Information Processing Letters 44 (1992) 107–111
North-Holland 19 November 1992

The worst case analysis of algorithm on multiple stacks manipulation *

Been-Chian Chien

Institute of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, ROC

Wei-Pang Yang

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan, ROC

Communicated by K. Ikeda
Received 6 February 1991
Revised 24 October 1991

Keywords: Analysis of algorithms; data structures; multiple stacks

1. Introduction

A *stack* is a simple and useful data structure. The simplest and most natural way to keep a stack inside a computer is to put items in a sequential memory area. It is quite convenient in dealing with only *one* stack. However, system developers frequently encounter programs which involve multiple stacks, each of which has dynamically varying size. In such a situation, keeping multiple stacks in a common area with sequential allocation with cause some trouble. First, developers would hate to impose a maximum size on each stack, since the size is usually unpredictable. Second, to store multiple variable-size stacks in sequential locations of a common memory area, the obstacle of *overflow* must be solved. An overflow situation will cause an "error"; it means the stack is already full, yet there are still more items that ought to be put in. A solution for overflow is *reallocating memory*, making room for the overflowed stacks by taking some space from stacks that are not yet filled. This operation may cause many items to be moved to their proper locations in order to keep correctness of the *push* operations coming later.

A number of possible solutions for overflow have been suggested. Knuth proposed a simple solution in reallocating memory by move operations [2]. The method will be described in detail in the next section. He also analyzed the average number of movements when overflow occurs and got a formula concerning the number of stacks and pushed items. Here, we focus on the worst sequence of pushed data instead of the individual worst case, getting some interesting properties similar to [1,3].

2. Knuth's method

The method of multiple stacks manipulation proposed by Knuth [2] is briefly presented here. Assume there are n stacks, and the value $BASE[i]$ and $TOP[i]$ represent the bottom location and the top location of stack i. These stacks all share a common memory area consisting of all locations L with $L_0 < L \leqslant L_\infty$, where L_0 and L_∞ are constants specifying the total number of locations available for use. Knuth's method starts out with

Correspondence to: W.-P. Yang, Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan, ROC
* This research was partly supported by the National Science Council of Taiwan, ROC under contract NSC 80-0408-E009-11 (1990).

all stacks empty, $BASE[i] = TOP[i] = L_0$, for all i, and $BASE[n+1] = L_\infty$. The *Push and Pop* algorithms are as follows:

```
Push: TOP[i] ← TOP[i] + 1;
      if TOP[i] = BASE[i + 1]
      then Overflow
      else CONTENTS[TOP[i]] ← Y;

Pop: if TOP[i] = BASE[i]
     then Underflow
     else
       begin
       Y ← CONTENTS[TOP[i]];
       TOP[i] ← TOP[i] − 1;
       end
```

When stack i overflows, the reallocating strategy will find the smallest k for which $i < k \leqslant n$ or the largest k for which $1 \leqslant k < i$, and k satisfying $TOP[k] < BASE[k+1]$. It then moves the items between stack $(i+1)$ and stack k one entry to the right, if $i < k \leqslant n$; and between stack $(k+1)$ and stack i one entry to the left, otherwise.

This method works simply. It needs, however, some move operations when overflow occurs. Knuth found that the average number of move operations required is

$$\tfrac{1}{2}\left(1 - \frac{1}{n}\right)\binom{m}{2},$$

where m is the number of pushed items and n is the number of stacks. The number of movements is essentially proportional to the square of number of pushed items.

3. The worst push sequence and its analysis

Some important symbols and terminologies must first be defined:

Definition 1. Assume there are n disjoint sets: $S_1, S_2, \ldots, S_n$, $(S_j \cap S_k) = \emptyset$, for $1 \leqslant j,k \leqslant n$, $j \neq k$.

(1) I_i denotes an element in set S_i, and $\bigcup_{1 \leqslant i \leqslant n} S_i = U$.

(2) A push sequence P with m numbers denoted $p_1, \ldots, p_m$ where $p_j \in U$; and let $p_j \equiv I_i$, if $p_j \in S_i$.

(3) $\#(I_i)_P$ is the number of I_i, in push sequence P.

Example 2. If there are 4 stacks and 6 push operations, the 4 stacks here can be viewed as 4 disjoint sets and I_1 representing pushing an item into stack 1. The push sequence $p_1 p_2 p_3 p_4 p_5 p_6$ may be

$$I_1 I_1 I_1 I_1 I_1 I_1,\ I_1 I_2 I_1 I_2 I_1 I_2,\ I_1 I_2 I_3 I_4 I_1 I_2, \ldots .$$

For push sequence $I_1 I_2 I_3 I_4 I_1 I_2$ we have: $\#(I_1)_P = 2$, $\#(I_2)_P = 2$, $\#(I_3)_P = 1$, $\#(I_4)_P = 1$. The total number of push sequences is 4^6 in this example.

Definition 3. Let P be a push sequence with m numbers and n disjoint sets as defined in Definition 1. We say $p_i > p_k$ if $p_i \equiv I_j$ and $p_k \equiv I_l$, $j > l$. $\Phi(p_k)$, the number of p_i, with $1 \leqslant i \leqslant k$ and $p_i > p_k$, is called the *potential* of P_h.

The potential defined above is actually equal to the number of movements when p_k are pushed into stack l.

Example 4. Following Example 2, the relations between push sequence and the total number of movements are shown in Table 1.

Questions arising are: how many movements are needed under the worst push sequence and which push sequence is the worst one?

Table 1

Push sequence	$\Sigma_{i=1}^{6}\Phi(p_k)$ = number of movements
$I_1I_1I_1I_1I_1I_1$	0+0+0+0+0+0 = 0
$I_1I_2I_1I_2I_1I_2$	0+0+1+0+2+0 = 3
⋮	⋮
$I_4I_3I_2I_1I_1I_1$	0+1+2+3+3+3 = 12
$I_4I_3I_2I_1I_2I_1$	0+1+2+3+2+4 = 12
$I_4I_3I_2I_2I_1I_1$	0+1+2+2+4+4 = 13
$I_4I_4I_3I_2I_1I_1$	0+0+2+3+4+4 = 13
$I_4I_3I_3I_2I_1I_1$	0+1+1+3+4+4 = 13
⋮	⋮

The above example gives us important information; i.e., for different push sequences, when their $\#(I_i)_P$, $1 \leqslant i \leqslant n$, are all the same, the number of pushed items are fixed, the sequence of pushing will affect the number of movements. For instance, the number of movements is 12 when the push sequence is $I_4 I_3 I_2 I_1 I_2 I_1$ and it is 13 when the push sequence is $I_4 I_3 I_2 I_2 I_1 I_1$. The following lemma explains this fact.

Lemma 5. *Let there be given n disjoint sets and a push sequence P defined as in Definition* 1. *Let* $\#(I_j)_P = x_j$, $1 \leqslant j \leqslant n$; $\sum_{j=1}^{n} x_j = m$. *We get a maximal* $\sum_{k=1}^{m} \Phi(p_k)$ *if and only if* $p_i \equiv I_n$, *for* $0 < i \leqslant x_n$; $p_i \equiv I_{n-1}$, *for* $x_n < i \leqslant (x_n + x_{n-1})$; $\ldots$; $p_i \equiv I_1$, *for* $\sum_{j=2}^{n} x_j < i \leqslant \sum_{j=1}^{n} x_j$.

Proof. $\Phi(p_k)$ is the number of p_i with $1 \leqslant i \leqslant k$ and $p_i > p_k$. We have

$$\max\left(\sum_{k=1}^{m} \Phi(p_k)\right) = \sum_{k=1}^{m} (\max(\Phi(p_k))),$$

and

$$\max(\Phi(p_k)) = \begin{cases} (x_n + \cdots + x_{j+1}) & \\ \quad \text{if } p_k \equiv I_j, \text{ for } 1 \leqslant j \leqslant n-1, & \\ 0 \quad \text{if } j = n. & \end{cases}$$

There are at least $(x_n + \cdots + x_{j+1})$ elements that appear before p_k, i.e. $k > (x_n + \cdots + x_{j+1})$. Furthermore, the number of p_i satisfying $p_i > p_k$ for $1 \leqslant i < k$ is also at most $(x_n + \cdots + x_{j+1})$. These force sequence p_k, $1 \leqslant k \leqslant m$ to become

$$\begin{aligned} p_i &\equiv I_n, && \text{for } 0 < i \leqslant x_n, \\ p_i &\equiv I_{n-1}, && \text{for } x_n < i \leqslant (x_n + x_{n-1}), \\ &\vdots && \vdots \\ p_i &\equiv I_1, && \text{for } \sum_{j=2}^{n} x_j < i \leqslant \sum_{j=1}^{n} x_j = m, \end{aligned}$$

if and only if $\max(\sum_{k=1}^{m} \Phi(p_k))$ is achieved. □

Let x_i, $\Phi(p_k)$ be defined as in Lemma 5 and Definition 1. $\sum_{k=1}^{m} \Phi(p_k)$, summation of total potential, is equal to the total number of movements. From Lemma 5,

$$\begin{aligned} \sum_{k=1}^{m} \Phi(p_k) &= x_n x_{n-1} + (x_n + x_{n-1}) x_{n-2} + \cdots \\ &\quad + (x_n + \cdots + x_{n-j}) x_{n-j-1} + \cdots \\ &\quad + (x_n + \cdots + x_2) x_1 \\ &= \sum_{i=1}^{n-1} \left(\sum_{j=i+1}^{n} x_j \right) \cdot x_i \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_j \cdot x_i. \end{aligned}$$

Now, the following needs to be found:

$$\max\left(\sum_{k=1}^{m} \Phi(p_k)\right) = \max\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_i \cdot x_j\right),$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i = m, \quad x_i \geqslant 0.$$

It can be formulated as an $(n-1) \times (n-1)$ triangle matrix:

$$\begin{bmatrix} x_n x_{n-1} & & & & \\ x_n x_{n-2} & x_{n-1} x_{n-2} & & & \\ x_n x_{n-3} & x_{n-1} x_{n-3} & x_{n-2} x_{n-3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ x_n x_1 & x_{n-1} x_1 & x_{n-2} x_1 & \cdots & x_2 x_1 \end{bmatrix}$$

$\sum_{k=1}^{m} \Phi(p_k)$ is the sum of the terms in the above matrix. In order to sum these terms, another similar $n \times n$ matrix is considered:

$$\begin{bmatrix} x_n x_n & x_{n-1} x_n & x_{n-2} x_n & \cdots & x_1 x_n \\ x_n x_{n-1} & x_{n-1} x_{n-1} & x_{n-2} x_{n-1} & \cdots & x_1 x_{n-1} \\ x_n x_{n-2} & x_{n-1} x_{n-2} & x_{n-2} x_{n-2} & \cdots & x_1 x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n x_1 & x_{n-1} x_1 & x_{n-2} x_1 & \cdots & x_1 x_1 \end{bmatrix}$$

It has the property $x_i x_j = x_j x_i$ and is symmetrical about the main diagonal. Let

$$T_{\text{upper}} = \sum_{1 \leqslant i \leqslant j \leqslant n} x_i x_j = \sum_{i=1}^{n} \sum_{j=i}^{n} x_i x_j,$$

$$T_{\text{lower}} = \sum_{1 \leqslant i \leqslant j \leqslant n} x_j x_i = \sum_{i=1}^{n} \sum_{j=i}^{n} x_j x_i.$$

In fact, $T_{\text{upper}} = T_{\text{lower}}$, therefore

$$\begin{aligned} T_{\text{upper}} + T_{\text{lower}} &= \sum_{1 \leqslant i,j \leqslant n} x_i x_j + \sum_{i=1}^{n} x_i x_i \\ &= \left(\sum_{i=1}^{n} x_i \right)^2 + \sum_{i=1}^{n} x_i^2 \\ &= 2T_{\text{lower}}. \end{aligned}$$

Summation of triangle matrix is

$$\begin{aligned} \sum_{k=1}^{m} \Phi(p_k) &= T_{\text{lower}} - (\text{main diagonal}) \\ &= \frac{1}{2}\left(\left(\sum_{i=1}^{n} x_i \right)^2 + \sum_{i=1}^{n} x_i^2 \right) - \sum_{i=1}^{n} x_i^2 \\ &= \frac{1}{2}\left(\left(\sum_{i=1}^{n} x_i \right)^2 - \sum_{i=1}^{n} x_i^2 \right), \end{aligned}$$

$$\begin{aligned} &\max\left(\sum_{k=1}^{m} \Phi(p_k) \right) \\ &\quad = \max\left(\frac{1}{2}\left(\left(\sum_{i=1}^{n} x_i \right)^2 - \sum_{i=1}^{n} x_i^2 \right) \right) \\ &\quad = \frac{1}{2}\left[\left(\sum_{i=1}^{n} x_i \right)^2 + \min\left(\sum_{i=1}^{n} x_i^2 \right) \right]. \end{aligned}$$

From Schwarz' inequality,

$$\sum_{i=1}^{n} x_i^2 \geqslant \frac{\left(\sum_{i=1}^{n} x_i \right)^2}{n}, \quad \text{and} \quad \sum_{i=1}^{n} x_i = m,$$

$x_i \geqslant 0$, for $1 \leqslant i \leqslant n$.

$$\begin{aligned} &\max\left(\sum_{k=1}^{m} \Phi(p_k) \right) \\ &\quad = \frac{1}{2}\left(m^2 - \frac{m^2}{n} \right) = \frac{1}{2}\left(1 - \frac{1}{n} \right) m^2. \end{aligned}$$

Theorem 6. *There are n stacks and $m_{\max}$ sequential memory locations. The total number of movements caused by Knuth's method is at most $\frac{1}{2}(1 - 1/n)m^2$ after any sequence of m push operations, where $0 \leqslant m \leqslant m_{\max}$.*

Proof. Let the n stacks be the n disjoint sets $S_1, \ldots, S_n$. The *m push* operations are m numbers in push sequence P. After Knuth's manipulating method is executed, the total number of movements is exactly equal to $\sum_{k=1}^{m} \Phi(p_k)$. From the above description, Knuth's method has number of movements at most $\frac{1}{2}(1 - 1/n)m^2$ after any sequence of *m push* operations. □

The result of worst case after a sequence of *m push* operations is smaller than the summation of individual worst cases, and it approximates 2 times the average case. The number of movements for individual worst case might be equal to the total present items each time a new item is pushed. That is, the total number of movements is the summation from 1 to $m - 1$. It is equal to $m(m - 1)/2$. This case is actually only satisfied while $m = n$. For most cases, they will be smaller than this.

Another question is: how many item numbers in each stack are there when the maximum value occurs. This will be shown in the next theorem.

Theorem 7. *The maximum value in Theorem 6 can be attained while there being $(m - n\lfloor m/n \rfloor)$ number of $x_i = \lfloor m/n \rfloor + 1$ and the others $x_i = \lfloor m/n \rfloor$ for $1 \leqslant i \leqslant n$.*

Proof. From the proof of Theorem 6,

$$\max\left(\sum_{k=1}^{m} \Phi(p_k) \right) = \frac{1}{2}\left[\left(\sum_{i=1}^{n} x_i \right)^2 - \min\left(\sum_{i=1}^{n} x_i^2 \right) \right].$$

If $x_1, \ldots, x_n$ are real numbers, the minimum can be achieved when $x_1 = \cdots = x_n = m/n$.

Unfortunately, $x_1, \ldots, x_n$ are integers here. Let $x_i = \lfloor m/n \rfloor + y_i$, y_i integer, and $-\lfloor m/n \rfloor \leqslant y_i \leqslant m$, for $1 \leqslant i \leqslant n$. We have

$$\sum_{i=1}^{n} y_i = (m \bmod n) = m - n\left\lfloor \frac{m}{n} \right\rfloor,$$

and

$$\min\left(\sum_{i=1}^{n} x_i^2\right)$$
$$= \min\left(\sum_{i=1}^{n}\left(\left\lfloor\frac{m}{n}\right\rfloor + y_i\right)^2\right)$$
$$= \min\left(\sum_{i=1}^{n}\left\lfloor\frac{m}{n}\right\rfloor^2 + 2\left\lfloor\frac{m}{n}\right\rfloor\sum_{i=1}^{n} y_i + \sum_{i=1}^{n} y_i^2\right)$$
$$= n\left\lfloor\frac{m}{n}\right\rfloor^2 + 2\left\lfloor\frac{m}{n}\right\rfloor\left(m - n\left\lfloor\frac{m}{n}\right\rfloor\right) + \min\left(\sum_{i=1}^{n} y_i^2\right).$$

Therefore, $\min(\sum_{i=1}^{n} y_i^2)$ can be achieved when the number of $y_i = 1$ will be $(m - n\lfloor m/n\rfloor)$ and else $y_i = 0$. Restated, the number of $x_i = \lfloor m/n\rfloor + 1$ is $(m - n\lfloor m/n\rfloor)$. Furthermore, $x_i = \lfloor m/n\rfloor + 1$ can be at any i, for $1 \leqslant i \leqslant n$. Except for i of $x_i = \lfloor m/n\rfloor + 1$, x_i is equal to $\lfloor m/n\rfloor$ elsewhere. □

By Theorem 7, the total number of movements may be smaller than the number in Theorem 6. Because

$$\left(\min\sum_{i=1}^{n} x_i^2 - \min\sum_{i=1}^{n}\left(\frac{m}{n}\right)^2\right) \geqslant 0,$$

and

$$\max\left(\min\sum_{i=1}^{n} x_i^2 - \min\sum_{i=1}^{n}\left(\frac{m}{n}\right)^2\right)$$
$$= \max\left(\left(\sum_{i=1}^{n-k}\left\lfloor\frac{m}{n}\right\rfloor^2\right) + \sum_{i=1}^{k}\left(\left\lfloor\frac{m}{n}\right\rfloor + 1\right)^2\right) - \sum_{i=1}^{n}\left(\frac{m}{n}\right)^2$$
$$= \frac{n}{4}, \quad \text{while } (m \bmod n) = \frac{n}{2};$$

where $k = m - n\lfloor m/n\rfloor$.

The actual bound of number of movements may then be $\frac{1}{2}(1 - 1/n)m^2 - n/8$, while $(m \bmod n) = n/2$ exactly occurs.

References

[1] J.L. Bentley and C.C. McGeoch, Amortized analyses of self-organizing sequential search heuristics, *Comm. ACM* **28** (1985) 404–411.

[2] D.E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (Addison-Wesley, Reading, MA, 1973).

[3] D.D. Sleator and R.E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. ACM* **28** (1985) 202–208.