118	13	1088	1770	39
118	15	1238	1770	30
118	17	1162	1770	34
118	19	1086	1770	39
118	21	1390	1770	21
118	23	1352	1770	24
118	25	1314	1770	26
118	27	1276	1770	28
118	29	1238	1770	30
118	31	1200	1770	32
118	33	1162	1770	34
118	35	1124	1770	36
118	37	1086	1770	39
118	39	1048	1770	41
118	41	1012	1770	43

N	w	t_{our}	t_A	%
118	43	984	1770	44
118	45	964	1770	46
118	47	952	1770	46
118	49	948	1770	46
118	51	952	1770	46
118	53	964	1770	46
118	55	984	1770	44
118	57	1012	1770	43
118	59	1010	1770	43
120	3	1450	1830	21
120	5	1279	1830	30
120	7	1260	1830	31
120	9	1222	1830	33
120	11	1260	1830	31
120	13	1127	1830	38
120	15	1298	1830	29
120	17	1222	1830	33
120	19	1146	1830	37
120	21	1450	1830	21
120	23	1412	1830	23
120	25	1374	1830	25
120	27	1336	1830	27
120	29	1298	1830	29
120	31	1260	1830	31
120	33	1222	1830	33
120	35	1184	1830	35
120	37	1146	1830	37
120	39	1108	1830	39
120	41	1051	1830	43
120	43	1038	1830	43
120	45	1014	1830	45
120	47	998	1830	45
120	49	990	1830	46
120	51	990	1830	46
120	53	998	1830	45
120	55	1014	1830	45
120	57	1038	1830	43
120	59	1070	1830	42
122	3	1491	1891	21
122	5	1311	1891	31
122	7	1291	1891	32
122	9	1251	1891	34
122	11	1291	1891	32
122	13	1151	1891	39
122	15	1331	1891	30
122	17	1251	1891	34
122	19	1171	1891	38
122	21	1491	1891	21
122	23	1451	1891	23
122	25	1411	1891	25
122	27	1371	1891	27
122	29	1331	1891	30
122	31	1291	1891	32
122	33	1251	1891	34
122	35	1211	1891	36
122	37	1171	1891	38
122	39	1131	1891	40
122	41	1071	1891	43
122	43	1057	1891	44
122	45	1031	1891	45
122	47	1013	1891	46
122	49	1003	1891	47
122	51	1001	1891	47
122	53	1007	1891	47
122	55	1021	1891	46
122	57	1043	1891	45
122	59	1073	1891	43
122	61	1071	1891	43
124	3	1515	1953	22
124	5	1393	1953	29
124	7	1353	1953	31
124	9	1313	1953	33
124	11	1353	1953	31
124	13	1233	1953	37
124	15	1393	1953	29
124	17	1313	1953	33
124	19	1233	1953	37
124	21	1155	1953	41
124	23	1513	1953	23
124	25	1473	1953	25
124	27	1433	1953	27
124	29	1393	1953	29
124	31	1353	1953	31
124	33	1313	1953	33
124	35	1273	1953	35
124	37	1233	1953	37
124	39	1193	1953	39
124	41	1153	1953	41
124	43	1115	1953	43
124	45	1085	1953	44
124	47	1063	1953	46
124	49	1049	1953	46

N	w	t_{our}	t_A	%
124	51	1043	1953	47
124	53	1045	1953	46
124	55	1055	1953	46
124	57	1073	1953	45
124	59	1099	1953	44
124	61	1133	1953	42
126	3	1556	2016	23
126	5	1456	2016	28
126	7	1416	2016	30
126	9	1376	2016	32
126	11	1416	2016	30
126	13	1296	2016	36
126	15	1456	2016	28
126	17	1376	2016	32
126	19	1296	2016	36
126	21	1196	2016	41
126	23	1576	2016	22
126	25	1536	2016	24
126	27	1496	2016	26
126	29	1456	2016	28
126	31	1416	2016	30
126	33	1376	2016	32
126	35	1336	2016	34
126	37	1296	2016	36
126	39	1256	2016	38
126	41	1216	2016	40
126	43	1156	2016	43
126	45	1142	2016	43
126	47	1116	2016	45
126	49	1098	2016	46
126	51	1088	2016	46
126	53	1086	2016	46
126	55	1092	2016	46
126	57	1106	2016	45
126	59	1128	2016	44
126	61	1158	2016	43
126	63	1156	2016	43
128	3	1640	2080	21
128	5	1520	2080	27
128	7	1480	2080	29
128	9	1440	2080	31
128	11	1480	2080	29
128	13	1360	2080	35
128	15	1520	2080	27
128	17	1440	2080	31
128	19	1360	2080	35
128	21	1280	2080	38
128	23	1640	2080	21
128	25	1600	2080	23
128	27	1560	2080	25
128	29	1520	2080	27
128	31	1480	2080	29
128	33	1440	2080	31
128	35	1400	2080	33
128	37	1360	2080	35
128	39	1320	2080	37
128	41	1280	2080	38
128	43	1240	2080	40
128	45	1202	2080	42
128	47	1172	2080	44
128	49	1150	2080	45
128	51	1136	2080	45
128	53	1130	2080	46
128	55	1132	2080	46
128	57	1142	2080	45
128	59	1160	2080	44
128	61	1186	2080	43
128	63	1220	2080	41
130	3	1683	2145	22
130	5	1557	2145	27
130	7	1515	2145	29
130	9	1473	2145	31
130	11	1515	2145	29
130	13	1389	2145	35
130	15	1557	2145	27
130	17	1473	2145	31
130	19	1389	2145	35
130	21	1305	2145	39
130	23	1683	2145	22
130	25	1641	2145	23
130	27	1599	2145	25
130	29	1557	2145	27
130	31	1515	2145	29
130	33	1473	2145	31
130	35	1431	2145	33
130	37	1389	2145	35
130	39	1347	2145	37
130	41	1305	2145	39
130	43	1263	2145	41
130	45	1223	2145	43
130	47	1191	2145	44
130	49	1167	2145	46

N	w	t_{our}	t_A	%
130	51	1151	2145	46
130	53	1143	2145	47
130	55	1143	2145	47
130	57	1151	2145	46
130	59	1167	2145	46
130	61	1191	2145	44
130	63	1223	2145	43
130	65	1221	2145	43
132	3	1749	2211	21
132	5	1623	2211	27
132	7	1581	2211	28
132	9	1539	2211	30
132	11	1581	2211	28
132	13	1455	2211	34
132	15	1623	2211	27
132	17	1539	2211	30
132	19	1455	2211	34
132	21	1371	2211	38
132	23	1749	2211	21
132	25	1707	2211	23
132	27	1665	2211	25
132	29	1623	2211	27
132	31	1581	2211	28
132	33	1539	2211	30
132	35	1497	2211	32
132	37	1455	2211	34
132	39	1413	2211	36
132	41	1371	2211	38
132	43	1329	2211	40
132	45	1266	2211	43
132	47	1251	2211	43
132	49	1223	2211	45
132	51	1203	2211	46
132	53	1191	2211	46
132	55	1187	2211	46
132	57	1191	2211	46
132	59	1203	2211	46
132	61	1223	2211	45
132	63	1251	2211	43
132	65	1287	2211	42
134	3	1776	2278	22
134	5	1608	2278	29
134	7	1524	2278	33
134	9	1606	2278	29
134	11	1440	2278	37
134	13	1522	2278	33
134	15	1398	2278	39
134	17	1606	2278	29
134	19	1522	2278	33
134	21	1438	2278	37
134	23	1356	2278	40
134	25	1774	2278	22
134	27	1732	2278	24
134	29	1690	2278	26
134	31	1648	2278	28
134	33	1606	2278	29
134	35	1564	2278	31
134	37	1522	2278	33
134	39	1480	2278	35
134	41	1438	2278	37
134	43	1396	2278	39
134	45	1354	2278	41
134	47	1314	2278	42
134	49	1282	2278	44
134	51	1258	2278	45
134	53	1242	2278	45
134	55	1234	2278	46
134	57	1234	2278	46
134	59	1242	2278	45
134	61	1258	2278	45
134	63	1282	2278	44
134	65	1314	2278	42
134	67	1312	2278	42
136	3	1820	2346	22
136	5	1644	2346	30
136	7	1556	2346	34
136	9	1642	2346	30
136	11	1468	2346	37
136	13	1554	2346	34
136	15	1424	2346	39
136	17	1642	2346	30
136	19	1554	2346	34
136	21	1466	2346	38
136	23	1380	2346	41
136	25	1818	2346	23
136	27	1774	2346	24
136	29	1730	2346	26
136	31	1686	2346	28
136	33	1642	2346	30
136	35	1598	2346	32
136	37	1554	2346	34
136	39	1510	2346	36

N	w	t_{our}	t_A	%
136	41	1466	2346	38
136	43	1422	2346	39
136	45	1378	2346	41
136	47	1336	2346	43
136	49	1302	2346	45
136	51	1276	2346	46
136	53	1258	2346	46
136	55	1248	2346	47
136	57	1246	2346	47
136	59	1252	2346	47
136	61	1266	2346	46
136	63	1288	2346	45
136	65	1318	2346	44
136	67	1356	2346	42
138	3	1865	2415	23
138	5	1689	2415	30
138	7	1601	2415	34
138	9	1711	2415	29
138	11	1513	2415	37
138	13	1623	2415	33
138	15	1469	2415	39
138	17	1711	2415	29
138	19	1623	2415	33
138	21	1535	2415	36
138	23	1425	2415	41
138	25	1887	2415	22
138	27	1843	2415	24
138	29	1799	2415	26
138	31	1755	2415	27
138	33	1711	2415	29
138	35	1667	2415	31
138	37	1623	2415	33
138	39	1579	2415	35
138	41	1535	2415	36
138	43	1491	2415	38
138	45	1447	2415	40
138	47	1381	2415	43
138	49	1365	2415	43
138	51	1335	2415	45
138	53	1313	2415	46
138	55	1299	2415	46
138	57	1293	2415	46
138	59	1295	2415	46
138	61	1305	2415	46
138	63	1323	2415	45
138	65	1349	2415	44
138	67	1383	2415	43
138	69	1381	2415	43
140	3	1957	2485	21
140	5	1781	2485	28
140	7	1693	2485	32
140	9	1607	2485	35
140	11	1605	2485	35
140	13	1693	2485	32
140	15	1561	2485	37
140	17	1781	2485	28
140	19	1693	2485	32
140	21	1605	2485	35
140	23	1517	2485	39
140	25	1957	2485	21
140	27	1913	2485	23
140	29	1869	2485	25
140	31	1825	2485	27
140	33	1781	2485	28
140	35	1737	2485	30
140	37	1693	2485	32
140	39	1649	2485	34
140	41	1605	2485	35
140	43	1561	2485	37
140	45	1517	2485	39
140	47	1473	2485	41
140	49	1431	2485	42
140	51	1397	2485	44
140	53	1371	2485	45
140	55	1353	2485	46
140	57	1343	2485	46
140	59	1341	2485	46
140	61	1347	2485	46
140	63	1361	2485	45
140	65	1383	2485	44
140	67	1413	2485	43
140	69	1451	2485	42
142	3	2004	2556	22
142	5	1820	2556	29
142	7	1728	2556	32
142	9	1638	2556	36
142	11	1636	2556	36
142	13	1728	2556	32
142	15	1590	2556	38
142	17	1820	2556	29
142	19	1728	2556	32
142	21	1636	2556	36

N	w	t_{our}	t_A	%
142	23	1544	2556	40
142	25	2004	2556	22
142	27	1958	2556	23
142	29	1912	2556	25
142	31	1866	2556	27
142	33	1820	2556	29
142	35	1774	2556	31
142	37	1728	2556	32
142	39	1682	2556	34
142	41	1636	2556	36
142	43	1590	2556	38
142	45	1544	2556	40
142	47	1498	2556	41
142	49	1454	2556	43
142	51	1418	2556	45
142	53	1390	2556	46
142	55	1370	2556	46
142	57	1358	2556	47
142	59	1354	2556	47
142	61	1358	2556	47
142	63	1370	2556	46
142	65	1390	2556	46
142	67	1418	2556	45
142	69	1454	2556	43
142	71	1452	2556	43
144	3	2076	2628	21
144	5	1892	2628	28
144	7	1800	2628	32
144	9	1685	2628	36
144	11	1708	2628	35
144	13	1800	2628	32
144	15	1662	2628	37
144	17	1892	2628	28
144	19	1800	2628	32
144	21	1708	2628	35
144	23	1616	2628	39
144	25	2076	2628	21
144	27	2030	2628	23
144	29	1984	2628	25
144	31	1938	2628	26
144	33	1892	2628	28
144	35	1846	2628	30
144	37	1800	2628	32
144	39	1754	2628	33
144	41	1708	2628	35
144	43	1662	2628	37
144	45	1616	2628	39
144	47	1570	2628	40
144	49	1501	2628	43
144	51	1484	2628	44
144	53	1452	2628	45
144	55	1428	2628	46
144	57	1412	2628	46
144	59	1404	2628	47
144	61	1404	2628	47
144	63	1412	2628	46
144	65	1428	2628	46
144	67	1452	2628	45
144	69	1484	2628	44
144	71	1524	2628	42
146	3	2105	2701	22
146	5	1965	2701	27
146	7	1873	2701	31
146	9	1781	2701	34
146	11	1781	2701	34
146	13	1873	2701	31
146	15	1735	2701	36
146	17	1965	2701	27
146	19	1873	2701	31
146	21	1781	2701	34
146	23	1689	2701	37
146	25	1599	2701	41
146	27	2103	2701	22
146	29	2057	2701	24
146	31	2011	2701	26
146	33	1965	2701	27
146	35	1919	2701	29
146	37	1873	2701	31
146	39	1827	2701	32
146	41	1781	2701	34
146	43	1735	2701	36
146	45	1689	2701	37
146	47	1643	2701	39
146	49	1597	2701	41
146	51	1553	2701	43
146	53	1517	2701	44
146	55	1489	2701	45
146	57	1469	2701	46
146	59	1457	2701	46
146	61	1453	2701	46
146	63	1457	2701	46
146	65	1469	2701	46

N	w	t_{our}	t_A	%
146	67	1489	2701	45
146	69	1517	2701	44
146	71	1553	2701	43
146	73	1551	2701	43
148	3	2153	2775	22
148	5	2007	2775	28
148	7	1911	2775	31
148	9	1815	2775	35
148	11	1815	2775	35
148	13	1911	2775	31
148	15	1767	2775	36
148	17	2007	2775	28
148	19	1911	2775	31
148	21	1815	2775	35
148	23	1719	2775	38
148	25	1625	2775	41
148	27	2151	2775	22
148	29	2103	2775	24
148	31	2055	2775	26
148	33	2007	2775	28
148	35	1959	2775	29
148	37	1911	2775	31
148	39	1863	2775	33
148	41	1815	2775	35
148	43	1767	2775	36
148	45	1719	2775	38
148	47	1671	2775	40
148	49	1623	2775	42
148	51	1577	2775	43
148	53	1539	2775	45
148	55	1509	2775	46
148	57	1487	2775	46
148	59	1473	2775	47
148	61	1467	2775	47
148	63	1469	2775	47
148	65	1479	2775	47
148	67	1497	2775	46
148	69	1523	2775	45
148	71	1557	2775	44
148	73	1599	2775	42
150	3	2202	2850	23
150	5	2082	2850	27
150	7	1986	2850	30
150	9	1890	2850	34
150	11	1890	2850	34
150	13	1986	2850	30
150	15	1842	2850	35
150	17	2082	2850	27
150	19	1986	2850	30
150	21	1890	2850	34
150	23	1794	2850	37
150	25	1674	2850	41
150	27	2226	2850	22
150	29	2178	2850	24
150	31	2130	2850	25
150	33	2082	2850	27
150	35	2034	2850	29
150	37	1986	2850	30
150	39	1938	2850	32
150	41	1890	2850	34
150	43	1842	2850	35
150	45	1794	2850	37
150	47	1746	2850	39
150	49	1698	2850	40
150	51	1626	2850	43
150	53	1608	2850	44
150	55	1574	2850	45
150	57	1548	2850	46
150	59	1530	2850	46
150	61	1520	2850	47
150	63	1518	2850	47
150	65	1524	2850	47
150	67	1538	2850	46
150	69	1560	2850	45
150	71	1590	2850	44
150	73	1628	2850	43
150	75	1626	2850	43
152	3	2302	2926	21
152	5	2064	2926	29
152	7	2062	2926	30
152	9	1966	2926	33
152	11	1966	2926	33
152	13	2062	2926	30
152	15	1918	2926	34
152	17	1776	2926	39
152	19	2062	2926	30
152	21	1966	2926	33
152	23	1870	2926	36
152	25	1774	2926	39
152	27	2302	2926	21
152	29	2254	2926	23
152	31	2206	2926	25

N	w	t_{our}	t_A	%
152	33	2158	2926	26
152	35	2110	2926	28
152	37	2062	2926	30
152	39	2014	2926	31
152	41	1966	2926	33
152	43	1918	2926	34
152	45	1870	2926	36
152	47	1822	2926	38
152	49	1774	2926	39
152	51	1726	2926	41
152	53	1680	2926	43
152	55	1642	2926	44
152	57	1612	2926	45
152	59	1590	2926	46
152	61	1576	2926	46
152	63	1570	2926	46
152	65	1572	2926	46
152	67	1582	2926	46
152	69	1600	2926	45
152	71	1626	2926	44
152	73	1660	2926	43
152	75	1702	2926	42
154	3	2353	3003	22
154	5	2105	3003	30
154	7	2103	3003	30
154	9	2003	3003	33
154	11	2003	3003	33
154	13	2103	3003	30
154	15	1953	3003	35
154	17	1805	3003	40
154	19	2103	3003	30
154	21	2003	3003	33
154	23	1903	3003	37
154	25	1803	3003	40
154	27	2353	3003	22
154	29	2303	3003	23
154	31	2253	3003	25
154	33	2203	3003	27
154	35	2153	3003	28
154	37	2103	3003	30
154	39	2053	3003	32
154	41	2003	3003	33
154	43	1953	3003	35
154	45	1903	3003	37
154	47	1853	3003	38
154	49	1803	3003	40
154	51	1753	3003	42
154	53	1705	3003	43
154	55	1665	3003	45
154	57	1633	3003	46
154	59	1609	3003	46
154	61	1593	3003	47
154	63	1585	3003	47
154	65	1585	3003	47
154	67	1593	3003	47
154	69	1609	3003	46
154	71	1633	3003	46
154	73	1665	3003	45
154	75	1705	3003	43
154	77	1703	3003	43
156	3	2431	3081	21
156	5	2156	3081	30
156	7	2181	3081	29
156	9	2081	3081	32
156	11	2081	3081	32
156	13	2181	3081	29
156	15	2031	3081	34
156	17	1856	3081	40
156	19	2181	3081	29
156	21	2081	3081	32
156	23	1981	3081	36
156	25	1881	3081	39
156	27	2431	3081	21
156	29	2381	3081	23
156	31	2331	3081	24
156	33	2281	3081	26
156	35	2231	3081	28
156	37	2181	3081	29
156	39	2131	3081	31
156	41	2081	3081	32
156	43	2031	3081	34
156	45	1981	3081	36
156	47	1931	3081	37
156	49	1881	3081	39
156	51	1831	3081	41
156	53	1756	3081	43
156	55	1737	3081	44
156	57	1701	3081	45
156	59	1673	3081	46
156	61	1653	3081	46
156	63	1641	3081	47
156	65	1637	3081	47

N	w	t_{our}	t_A	%
156	67	1641	3081	47
156	69	1653	3081	46
156	71	1673	3081	46
156	73	1701	3081	45
156	75	1737	3081	44
156	77	1781	3081	42
158	3	2462	3160	22
158	5	2260	3160	28
158	7	2112	3160	33
158	9	2160	3160	32
158	11	2160	3160	32
158	13	1962	3160	38
158	15	2110	3160	33
158	17	1960	3160	38
158	19	2260	3160	28
158	21	2160	3160	32
158	23	2060	3160	35
158	25	1960	3160	38
158	27	1862	3160	41
158	29	2460	3160	22
158	31	2410	3160	24
158	33	2360	3160	25
158	35	2310	3160	27
158	37	2260	3160	28
158	39	2210	3160	30
158	41	2160	3160	32
158	43	2110	3160	33
158	45	2060	3160	35
158	47	2010	3160	36
158	49	1960	3160	38
158	51	1910	3160	40
158	53	1860	3160	41
158	55	1812	3160	43
158	57	1772	3160	44
158	59	1740	3160	45
158	61	1716	3160	46
158	63	1700	3160	46
158	65	1692	3160	46
158	67	1692	3160	46
158	69	1700	3160	46
158	71	1716	3160	46
158	73	1740	3160	45
158	75	1772	3160	44
158	77	1812	3160	43
158	79	1810	3160	43
160	3	2515	3240	22
160	5	2340	3240	28
160	7	2165	3240	33
160	9	2240	3240	31
160	11	2240	3240	31
160	13	2015	3240	38
160	15	2190	3240	32
160	17	2040	3240	37
160	19	2340	3240	28
160	21	2240	3240	31
160	23	2140	3240	34
160	25	2040	3240	37
160	27	1915	3240	41
160	29	2540	3240	22
160	31	2490	3240	23
160	33	2440	3240	25
160	35	2390	3240	26
160	37	2340	3240	28
160	39	2290	3240	29
160	41	2240	3240	31
160	43	2190	3240	32
160	45	2140	3240	34
160	47	2090	3240	35
160	49	2040	3240	37
160	51	1990	3240	39
160	53	1940	3240	40
160	55	1865	3240	42
160	57	1846	3240	43
160	59	1810	3240	44
160	61	1782	3240	45
160	63	1762	3240	46
160	65	1750	3240	46
160	67	1746	3240	46
160	69	1750	3240	46
160	71	1762	3240	46
160	73	1782	3240	45
160	75	1810	3240	44
160	77	1846	3240	43
160	79	1890	3240	42
162	3	2567	3321	23
162	5	2385	3321	28
162	7	2203	3321	34
162	9	2281	3321	31
162	11	2281	3321	31
162	13	2047	3321	38
162	15	2229	3321	33
162	17	2073	3321	38

N	w	t_{our}	t_A	%
162	19	2385	3321	28
162	21	2281	3321	31
162	23	2177	3321	34
162	25	2073	3321	38
162	27	1943	3321	41
162	29	2593	3321	22
162	31	2541	3321	23
162	33	2489	3321	25
162	35	2437	3321	27
162	37	2385	3321	28
162	39	2333	3321	30
162	41	2281	3321	31
162	43	2229	3321	33
162	45	2177	3321	34
162	47	2125	3321	36
162	49	2073	3321	38
162	51	2021	3321	39
162	53	1969	3321	41
162	55	1891	3321	43
162	57	1871	3321	44
162	59	1833	3321	45
162	61	1803	3321	46
162	63	1781	3321	46
162	65	1767	3321	47
162	67	1761	3321	47
162	69	1763	3321	47
162	71	1773	3321	47
162	73	1791	3321	46
162	75	1817	3321	45
162	77	1851	3321	44
162	79	1893	3321	43
162	81	1891	3321	43
164	3	2675	3403	21
164	5	2467	3403	28
164	7	2311	3403	32
164	9	2363	3403	31
164	11	2363	3403	31
164	13	2155	3403	37
164	15	2311	3403	32
164	17	2155	3403	37
164	19	2467	3403	28
164	21	2363	3403	31
164	23	2259	3403	34
164	25	2155	3403	37
164	27	2051	3403	40
164	29	2675	3403	21
164	31	2623	3403	23
164	33	2571	3403	24
164	35	2519	3403	26
164	37	2467	3403	28
164	39	2415	3403	29
164	41	2363	3403	31
164	43	2311	3403	32
164	45	2259	3403	34
164	47	2207	3403	35
164	49	2155	3403	37
164	51	2103	3403	38
164	53	2051	3403	40
164	55	1999	3403	41
164	57	1949	3403	43
164	59	1907	3403	44
164	61	1873	3403	45
164	63	1847	3403	46
164	65	1829	3403	46
164	67	1819	3403	47
164	69	1817	3403	47
164	71	1823	3403	46
164	73	1837	3403	46
164	75	1859	3403	45
164	77	1889	3403	44
164	79	1927	3403	43
164	81	1973	3403	42
166	3	2758	3486	21
166	5	2550	3486	27
166	7	2394	3486	31
166	9	2446	3486	30
166	11	2446	3486	30
166	13	2238	3486	36
166	15	2394	3486	31
166	17	2238	3486	36
166	19	2550	3486	27
166	21	2446	3486	30
166	23	2342	3486	33
166	25	2238	3486	36
166	27	2134	3486	39
166	29	2758	3486	21
166	31	2706	3486	22
166	33	2654	3486	24
166	35	2602	3486	25
166	37	2550	3486	27
166	39	2498	3486	28
166	41	2446	3486	30

N	w	t_{our}	t_A	%
166	43	2394	3486	31
166	45	2342	3486	33
166	47	2290	3486	34
166	49	2238	3486	36
166	51	2186	3486	37
166	53	2134	3486	39
166	55	2082	3486	40
166	57	2004	3486	43
166	59	1984	3486	43
166	61	1946	3486	44
166	63	1916	3486	45
166	65	1894	3486	46
166	67	1880	3486	46
166	69	1874	3486	46
166	71	1876	3486	46
166	73	1886	3486	46
166	75	1904	3486	45
166	77	1930	3486	45
166	79	1964	3486	44
166	81	2006	3486	42
166	83	2004	3486	43
168	3	2814	3570	21
168	5	2598	3570	27
168	7	2436	3570	32
168	9	2490	3570	30
168	11	2490	3570	30
168	13	2274	3570	36
168	15	2436	3570	32
168	17	2274	3570	36
168	19	2598	3570	27
168	21	2490	3570	30
168	23	2382	3570	33
168	25	2274	3570	36
168	27	2166	3570	39
168	29	2814	3570	21
168	31	2760	3570	23
168	33	2706	3570	24
168	35	2652	3570	26
168	37	2598	3570	27
168	39	2544	3570	29
168	41	2490	3570	30
168	43	2436	3570	32
168	45	2382	3570	33
168	47	2328	3570	35
168	49	2274	3570	36
168	51	2220	3570	38
168	53	2166	3570	39
168	55	2112	3570	41
168	57	2031	3570	43
168	59	2010	3570	44
168	61	1970	3570	45
168	63	1938	3570	46
168	65	1914	3570	46
168	67	1898	3570	47
168	69	1890	3570	47
168	71	1890	3570	47
168	73	1898	3570	47
168	75	1914	3570	46
168	77	1938	3570	46
168	79	1970	3570	45
168	81	2010	3570	44
168	83	2058	3570	42
170	3	2847	3655	22
170	5	2577	3655	29
170	7	2521	3655	31
170	9	2361	3655	35
170	11	2307	3655	37
170	13	2359	3655	35
170	15	2521	3655	31
170	17	2359	3655	35
170	19	2199	3655	40
170	21	2575	3655	30
170	23	2467	3655	33
170	25	2359	3655	35
170	27	2251	3655	38
170	29	2145	3655	41
170	31	2845	3655	22
170	33	2791	3655	24
170	35	2737	3655	25
170	37	2683	3655	27
170	39	2629	3655	28
170	41	2575	3655	30
170	43	2521	3655	31
170	45	2467	3655	33
170	47	2413	3655	34
170	49	2359	3655	35
170	51	2305	3655	37
170	53	2251	3655	38
170	55	2197	3655	40
170	57	2143	3655	41
170	59	2091	3655	43
170	61	2047	3655	44

N	w	t_{our}	t_A	%
170	63	2011	3655	45
170	65	1983	3655	46
170	67	1963	3655	46
170	69	1951	3655	47
170	71	1947	3655	47
170	73	1951	3655	47
170	75	1963	3655	46
170	77	1983	3655	46
170	79	2011	3655	45
170	81	2047	3655	44
170	83	2091	3655	43
170	85	2089	3655	43
172	3	2904	3741	22
172	5	2634	3741	30
172	7	2607	3741	30
172	9	2418	3741	35
172	11	2364	3741	37
172	13	2445	3741	35
172	15	2607	3741	30
172	17	2445	3741	35
172	19	2256	3741	40
172	21	2661	3741	29
172	23	2553	3741	32
172	25	2445	3741	35
172	27	2337	3741	38
172	29	2202	3741	41
172	31	2931	3741	22
172	33	2877	3741	23
172	35	2823	3741	25
172	37	2769	3741	26
172	39	2715	3741	27
172	41	2661	3741	29
172	43	2607	3741	30
172	45	2553	3741	32
172	47	2499	3741	33
172	49	2445	3741	35
172	51	2391	3741	36
172	53	2337	3741	38
172	55	2283	3741	39
172	57	2229	3741	40
172	59	2148	3741	43
172	61	2127	3741	43
172	63	2087	3741	44
172	65	2055	3741	45
172	67	2031	3741	46
172	69	2015	3741	46
172	71	2007	3741	46
172	73	2007	3741	46
172	75	2015	3741	46
172	77	2031	3741	46
172	79	2055	3741	45
172	81	2087	3741	44
172	83	2127	3741	43
172	85	2175	3741	42
174	3	2960	3828	23
174	5	2680	3828	30
174	7	2652	3828	31
174	9	2456	3828	36
174	11	2400	3828	37
174	13	2484	3828	35
174	15	2652	3828	31
174	17	2484	3828	35
174	19	2288	3828	40
174	21	2708	3828	29
174	23	2596	3828	32
174	25	2484	3828	35
174	27	2372	3828	38
174	29	2232	3828	42
174	31	2988	3828	22
174	33	2932	3828	23
174	35	2876	3828	25
174	37	2820	3828	26
174	39	2764	3828	28
174	41	2708	3828	29
174	43	2652	3828	31
174	45	2596	3828	32
174	47	2540	3828	34
174	49	2484	3828	35
174	51	2428	3828	37
174	53	2372	3828	38
174	55	2316	3828	39
174	57	2260	3828	41
174	59	2176	3828	43
174	61	2154	3828	44
174	63	2112	3828	45
174	65	2078	3828	46
174	67	2052	3828	46
174	69	2034	3828	47
174	71	2024	3828	47
174	73	2022	3828	47
174	75	2028	3828	47
174	77	2042	3828	47

N	w	t_{our}	t_A	%
174	79	2064	3828	46
174	81	2094	3828	45
174	83	2132	3828	44
174	85	2178	3828	43
174	87	2176	3828	43
176	3	3076	3916	21
176	5	2796	3916	29
176	7	2740	3916	30
176	9	2572	3916	34
176	11	2516	3916	36
176	13	2572	3916	34
176	15	2740	3916	30
176	17	2572	3916	34
176	19	2404	3916	39
176	21	2796	3916	29
176	23	2684	3916	31
176	25	2572	3916	34
176	27	2460	3916	37
176	29	2348	3916	40
176	31	3076	3916	21
176	33	3020	3916	23
176	35	2964	3916	24
176	37	2908	3916	26
176	39	2852	3916	27
176	41	2796	3916	29
176	43	2740	3916	30
176	45	2684	3916	31
176	47	2628	3916	33
176	49	2572	3916	34
176	51	2516	3916	36
176	53	2460	3916	37
176	55	2404	3916	39
176	57	2348	3916	40
176	59	2292	3916	41
176	61	2238	3916	43
176	63	2192	3916	44
176	65	2154	3916	45
176	67	2124	3916	46
176	69	2102	3916	46
176	71	2088	3916	47
176	73	2082	3916	47
176	75	2084	3916	47
176	77	2094	3916	47
176	79	2112	3916	46
176	81	2138	3916	45
176	83	2172	3916	45
176	85	2214	3916	43
176	87	2264	3916	42
178	3	3165	4005	21
178	5	2885	4005	28
178	7	2829	4005	29
178	9	2661	4005	34
178	11	2605	4005	35
178	13	2661	4005	34
178	15	2829	4005	29
178	17	2661	4005	34
178	19	2493	4005	38
178	21	2885	4005	28
178	23	2773	4005	31
178	25	2661	4005	34
178	27	2549	4005	36
178	29	2437	4005	39
178	31	3165	4005	21
178	33	3109	4005	22
178	35	3053	4005	24
178	37	2997	4005	25
178	39	2941	4005	27
178	41	2885	4005	28
178	43	2829	4005	29
178	45	2773	4005	31
178	47	2717	4005	32
178	49	2661	4005	34
178	51	2605	4005	35
178	53	2549	4005	36
178	55	2493	4005	38
178	57	2437	4005	39
178	59	2381	4005	41
178	61	2297	4005	43
178	63	2275	4005	43
178	65	2233	4005	44
178	67	2199	4005	45
178	69	2173	4005	46
178	71	2155	4005	46
178	73	2145	4005	46
178	75	2143	4005	46
178	77	2149	4005	46
178	79	2163	4005	46
178	81	2185	4005	45
178	83	2215	4005	45
178	85	2253	4005	44
178	87	2299	4005	43
178	89	2297	4005	43

N	w	t_{our}	t_A	%
180	3	3225	4095	21
180	5	2935	4095	28
180	7	2877	4095	30
180	9	2703	4095	34
180	11	2645	4095	35
180	13	2703	4095	34
180	15	2877	4095	30
180	17	2703	4095	34
180	19	2529	4095	38
180	21	2935	4095	28
180	23	2819	4095	31
180	25	2703	4095	34
180	27	2587	4095	37
180	29	2471	4095	40
180	31	3225	4095	21
180	33	3167	4095	23
180	35	3109	4095	24
180	37	3051	4095	25
180	39	2993	4095	27
180	41	2935	4095	28
180	43	2877	4095	30
180	45	2819	4095	31
180	47	2761	4095	33
180	49	2703	4095	34
180	51	2645	4095	35
180	53	2587	4095	37
180	55	2529	4095	38
180	57	2471	4095	40
180	59	2413	4095	41
180	61	2326	4095	43
180	63	2303	4095	44
180	65	2259	4095	45
180	67	2223	4095	46
180	69	2195	4095	46
180	71	2175	4095	47
180	73	2163	4095	47
180	75	2159	4095	47
180	77	2163	4095	47
180	79	2175	4095	47
180	81	2195	4095	46
180	83	2223	4095	46
180	85	2259	4095	45
180	87	2303	4095	44
180	89	2355	4095	42
182	3	3260	4186	22
182	5	3026	4186	28
182	7	2796	4186	33
182	9	2794	4186	33
182	11	2736	4186	35
182	13	2794	4186	33
182	15	2564	4186	39
182	17	2794	4186	33
182	19	2620	4186	37
182	21	3026	4186	28
182	23	2910	4186	30
182	25	2794	4186	33
182	27	2678	4186	36
182	29	2562	4186	39
182	31	2448	4186	42
182	33	3258	4186	22
182	35	3200	4186	24
182	37	3142	4186	25
182	39	3084	4186	26
182	41	3026	4186	28
182	43	2968	4186	29
182	45	2910	4186	30
182	47	2852	4186	32
182	49	2794	4186	33
182	51	2736	4186	35
182	53	2678	4186	36
182	55	2620	4186	37
182	57	2562	4186	39
182	59	2504	4186	40
182	61	2446	4186	42
182	63	2390	4186	43
182	65	2342	4186	44
182	67	2302	4186	45
182	69	2270	4186	46
182	71	2246	4186	46
182	73	2230	4186	47
182	75	2222	4186	47
182	77	2222	4186	47
182	79	2230	4186	47
182	81	2246	4186	46
182	83	2270	4186	46
182	85	2302	4186	45
182	87	2342	4186	44
182	89	2390	4186	43
182	91	2388	4186	43
184	3	3321	4278	22
184	5	3118	4278	27
184	7	2857	4278	33

N	w	t_{our}	t_A	%
184	9	2886	4278	33
184	11	2828	4278	34
184	13	2886	4278	33
184	15	2625	4278	39
184	17	2886	4278	33
184	19	2712	4278	37
184	21	3118	4278	27
184	23	3002	4278	30
184	25	2886	4278	33
184	27	2770	4278	35
184	29	2654	4278	38
184	31	2509	4278	41
184	33	3350	4278	22
184	35	3292	4278	23
184	37	3234	4278	24
184	39	3176	4278	26
184	41	3118	4278	27
184	43	3060	4278	28
184	45	3002	4278	30
184	47	2944	4278	31
184	49	2886	4278	33
184	51	2828	4278	34
184	53	2770	4278	35
184	55	2712	4278	37
184	57	2654	4278	38
184	59	2596	4278	39
184	61	2538	4278	41
184	63	2451	4278	43
184	65	2428	4278	43
184	67	2384	4278	44
184	69	2348	4278	45
184	71	2320	4278	46
184	73	2300	4278	46
184	75	2288	4278	47
184	77	2284	4278	47
184	79	2288	4278	47
184	81	2300	4278	46
184	83	2320	4278	46
184	85	2348	4278	45
184	87	2384	4278	44
184	89	2428	4278	43
184	91	2480	4278	42
186	3	3381	4371	23
186	5	3171	4371	27
186	7	2901	4371	34
186	9	2931	4371	33
186	11	2871	4371	34
186	13	2931	4371	33
186	15	2661	4371	39
186	17	2931	4371	33
186	19	2751	4371	37
186	21	3171	4371	27
186	23	3051	4371	30
186	25	2931	4371	33
186	27	2811	4371	36
186	29	2691	4371	38
186	31	2541	4371	42
186	33	3411	4371	22
186	35	3351	4371	23
186	37	3291	4371	25
186	39	3231	4371	26
186	41	3171	4371	27
186	43	3111	4371	29
186	45	3051	4371	30
186	47	2991	4371	32
186	49	2931	4371	33
186	51	2871	4371	34
186	53	2811	4371	36
186	55	2751	4371	37
186	57	2691	4371	38
186	59	2631	4371	40
186	61	2571	4371	41
186	63	2481	4371	43
186	65	2457	4371	44
186	67	2411	4371	45
186	69	2373	4371	46
186	71	2343	4371	46
186	73	2321	4371	47
186	75	2307	4371	47
186	77	2301	4371	47
186	79	2303	4371	47
186	81	2313	4371	47
186	83	2331	4371	47
186	85	2357	4371	46
186	87	2391	4371	45
186	89	2433	4371	44
186	91	2483	4371	43
186	93	2481	4371	43
188	3	3505	4465	22
188	5	3147	4465	30
188	7	3025	4465	32
188	9	3025	4465	32

N	w	t_{our}	t_A	%
188	11	2965	4465	34
188	13	3025	4465	32
188	15	2785	4465	38
188	17	3025	4465	32
188	19	2845	4465	36
188	21	2667	4465	40
188	23	3145	4465	30
188	25	3025	4465	32
188	27	2905	4465	35
188	29	2785	4465	38
188	31	2665	4465	40
188	33	3505	4465	22
188	35	3445	4465	23
188	37	3385	4465	24
188	39	3325	4465	26
188	41	3265	4465	27
188	43	3205	4465	28
188	45	3145	4465	30
188	47	3085	4465	31
188	49	3025	4465	32
188	51	2965	4465	34
188	53	2905	4465	35
188	55	2845	4465	36
188	57	2785	4465	38
188	59	2725	4465	39
188	61	2665	4465	40
188	63	2605	4465	42
188	65	2547	4465	43
188	67	2497	4465	44
188	69	2455	4465	45
188	71	2421	4465	46
188	73	2395	4465	46
188	75	2377	4465	47
188	77	2367	4465	47
188	79	2365	4465	47
188	81	2371	4465	47
188	83	2385	4465	47
188	85	2407	4465	46
188	87	2437	4465	45
188	89	2475	4465	45
188	91	2521	4465	44
188	93	2575	4465	42
190	3	3600	4560	21
190	5	3210	4560	30
190	7	3120	4560	32
190	9	3120	4560	32
190	11	3060	4560	33
190	13	3120	4560	32
190	15	2880	4560	37
190	17	3120	4560	32
190	19	2940	4560	36
190	21	2730	4560	40
190	23	3240	4560	29
190	25	3120	4560	32
190	27	3000	4560	34
190	29	2880	4560	37
190	31	2760	4560	39
190	33	3600	4560	21
190	35	3540	4560	22
190	37	3480	4560	24
190	39	3420	4560	25
190	41	3360	4560	26
190	43	3300	4560	28
190	45	3240	4560	29
190	47	3180	4560	30
190	49	3120	4560	32
190	51	3060	4560	33
190	53	3000	4560	34
190	55	2940	4560	36
190	57	2880	4560	37
190	59	2820	4560	38
190	61	2760	4560	39
190	63	2700	4560	41
190	65	2610	4560	43
190	67	2586	4560	43
190	69	2540	4560	44
190	71	2502	4560	45
190	73	2472	4560	46
190	75	2450	4560	46
190	77	2436	4560	47
190	79	2430	4560	47
190	81	2432	4560	47
190	83	2442	4560	46
190	85	2460	4560	46
190	87	2486	4560	45
190	89	2520	4560	45
190	91	2562	4560	44
190	93	2612	4560	43
190	95	2610	4560	43
192	3	3638	4656	22
192	5	3336	4656	28
192	7	3216	4656	31

N	w	t_{our}	t_A	%
192	9	3216	4656	31
192	11	3156	4656	32
192	13	3216	4656	31
192	15	2976	4656	36
192	17	3216	4656	31
192	19	3036	4656	35
192	21	2856	4656	39
192	23	3336	4656	28
192	25	3216	4656	31
192	27	3096	4656	34
192	29	2976	4656	36
192	31	2856	4656	39
192	33	2738	4656	41
192	35	3636	4656	22
192	37	3576	4656	23
192	39	3516	4656	24
192	41	3456	4656	26
192	43	3396	4656	27
192	45	3336	4656	28
192	47	3276	4656	30
192	49	3216	4656	31
192	51	3156	4656	32
192	53	3096	4656	34
192	55	3036	4656	35
192	57	2976	4656	36
192	59	2916	4656	37
192	61	2856	4656	39
192	63	2796	4656	40
192	65	2736	4656	41
192	67	2678	4656	42
192	69	2628	4656	44
192	71	2586	4656	44
192	73	2552	4656	45
192	75	2526	4656	46
192	77	2508	4656	46
192	79	2498	4656	46
192	81	2496	4656	46
192	83	2502	4656	46
192	85	2516	4656	46
192	87	2538	4656	45
192	89	2568	4656	45
192	91	2606	4656	44
192	93	2652	4656	43
192	95	2706	4656	42
194	3	3701	4753	22
194	5	3389	4753	29
194	7	3265	4753	31
194	9	3265	4753	31
194	11	3203	4753	33
194	13	3265	4753	31
194	15	3017	4753	37
194	17	3265	4753	31
194	19	3079	4753	35
194	21	2893	4753	39
194	23	3389	4753	29
194	25	3265	4753	31
194	27	3141	4753	34
194	29	3017	4753	37
194	31	2893	4753	39
194	33	2771	4753	42
194	35	3699	4753	22
194	37	3637	4753	23
194	39	3575	4753	25
194	41	3513	4753	26
194	43	3451	4753	27
194	45	3389	4753	29
194	47	3327	4753	30
194	49	3265	4753	31
194	51	3203	4753	33
194	53	3141	4753	34
194	55	3079	4753	35
194	57	3017	4753	37
194	59	2955	4753	38
194	61	2893	4753	39
194	63	2831	4753	40
194	65	2769	4753	42
194	67	2709	4753	43
194	69	2657	4753	44
194	71	2613	4753	45
194	73	2577	4753	46
194	75	2549	4753	46
194	77	2529	4753	47
194	79	2517	4753	47
194	81	2513	4753	47
194	83	2517	4753	47
194	85	2529	4753	47
194	87	2549	4753	46
194	89	2577	4753	46
194	91	2613	4753	45
194	93	2657	4753	44
194	95	2709	4753	43
194	97	2707	4753	43

N	w	t_{our}	t_A	%
196	3	3766	4851	22
196	5	3487	4851	28
196	7	3363	4851	31
196	9	3363	4851	31
196	11	3301	4851	32
196	13	3363	4851	31
196	15	3115	4851	36
196	17	3363	4851	31
196	19	3177	4851	35
196	21	2991	4851	38
196	23	3487	4851	28
196	25	3363	4851	31
196	27	3239	4851	33
196	29	3115	4851	36
196	31	2991	4851	38
196	33	2836	4851	42
196	35	3797	4851	22
196	37	3735	4851	23
196	39	3673	4851	24
196	41	3611	4851	26
196	43	3549	4851	27
196	45	3487	4851	28
196	47	3425	4851	29
196	49	3363	4851	31
196	51	3301	4851	32
196	53	3239	4851	33
196	55	3177	4851	35
196	57	3115	4851	36
196	59	3053	4851	37
196	61	2991	4851	38
196	63	2929	4851	40
196	65	2867	4851	41
196	67	2774	4851	43
196	69	2749	4851	43
196	71	2701	4851	44
196	73	2661	4851	45
196	75	2629	4851	46
196	77	2605	4851	46
196	79	2589	4851	47
196	81	2581	4851	47
196	83	2581	4851	47
196	85	2589	4851	47
196	87	2605	4851	46
196	89	2629	4851	46
196	91	2661	4851	45
196	93	2701	4851	44
196	95	2749	4851	43
196	97	2805	4851	42
198	3	3896	4950	21
198	5	3586	4950	28
198	7	3462	4950	30
198	9	3216	4950	35
198	11	3400	4950	31
198	13	3092	4950	38
198	15	3214	4950	35
198	17	3462	4950	30
198	19	3276	4950	34
198	21	3090	4950	38
198	23	3586	4950	28
198	25	3462	4950	30
198	27	3338	4950	33
198	29	3214	4950	35
198	31	3090	4950	38
198	33	2966	4950	40
198	35	3896	4950	21
198	37	3834	4950	23
198	39	3772	4950	24
198	41	3710	4950	25
198	43	3648	4950	26
198	45	3586	4950	28
198	47	3524	4950	29
198	49	3462	4950	30
198	51	3400	4950	31
198	53	3338	4950	33
198	55	3276	4950	34
198	57	3214	4950	35
198	59	3152	4950	36
198	61	3090	4950	38
198	63	3028	4950	39
198	65	2966	4950	40
198	67	2904	4950	41
198	69	2844	4950	43
198	71	2792	4950	44
198	73	2748	4950	44
198	75	2712	4950	45
198	77	2684	4950	46
198	79	2664	4950	46
198	81	2652	4950	46
198	83	2648	4950	47
198	85	2652	4950	46
198	87	2664	4950	46
198	89	2684	4950	46

N	w	t_{our}	t_A	%
198	91	2712	4950	45
198	93	2748	4950	44
198	95	2792	4950	44
198	97	2844	4950	43
198	99	2842	4950	43
200	3	3962	5050	22
200	5	3642	5050	28
200	7	3514	5050	30
200	9	3260	5050	35
200	11	3450	5050	32
200	13	3132	5050	38
200	15	3258	5050	35
200	17	3514	5050	30
200	19	3322	5050	34
200	21	3130	5050	38
200	23	3642	5050	28
200	25	3514	5050	30
200	27	3386	5050	33
200	29	3258	5050	35
200	31	3130	5050	38
200	33	3002	5050	41
200	35	3962	5050	22
200	37	3898	5050	23
200	39	3834	5050	24
200	41	3770	5050	25
200	43	3706	5050	27
200	45	3642	5050	28
200	47	3578	5050	29
200	49	3514	5050	30
200	51	3450	5050	32
200	53	3386	5050	33
200	55	3322	5050	34
200	57	3258	5050	35
200	59	3194	5050	37
200	61	3130	5050	38
200	63	3066	5050	39
200	65	3002	5050	41
200	67	2938	5050	42
200	69	2876	5050	43
200	71	2822	5050	44
200	73	2776	5050	45
200	75	2738	5050	46
200	77	2708	5050	46
200	79	2686	5050	47
200	81	2672	5050	47
200	83	2666	5050	47
200	85	2668	5050	47
200	87	2678	5050	47
200	89	2696	5050	47
200	91	2722	5050	46
200	93	2756	5050	45
200	95	2798	5050	45
200	97	2848	5050	44
200	99	2906	5050	42
202	3	4063	5151	21
202	5	3743	5151	27
202	7	3615	5151	30
202	9	3327	5151	35
202	11	3551	5151	31
202	13	3199	5151	38
202	15	3359	5151	35
202	17	3615	5151	30
202	19	3423	5151	34
202	21	3231	5151	37
202	23	3743	5151	27
202	25	3615	5151	30
202	27	3487	5151	32
202	29	3359	5151	35
202	31	3231	5151	37
202	33	3103	5151	40
202	35	4063	5151	21
202	37	3999	5151	22
202	39	3935	5151	24
202	41	3871	5151	25
202	43	3807	5151	26
202	45	3743	5151	27
202	47	3679	5151	29
202	49	3615	5151	30
202	51	3551	5151	31
202	53	3487	5151	32
202	55	3423	5151	34
202	57	3359	5151	35
202	59	3295	5151	36
202	61	3231	5151	37
202	63	3167	5151	39
202	65	3103	5151	40
202	67	3039	5151	41
202	69	2943	5151	43
202	71	2917	5151	43
202	73	2867	5151	44
202	75	2825	5151	45
202	77	2791	5151	46

N	w	t_{our}	t_A	%
202	79	2765	5151	46
202	81	2747	5151	47
202	83	2737	5151	47
202	85	2735	5151	47
202	87	2741	5151	47
202	89	2755	5151	47
202	91	2777	5151	46
202	93	2807	5151	46
202	95	2845	5151	45
202	97	2891	5151	44
202	99	2945	5151	43
202	101	2943	5151	43
204	3	4103	5253	22
204	5	3719	5253	29
204	7	3527	5253	33
204	9	3461	5253	34
204	11	3335	5253	37
204	13	3333	5253	37
204	15	3461	5253	34
204	17	3207	5253	39
204	19	3525	5253	33
204	21	3333	5253	37
204	23	3143	5253	40
204	25	3717	5253	29
204	27	3589	5253	32
204	29	3461	5253	34
204	31	3333	5253	37
204	33	3205	5253	39
204	35	3079	5253	41
204	37	4101	5253	22
204	39	4037	5253	23
204	41	3973	5253	24
204	43	3909	5253	26
204	45	3845	5253	27
204	47	3781	5253	28
204	49	3717	5253	29
204	51	3653	5253	30
204	53	3589	5253	32
204	55	3525	5253	33
204	57	3461	5253	34
204	59	3397	5253	35
204	61	3333	5253	37
204	63	3269	5253	38
204	65	3205	5253	39
204	67	3141	5253	40
204	69	3077	5253	41
204	71	3015	5253	43
204	73	2961	5253	44
204	75	2915	5253	45
204	77	2877	5253	45
204	79	2847	5253	46
204	81	2825	5253	46
204	83	2811	5253	46
204	85	2805	5253	47
204	87	2807	5253	47
204	89	2817	5253	46
204	91	2835	5253	46
204	93	2861	5253	46
204	95	2895	5253	45
204	97	2937	5253	44
204	99	2987	5253	43
204	101	3045	5253	42
206	3	4170	5356	22
206	5	3774	5356	30
206	7	3576	5356	33
206	9	3508	5356	35
206	11	3378	5356	37
206	13	3376	5356	37
206	15	3508	5356	35
206	17	3246	5356	39
206	19	3574	5356	33
206	21	3376	5356	37
206	23	3180	5356	41
206	25	3772	5356	30
206	27	3640	5356	32
206	29	3508	5356	35
206	31	3376	5356	37
206	33	3244	5356	39
206	35	3114	5356	42
206	37	4168	5356	22
206	39	4102	5356	23
206	41	4036	5356	25
206	43	3970	5356	26
206	45	3904	5356	27
206	47	3838	5356	28
206	49	3772	5356	30
206	51	3706	5356	31
206	53	3640	5356	32
206	55	3574	5356	33
206	57	3508	5356	35
206	59	3442	5356	36
206	61	3376	5356	37

N	w	t_{our}	t_A	%
206	63	3310	5356	38
206	65	3244	5356	39
206	67	3178	5356	41
206	69	3112	5356	42
206	71	3048	5356	43
206	73	2992	5356	44
206	75	2944	5356	45
206	77	2904	5356	46
206	79	2872	5356	46
206	81	2848	5356	47
206	83	2832	5356	47
206	85	2824	5356	47
206	87	2824	5356	47
206	89	2832	5356	47
206	91	2848	5356	47
206	93	2872	5356	46
206	95	2904	5356	46
206	97	2944	5356	45
206	99	2992	5356	44
206	101	3048	5356	43
206	103	3046	5356	43
208	3	4239	5460	22
208	5	3843	5460	30
208	7	3645	5460	33
208	9	3612	5460	34
208	11	3447	5460	37
208	13	3480	5460	36
208	15	3612	5460	34
208	17	3315	5460	39
208	19	3678	5460	33
208	21	3480	5460	36
208	23	3249	5460	40
208	25	3876	5460	29
208	27	3744	5460	31
208	29	3612	5460	34
208	31	3480	5460	36
208	33	3348	5460	39
208	35	3183	5460	42
208	37	4272	5460	22
208	39	4206	5460	23
208	41	4140	5460	24
208	43	4074	5460	25
208	45	4008	5460	27
208	47	3942	5460	28
208	49	3876	5460	29
208	51	3810	5460	30
208	53	3744	5460	31
208	55	3678	5460	33
208	57	3612	5460	34
208	59	3546	5460	35
208	61	3480	5460	36
208	63	3414	5460	37
208	65	3348	5460	39
208	67	3282	5460	40
208	69	3216	5460	41
208	71	3117	5460	43
208	73	3090	5460	43
208	75	3038	5460	44
208	77	2994	5460	45
208	79	2958	5460	46
208	81	2930	5460	46
208	83	2910	5460	47
208	85	2898	5460	47
208	87	2894	5460	47
208	89	2898	5460	47
208	91	2910	5460	47
208	93	2930	5460	46
208	95	2958	5460	46
208	97	2994	5460	45
208	99	3038	5460	44
208	101	3090	5460	43
208	103	3150	5460	42
210	3	4377	5565	21
210	5	3981	5565	28
210	7	3783	5565	32
210	9	3717	5565	33
210	11	3585	5565	36
210	13	3585	5565	36
210	15	3717	5565	33
210	17	3453	5565	38
210	19	3783	5565	32
210	21	3585	5565	36
210	23	3387	5565	39
210	25	3981	5565	28
210	27	3849	5565	31
210	29	3717	5565	33
210	31	3585	5565	36
210	33	3453	5565	38
210	35	3321	5565	40
210	37	4377	5565	21
210	39	4311	5565	23
210	41	4245	5565	24

N	w	t_{our}	t_A	%
210	43	4179	5565	25
210	45	4113	5565	26
210	47	4047	5565	27
210	49	3981	5565	28
210	51	3915	5565	30
210	53	3849	5565	31
210	55	3783	5565	32
210	57	3717	5565	33
210	59	3651	5565	34
210	61	3585	5565	36
210	63	3519	5565	37
210	65	3453	5565	38
210	67	3387	5565	39
210	69	3321	5565	40
210	71	3255	5565	42
210	73	3191	5565	43
210	75	3135	5565	44
210	77	3087	5565	45
210	79	3047	5565	45
210	81	3015	5565	46
210	83	2991	5565	46
210	85	2975	5565	47
210	87	2967	5565	47
210	89	2967	5565	47
210	91	2975	5565	47
210	93	2991	5565	46
210	95	3015	5565	46
210	97	3047	5565	45
210	99	3087	5565	45
210	101	3135	5565	44
210	103	3191	5565	43
210	105	3189	5565	43
212	3	4447	5671	22
212	5	4039	5671	29
212	7	3835	5671	32
212	9	3767	5671	34
212	11	3631	5671	36
212	13	3631	5671	36
212	15	3767	5671	34
212	17	3495	5671	38
212	19	3835	5671	32
212	21	3631	5671	36
212	23	3427	5671	40
212	25	4039	5671	29
212	27	3903	5671	31
212	29	3767	5671	34
212	31	3631	5671	36
212	33	3495	5671	38
212	35	3359	5671	41
212	37	4447	5671	22
212	39	4379	5671	23
212	41	4311	5671	24
212	43	4243	5671	25
212	45	4175	5671	26
212	47	4107	5671	28
212	49	4039	5671	29
212	51	3971	5671	30
212	53	3903	5671	31
212	55	3835	5671	32
212	57	3767	5671	34
212	59	3699	5671	35
212	61	3631	5671	36
212	63	3563	5671	37
212	65	3495	5671	38
212	67	3427	5671	40
212	69	3359	5671	41
212	71	3291	5671	42
212	73	3225	5671	43
212	75	3167	5671	44
212	77	3117	5671	45
212	79	3075	5671	46
212	81	3041	5671	46
212	83	3015	5671	47
212	85	2997	5671	47
212	87	2987	5671	47
212	89	2985	5671	47
212	91	2991	5671	47
212	93	3005	5671	47
212	95	3027	5671	47
212	97	3057	5671	46
212	99	3095	5671	45
212	101	3141	5671	45
212	103	3195	5671	44
212	105	3257	5671	43
214	3	4554	5778	21
214	5	4146	5778	28
214	7	3942	5778	32
214	9	3874	5778	33
214	11	3738	5778	35
214	13	3738	5778	35
214	15	3874	5778	33
214	17	3602	5778	38

N	w	t_{our}	t_A	%
214	19	3942	5778	32
214	21	3738	5778	35
214	23	3534	5778	39
214	25	4146	5778	28
214	27	4010	5778	31
214	29	3874	5778	33
214	31	3738	5778	35
214	33	3602	5778	38
214	35	3466	5778	40
214	37	4554	5778	21
214	39	4486	5778	22
214	41	4418	5778	24
214	43	4350	5778	25
214	45	4282	5778	26
214	47	4214	5778	27
214	49	4146	5778	28
214	51	4078	5778	29
214	53	4010	5778	31
214	55	3942	5778	32
214	57	3874	5778	33
214	59	3806	5778	34
214	61	3738	5778	35
214	63	3670	5778	36
214	65	3602	5778	38
214	67	3534	5778	39
214	69	3466	5778	40
214	71	3398	5778	41
214	73	3296	5778	43
214	75	3268	5778	43
214	77	3214	5778	44
214	79	3168	5778	45
214	81	3130	5778	46
214	83	3100	5778	46
214	85	3078	5778	47
214	87	3064	5778	47
214	89	3058	5778	47
214	91	3060	5778	47
214	93	3070	5778	47
214	95	3088	5778	47
214	97	3114	5778	46
214	99	3148	5778	46
214	101	3190	5778	45
214	103	3240	5778	44
214	105	3298	5778	43
214	107	3296	5778	43
216	3	4596	5886	22
216	5	4254	5886	28
216	7	4050	5886	31
216	9	3982	5886	32
216	11	3846	5886	35
216	13	3846	5886	35
216	15	3982	5886	32
216	17	3710	5886	37
216	19	4050	5886	31
216	21	3846	5886	35
216	23	3642	5886	38
216	25	4254	5886	28
216	27	4118	5886	30
216	29	3982	5886	32
216	31	3846	5886	35
216	33	3710	5886	37
216	35	3574	5886	39
216	37	3440	5886	42
216	39	4594	5886	22
216	41	4526	5886	23
216	43	4458	5886	24
216	45	4390	5886	25
216	47	4322	5886	27
216	49	4254	5886	28
216	51	4186	5886	29
216	53	4118	5886	30
216	55	4050	5886	31
216	57	3982	5886	32
216	59	3914	5886	34
216	61	3846	5886	35
216	63	3778	5886	36
216	65	3710	5886	37
216	67	3642	5886	38
216	69	3574	5886	39
216	71	3506	5886	40
216	73	3438	5886	42
216	75	3372	5886	43
216	77	3314	5886	44
216	79	3264	5886	45
216	81	3222	5886	45
216	83	3188	5886	46
216	85	3162	5886	46
216	87	3144	5886	47
216	89	3134	5886	47
216	91	3132	5886	47
216	93	3138	5886	47
216	95	3152	5886	46

N	w	t_{our}	t_A	%
216	97	3174	5886	46
216	99	3204	5886	46
216	101	3242	5886	45
216	103	3288	5886	44
216	105	3342	5886	43
216	107	3404	5886	42
218	3	4667	5995	22
218	5	4315	5995	28
218	7	4105	5995	32
218	9	4035	5995	33
218	11	3895	5995	35
218	13	3895	5995	35
218	15	4035	5995	33
218	17	3755	5995	37
218	19	4105	5995	32
218	21	3895	5995	35
218	23	3685	5995	39
218	25	4315	5995	28
218	27	4175	5995	30
218	29	4035	5995	33
218	31	3895	5995	35
218	33	3755	5995	37
218	35	3615	5995	40
218	37	3477	5995	42
218	39	4665	5995	22
218	41	4595	5995	23
218	43	4525	5995	25
218	45	4455	5995	26
218	47	4385	5995	27
218	49	4315	5995	28
218	51	4245	5995	29
218	53	4175	5995	30
218	55	4105	5995	32
218	57	4035	5995	33
218	59	3965	5995	34
218	61	3895	5995	35
218	63	3825	5995	36
218	65	3755	5995	37
218	67	3685	5995	39
218	69	3615	5995	40
218	71	3545	5995	41
218	73	3475	5995	42
218	75	3407	5995	43
218	77	3347	5995	44
218	79	3295	5995	45
218	81	3251	5995	46
218	83	3215	5995	46
218	85	3187	5995	47
218	87	3167	5995	47
218	89	3155	5995	47
218	91	3151	5995	47
218	93	3155	5995	47
218	95	3167	5995	47
218	97	3187	5995	47
218	99	3215	5995	46
218	101	3251	5995	46
218	103	3295	5995	45
218	105	3347	5995	44
218	107	3407	5995	43
218	109	3405	5995	43
220	3	4740	6105	22
220	5	4425	6105	28
220	7	4215	6105	31
220	9	4145	6105	32
220	11	4005	6105	34
220	13	4005	6105	34
220	15	4145	6105	32
220	17	3865	6105	37
220	19	4215	6105	31
220	21	4005	6105	34
220	23	3795	6105	38
220	25	4425	6105	28
220	27	4285	6105	30
220	29	4145	6105	32
220	31	4005	6105	34
220	33	3865	6105	37
220	35	3725	6105	39
220	37	3550	6105	42
220	39	4775	6105	22
220	41	4705	6105	23
220	43	4635	6105	24
220	45	4565	6105	25
220	47	4495	6105	26
220	49	4425	6105	28
220	51	4355	6105	29
220	53	4285	6105	30
220	55	4215	6105	31
220	57	4145	6105	32
220	59	4075	6105	33
220	61	4005	6105	34
220	63	3935	6105	36
220	65	3865	6105	37

N	w	t_{our}	t_A	%
220	67	3795	6105	38
220	69	3725	6105	39
220	71	3655	6105	40
220	73	3585	6105	41
220	75	3480	6105	43
220	77	3451	6105	43
220	79	3395	6105	44
220	81	3347	6105	45
220	83	3307	6105	46
220	85	3275	6105	46
220	87	3251	6105	47
220	89	3235	6105	47
220	91	3227	6105	47
220	93	3227	6105	47
220	95	3235	6105	47
220	97	3251	6105	47
220	99	3275	6105	46
220	101	3307	6105	46
220	103	3347	6105	45
220	105	3395	6105	44
220	107	3451	6105	43
220	109	3515	6105	42
222	3	4886	6216	21
222	5	4398	6216	29
222	7	4326	6216	30
222	9	4256	6216	32
222	11	4116	6216	34
222	13	4116	6216	34
222	15	4256	6216	32
222	17	3976	6216	36
222	19	4326	6216	30
222	21	4116	6216	34
222	23	3906	6216	37
222	25	3698	6216	41
222	27	4396	6216	29
222	29	4256	6216	32
222	31	4116	6216	34
222	33	3976	6216	36
222	35	3836	6216	38
222	37	3696	6216	41
222	39	4886	6216	21
222	41	4816	6216	23
222	43	4746	6216	24
222	45	4676	6216	25
222	47	4606	6216	26
222	49	4536	6216	27
222	51	4466	6216	28
222	53	4396	6216	29
222	55	4326	6216	30
222	57	4256	6216	32
222	59	4186	6216	33
222	61	4116	6216	34
222	63	4046	6216	35
222	65	3976	6216	36
222	67	3906	6216	37
222	69	3836	6216	38
222	71	3766	6216	39
222	73	3696	6216	41
222	75	3626	6216	42
222	77	3558	6216	43
222	79	3498	6216	44
222	81	3446	6216	45
222	83	3402	6216	45
222	85	3366	6216	46
222	87	3338	6216	46
222	89	3318	6216	47
222	91	3306	6216	47
222	93	3302	6216	47
222	95	3306	6216	47
222	97	3318	6216	47
222	99	3338	6216	46
222	101	3366	6216	46
222	103	3402	6216	45
222	105	3446	6216	45
222	107	3498	6216	44
222	109	3558	6216	43
222	111	3556	6216	43
224	3	4998	6328	21
224	5	4473	6328	29
224	7	4438	6328	30
224	9	4368	6328	31
224	11	4228	6328	33
224	13	4228	6328	33
224	15	4368	6328	31
224	17	4088	6328	35
224	19	4438	6328	30
224	21	4228	6328	33
224	23	4018	6328	37
224	25	3773	6328	40
224	27	4508	6328	29
224	29	4368	6328	31
224	31	4228	6328	33

B Appendix: the source code for computing the percentage of improvements

```
% File Name: ComputePerformance.m
% Author: 曾慧棻
% Email address: uea.am96g@g2.nctu.edu.tw
% Description: Compute the performances of our all-to-all personal
exchange algorithm and Algorithm A for CR(n,w)
% Input: None
% Output: The performances of our algorithm and Algorithm A

%to teacher file

clc; close all; clear all; fclose('all');

nEnd = 200; % we run this program for n = 12, 14, ¡K, 2000
data = [];
fid = fopen('output.txt','wt+');
% the file output.txt is used to store output data

for n = 12 : 2 : nEnd
    wVec = 3 : 2 : n/2;
    p= (n*(n+2))/8;
    % p is the number of time units used by Algorithm A
    for i = 1 : length(wVec)
        w = wVec(i);
        h = floor(33*n/96);
        hp = n-1-2*h ;         % hp is h^\prime in our thesis
        if h >= w
            if mod(h,2) == 0
                for j = 11 : 12
                    if j == 11 % case 1a
                        k = floor(( h+1)/( w+1));
                        kp=(h+1)-k*(w+1);
                        if kp < w
                        t= (h*(h+1))/2+((hp+1)*(8*k+4*kp+hp-1)/8);
                        data = [data ; [n w k t p (p-t)/p]];
                        end
                    else         % case 1b
                         kp = 0 ;
                         k = ( h+1-w)/( w+1);
                         t=((h*(h+1))/2+((hp-1)*(8*k+hp+5)/8)+2*k+3);
                         data = [data ; [n w k t p (p-t)/p ]];
                    end
```

```
              end
            else
              for j = 13 : 14
                if j == 13 % case 1c
                  k = floor(h/(w+1));
                  kp= h-k*(w+1);
                  if kp < w
                  t=((h*(h+1))/2+(hp+1)*(8*k+4*kp+3+hp)/8);
                  data = [data ; [n w k t p (p-t)/p ]];
                  end
                else         % case 1d
                  kp = 0 ;
                  k = ( h-w)/( w+1);
                  t= ((h*(h+1)/2)+(hp+1)*(8*k+3+hp)/8);
                  data = [data ; [n w k t p (p-t)/p]];
                 end
              end
            end
          else
            if (n-2*w-2) >= 0 % case 2b
              j = 22;
              k = 0;
              kp = 0;

t=((h*(h+1))/2+(w-h+4)*(w-h+1)/2+(n-2*w+8)*(n-2*w-2)/8);
              data = [data ; [n w k t p (p-t)/p ]];
            else                 % case 2a
                j = 21;
                k = 0;
                kp = 0;
                t=((h*(h+1))/2+(n-w-h)*(n-w-h+1)/2);
                data = [data ; [n w k t p (p-t)/p ]];
             end
          end
      end
    end

[m,n] = size(data);
```

```
B = isnature(data); % judge if the data is legal
for ii = m : -1 : 1 % if k is positive, then store the ii-th row in
data
    if  B(ii,3) == 0
        data(ii,:) = [];
    end
end

final = [];
for i = 1 : length(data) % store performance to the data.
        final = [final ; data(i,:)];
end

final(:,6) = final(:,6)*100;
for i = 1 : size(final,1)
    fprintf(fid,'%d & %d & %d & %d & %d & %2.0f \\\\\n',final(i,:)
end
fclose(fid);



%============================================================
function B = isnature(A)
[m,n] = size(A);
B = A;
for ii = 1 : m*n  %judge nature
    if A(ii) == round(A(ii)) && A(ii) >= 0
        B(ii) = 1;
    else
        B(ii) = 0;
    end
end
```