BIT 32 (1992), 546–558.

AMORTIZED ANALYSIS OF SOME DISK SCHEDULING ALGORITHMS: SSTF, SCAN, AND *N*-STEP SCAN*

TUNG-SHOU CHEN, WEI-PANG YANG** and R. C. T. LEE

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

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

Department of Computer Science
National Tsing Hua University, Hsinchu, Taiwan, ROC

Abstract.

The amortized analysis is a useful tool for analyzing the time-complexity of performing a sequence of operations. The disk scheduling problem involves a sequence of requests in general. In this paper, the performances of representative disk scheduling algorithms, *SSTF*, *SCAN*, and *N-StepSCAN*, are analyzed in the amortized sense. A lower bound of the amortized complexity for the disk scheduling problem is also derived. According to our analysis, *SCAN* is not only better than *SSTF* and *N-StepSCAN*, but also an optimal algorithm. Various authors have studied the disk scheduling problem based on some probability models and concluded that the most acceptable performance is obtained from *SCAN*. Our result therefore supports their conclusion.

CR categories: F.2.2, D.4.2.
Keywords: amortized analysis, disk scheduling, on-line problem

1. Introduction.

Disk scheduling is important to operating systems and database management systems. Many disk scheduling algorithms have been proposed, such as the *First-Come-First-Service* (*FCFS*) [6], *Shortest-Seek-Time-First* (*SSTF*) [6], *SCAN* [6], *N-StepSCAN* [8], and *V*(*R*) [9]. In the literature, these algorithms are analyzed based upon probability models [4, 6, 8, 9, 11, 22, 23]. In this paper, we shall use the

---

* This research was supported by the National Science Council, Taiwan R. O. C. under contract: NSC80-0408-E009-11.

** To whom all correspondence should be sent.

Received November 1990. Revised December 1991.

amortized analysis techniques [18, 19, 20, 21] to analyze three disk scheduling algorithms: *SSTF*, *SCAN*, and *N-StepSCAN*.

The *amortized analysis* [18, 19, 20, 21] was proposed by Tarjan. It is a very useful tool for analyzing the time-complexity of performing a sequence of operations. Since the disk scheduling problem involves a sequence of requests, amortized analysis is a very suitable tool to analyze a disk scheduling algorithm which performs a sequence of operations on these requests.

According to our analysis, *SCAN* is the best among the three algorithms. We also showed that *SCAN* is optimal in amortized sense. This matches the previous result [4, 6, 8].

In the rest of this section, the disk scheduling problem and the technique of amortized analysis are introduced. The amortized analysis of the three disk scheduling algorithms is included in Section 2. A lower bound of the amortized complexity for the disk scheduling problem is derived in Section 3. Section 4 compares the three algorithms according to our analysis. Concluding remarks are given in Section 5.

1.1. *The disk scheduling problem.*

The disk scheduling problem can be described as follows: Consider a single disk. Data are stored on various cylinders. At any time, there are a set of requests to retrieve data on this disk. This set of requests is called a *waiting queue* and these requests are called *waiting requests*. The problem is a typical *on-line problem* [10, 13, 14]. It selects one of waiting requests as the next request to be served.

For example, assume that initially there is a sequence of requests (waiting requests) to access data stored on cylinder 9, 2, 8, 4 and 6 respectively. To simplify our illustration, we assume that the disk head is initially located on cylinder 0 and no more requests arrive afterwards.

Suppose that we use a very straight forward algorithm, namely the *FCFS* algorithm, to schedule this sequence. The disk head may first move from cylinder 0 to 9, then to 2, 8, 4, and 6. The total service time of this sequence of requests $(9, 2, 8, 4, 6)$ is $|0-9| + |9-2| + |2-8| + |8-4| + |4-6| = 9 + 7 + 6 + 4 + 2 = 28$. Here we assume that the time for the disk head to move from cylinder i to j is $|i-j|$.

On the other hand, suppose that we use another algorithm, namely the *SSTF* algorithm, to schedule this sequence. In this algorithm, the nearest request is served next. The disk head first moves from cylinder 0 to 2, then to 4, 6, 8, and 9 with total time $2 + 2 + 2 + 2 + 1 = 9$. This shows that different algorithms may produce different results.

In this paper we consider a sequence of m requests processed by a single disk. During the entire process, these requests may keep coming in. If the waiting queue is longer than m, we will ignore those requests outside the m requests. In other words, the maximum length of the waiting queue considered here is m. On the other hand, we assume that the minimum length of the waiting queue is W where $W \geq 2$. A disk scheduling algorithm would select a request in the waiting queue to process. Denote the ith service time by t_i. Then the *amortized complexity* of a disk scheduling algorithm is the worst case of $\sum_{i=1}^{m} t_i/m$.

In the rest of this paper we shall assume that a disk has $Q + 1$ cylinder numbered from 0 to Q with the disk head initially located on cylinder 0. Moreover, we shall assume that the time for the disk head to move from cylinder i to j is $|i - j|$.

Since we only consider a sequence of m requests, the length of the waiting queue will be less than W after the $(m - W + 1)$th servicing. To satisfy our assumption (that the minimum length of the waiting queue is W), $(W - 1)$ *dummy requests* placed on the same cylinder of the last served request are added after the considered sequence of requests. Note that since the dummy requests are all located on the same cylinder of the served request, they will not increase the total service time of the considered sequence of requests. Moreover, since the dummy requests are not processed really, they will not increase the length of the considered sequence of requests. These dummy requests are only added to satisfy our assumption and will not effect the amortized complexity.

1.2. *Technique of amortized analysis.*

The "*potential function*" technique [18, 19, 20, 21] is useful in amortized analysis, and is employed in our analysis. Consider a disk scheduling algorithm X. Let Φ_{i-1}^X and Φ_i^X be the potentials before and after the ith handling of the requests, respectively. The amortized time a_i^X of this is defined as

$$(1) \qquad a_i^X = t_i^X + \Phi_i^X - \Phi_{i-1}^X,$$

where t_i^X is the actual service time of the ith request. Summing the amortized time of all m requests, we have

$$(2) \qquad \sum_{i=1}^{m} a_i^X = \sum_{i=1}^{m} (t_i^X + \Phi_i^X - \Phi_{i-1}^X) = \sum_{i=1}^{m} t_i^X + \Phi_m^X - \Phi_0^X.$$

By deriving an upper bound A^X of a_i^X, we obtain an upper bound of $\sum_{i=1}^{m} t_i^X$ as follows:

$$(3) \qquad \sum_{i=1}^{m} t_i^X = \sum_{i=1}^{m} a_i^X + \Phi_0^X - \Phi_m^X \leq m \cdot A^X + \Phi_0^X - \Phi_m^X.$$

To prove that this upper bound cannot be tightened, a case in which $\sum_{i=1}^{m} t_i^X$ is exactly equal to the upper bound must be given. Then $m \cdot A^X + \Phi_0^X - \Phi_m^X$ is shown to be the worst case of $\sum_{i=1}^{m} t_i^X$. Averaging this result by m, the amortized complexity of the disk scheduling algorithm X is then obtained.

2. Amortized analysis for *SSTF*, *SCAN*, and *N-StepSCAN*.

2.1. *Analysis of Shortest-Seek-Time-First.*

In *SSTF*, the nearest request is always served next. In the case where more than one nearest request appears in the waiting queue, we assume, without loss of generality, that the one with the smallest cylinder number is selected. The direction toward the nearest request is called the *service-direction*.

Consider the status after the ith servicing, where $0 \le i \le m$. Let N_i^{ss} be the number of waiting requests located in the service-direction, L_i^{ss} the distance (in number of cylinders) between the disk head and the nearest request, and D_i^{ss} the number of cylinders in the service-direction. The definition of L_i^{ss} and D_i^{ss} can be seen clearly in Figure 1. The potential function of *SSTF* is defined as

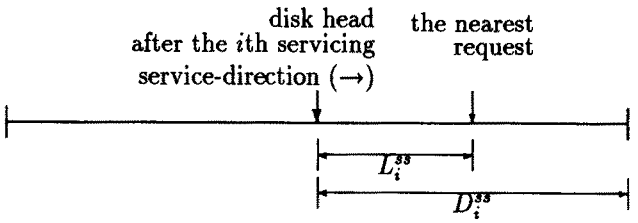

Fig. 1. The definition of L_i^{ss} and D_i^{ss}.

$$\Phi_i^{ss} = \begin{cases} L_i^{ss} & \text{if } N_i^{ss} = 1, \\ \min\left(L_i^{ss}, \dfrac{D_i^{ss}}{2}\right) & \text{if } N_i^{ss} > 1. \end{cases}$$

Three properties associated with Φ_i^{ss} are stated in the following lemmas:

LEMMA 1. $\Phi_i^{ss} \ge 0$ *where* $0 \le i \le m$.

PROOF. Holds trivially since N_i^{ss}, L_i^{ss} and D_i^{ss} are all ≥ 0. ■

LEMMA 2. $\Phi_i^{ss} \le \dfrac{Q}{2}$ *where* $0 \le i \le m$.

PROOF. Consider the case that $N_i^{ss} = 1$. Since $W \ge 2$, there is at least one waiting request located in the opposite direction (with respect to the service-direction). Assuming that the time needed for disk head moving in the opposite direction for the nearest request in that direction is t', it is clear that $L_i^{ss} \le t'$ or the disk head will move in the opposite direction. Thus, we have $L_i^{ss} + t' \le Q$. However, since $L_i^{ss} \le t'$, $2L_i^{ss} \le Q$, i.e., $L_i^{ss} \le Q/2$. Therefore, $\Phi_i^{ss} \le Q/2$ when $N_i^{ss} = 1$.

In the case when $N_i^{ss} \ge 2$, since $D_i^{ss} \le Q$, we have $\Phi_i^{ss} = \min(L_i^{ss}, D_i^{ss}/2) \le D_i^{ss}/2 \le Q/2$. ■

LEMMA 3. *$\Phi_i^{ss} \leqq L_i^{ss}$ where $0 \leq i \leq m$.*

PROOF. Holds trivially by the definition of Φ_i^{ss}. ■

The above three lemmas are used in deriving the amortized complexity of *SSTF* in the following theorem.

THEOREM 1 (Amortized Complexity of *SSTF*). *The amortized complexity of SSTF is $(m + 1)Q/2m$.*

PROOF. Consider the *i*th transaction, where $0 \leq i \leq m$. The actual service time to treat this request is t_i^{ss} and the amortized time is $a_i^{ss} = t_i^{ss} + \Phi_i^{ss} - \Phi_{i-1}^{ss}$. To derive an upper bound A^{ss} of a_i^{ss}, the following three cases are considered:

[Case 1]. $N_{i-1}^{ss} = 1$.
In this case, $\Phi_{i-1}^{ss} = L_{i-1}^{ss}$ by the definition of potential function. Then

$$\begin{aligned} a_i^{ss} &= t_i^{ss} + \Phi_i^{ss} - \Phi_{i-1}^{ss} && \text{according to (1),} \\ &= t_i^{ss} + \Phi_i^{ss} - L_{i-1}^{ss} && \text{since } \Phi_{i-1}^{ss} = L_{i-1}^{ss} \text{ by definition,} \\ &\leq t_i^{ss} + Q/2 - L_{i-1}^{ss} && \text{since } \Phi_i^{ss} \leq Q/2 \text{ by Lemma 2,} \\ &\leq Q/2 && \text{since } L_{i-1}^{ss} = t_i^{ss}. \end{aligned}$$

[Case 2]. $N_{i-1}^{ss} > 1$ and $L_{i-1}^{ss} \leq D_{i-1}^{ss}/2$.
In this case, Φ_{i-1}^{ss} is also equal to L_{i-1}^{ss}. Hence $a_{i-1}^{ss} \leq Q/2$ as in Case 1.

[Case 3]. $N_{i-1}^{ss} > 1$ and $L_{i-1}^{ss} > D_{i-1}^{ss}/2$.
In this case, $\Phi_{i-1}^{ss} = D_{i-1}^{ss}$. Since $N_{i-1}^{ss} > 1$, there are some requests, other than the nearest request, located in the service-direction. Let t'' be the distance (in cylinders) between the nearest request and the second nearest one among all requests located in the service-direction. D_{i-1}^{ss}, L_{i-1}^{ss}, and t'' are shown in Figure 2. After the *i*th

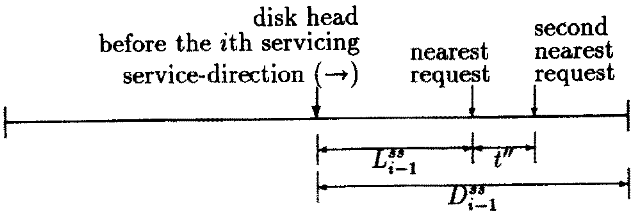

Fig. 2. The disk system before the *i*th transaction.

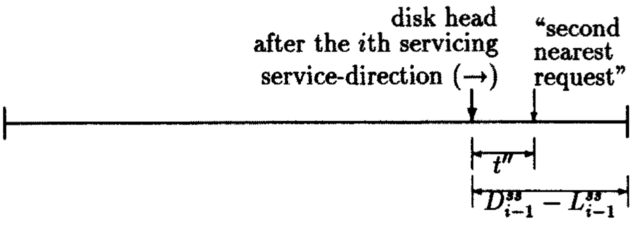

Fig. 3. The disk system after the ith turn.

transaction, the disk system can be illustrated by Figure 3. Because of the "second nearest request", the service time of the $(i+1)$th turn is bounded by t'', i.e., $L^{ss}_i \le t''$. Therefore, we have

(4) $$L^{ss}_i \le t'' \le D^{ss}_{i-1} - L^{ss}_{i-1}.$$

Then,
$$\begin{aligned} a^{ss}_i &= t^{ss}_i + \Phi^{ss}_i - \Phi^{ss}_{i-1} \\ &\le t^{ss}_i + L^{ss}_i - \Phi^{ss}_{i-1} && \text{by Lemma 3,} \\ &\le t^{ss}_i + [D^{ss}_{i-1} - L^{ss}_{i-1}] - \Phi^{ss}_{i-1} && \text{by (4),} \\ &\le D^{ss}_{i-1} - \Phi^{ss}_{i-1} && \text{since } L^{ss}_{i-1} = t^{ss}_i, \\ &= D^{ss}_{i-1}/2 \le Q/2. && \text{since } \Phi^{ss}_{i-1} = D^{ss}_{i-1}/2, \end{aligned}$$

Hence $A^{ss} = Q/2$.

The total service time T^{ss} to serve a sequence of m requests is then obtained by (3) as follows:

$$\begin{aligned} T^{ss} &= \textstyle\sum_{i=1}^m t^{ss}_i = \sum_{i=1}^m a^{ss}_i + \Phi^{ss}_0 - \Phi^{ss}_m \\ &\le m \cdot Q/2 + \Phi^{ss}_0 - \Phi^{ss}_m && \text{since } a^{ss}_i \le A^{ss} = Q/2 \\ &\le (m+1) \cdot Q/2 && \text{since, by Lemma 1 and Lemma 2, the maximum of } \Phi^{ss}_0 \text{ is } Q/2 \text{ and the minimum of } \Phi^{ss}_m \text{ is } 0. \end{aligned}$$

To show that the upper bound $(m+1)\cdot Q/2$ cannot be tightened, consider a sequence of m requests located on cylinders as follows:

$$(Q/2, Q, (0, Q/2)^{(m-3)/2}, 0),$$

where X^y means $\overbrace{(X, X, \ldots, X)}^{y}$. Suppose that the number of waiting requests is 2 at any time and m odd. Then this sequence of requests would be processed as follows:

$$((Q/2, 0)^{(m-1)/2}, Q).$$

Hence the total service time of this sequence is $(m + 1) \cdot Q/2$, which is just equal to the upper bound of T^{ss}. The amortized complexity of *SSTF* is therefore $(m + 1)Q/2m$. ■

2.2. *Analysis for SCAN.*

SCAN is similar to *SSTF* except that it chooses the nearest request in the *sweepdirection.* Assuming that initially the sweep direction is outward, *SCAN* will not change this direction until the disk head reaches the outermost cylinder or until there is no waiting request in this direction, and vice versa.

Consider the status after the *i*th transaction, where $0 \le i \le m$. If the sweep-direction is not changed, N_i^{sc} is defined as the number of requests having been served, and D_i^{sc} is defined as the distance which the disk head has been moved in the current sweep (including this transaction); otherwise, N_i^{sc} and D_i^{sc} are set to zero. N_0^{sc} and D_0^{sc} are zero intuitively. The potential function of *SCAN* is then defined as

$$\Phi_i^{sc} = N_i^{sc} \cdot Q/W - D_i^{sc}.$$

THEOREM 2 (Amortized Complexity of *SCAN*). *The amortized complexity of SCAN is $Q(m - 1)/Wm + Q/m$.*

PROOF.

Consider the *i*th serving, where $0 \le i \le m$. Suppose that the actual service time to serve this request is t_i^{sc}, the amortized time is a_i^{sc}, and an upper bound of a_i^{sc} is A^{sc}. To derive A^{sc}, the following two cases are considered:

[Case 1]. The sweep-direction is not changed after serving this request. In this case, $N_i^{sc} = N_{i-1}^{sc} + 1$ and $D_i^{sc} = D_{i-1}^{sc} + t_i^{sc}$. Then

$$\begin{aligned} a_i^{sc} &= t_i^{sc} + \Phi_i^{sc} - \Phi_{i-1}^{sc} \\ &= t_i^{sc} + [N_i^{sc} Q/W - D_i^{sc}] - [N_{i-1}^{sc} Q/W - D_{i-1}^{sc}] \\ &= t_i^{sc} + [(N_{i-1}^{sc} + 1) \cdot Q/W - (D_{i-1}^{sc} + t_i^{sc})] - [N_{i-1}^{sc} \cdot Q/W - D_{i-1}^{sc}] \\ &= Q/W. \end{aligned}$$

[Case 2]. The sweep-direction is changed after serving this request. In this case, $N_i^{sc} = 0$ and $D_i^{sc} = 0$. $\Phi_i^{sc} = 0$.

$$\begin{aligned} a_i^{sc} &= t_i^{sc} + \Phi_i^{sc} - \Phi_{i-1}^{sc} \\ &= t_i^{sc} + [0] - [N_{i-1}^{sc} Q/W - D_{i-1}^{sc}] \\ &= t_i^{sc} - [N_{i-1}^{sc} Q/W - D_{i-1}^{sc}]. \end{aligned}$$

Since the minimum length of the waiting queue is W by assumption, the minimum number of requests served in one sweep is also W. That is, the sweep-direction cannot be changed when the number of requests served in the current sweep is less than W. Therefore, $N_{i-1}^{sc} \geq (W-1)$ and

$$\begin{aligned} a_i^{sc} &\leq t_i^{sc} - [(W-1)Q/W - D_{i-1}^{sc}] \\ &\leq [t_i^{sc} + D_{i-1}^{sc} - Q] + Q/W. \end{aligned}$$

Moreover, since the maximum distance which the disk head moves in one sweep is Q, $t_i^{sc} + D_{i-1}^{sc} \leq Q$. Therefore, $a_i^{sc} \leq Q/W$.

According to the above discussions, $A^{sc} = Q/W$. Let T^{sc} be the total service time to serve a sequence of m requests. Then

$$\begin{aligned} T^{sc} &= \textstyle\sum_{i=1}^{m} t_i^{sc} \\ &= \textstyle\sum_{i=1}^{m} a_i^{sc} + \Phi_0^{sc} - \Phi_m^{sc} \quad \text{by (3),} \\ &\leq mQ/W + \Phi_0^{sc} - \Phi_m^{sc} \quad \text{since } a_i^{sc} \leq A^{sc} = Q/W. \end{aligned}$$

Since both N_0^{sc} and D_0^{sc} are zero, $\Phi_0^{sc} = 0$ and hence $T^{sc} \leq mQ/W - \Phi_m^{sc}$. Moreover, if $N_m^{sc} = 0$, then $D_m^{sc} = 0$ and $\Phi_m^{sc} = 0$. Otherwise, $D_m^{sc} \in [0, Q]$ and $\Phi_m^{sc} = N_m^{sc} \cdot Q/W - D_m^{sc} \geq Q/W - Q$. Since $Q/W - Q \leq 0$, the minimum of Φ_m^{sc} is $Q/W - Q$. Accordingly,

$$T^{sc} \leq mQ/W - (Q/W - Q) \leq (m-1)Q/W + Q.$$

Consider a sequence of m requests which are respectively located on cylinders

$$((Q^4, 0^4)^{(m-1)/8}, Q)$$

and the number of waiting requests is 4 at any time. Let $W = 4$. Suppose that $(m-1)$ is a multiple of 8. Then, this sequence of requests should also be scheduled and processed by the sequence $((Q^4, 0^4)^{(m-1)/8}, Q)$. The total service time of this sequence is $2Q(m-1)/8 + Q = 2Q(m-1)/2W + Q = (m-1) \cdot Q/W + Q$, which is just equal to the upper bound of T^{sc} obtained. Therefore, the amortized complexity of *SCAN* is $Q(m-1)/Wm + Q/m$. ■

2.3. *Analysis for N-StepsSCAN.*

In *N-stepSCAN*, requests are grouped in size of N or less (if less than N requests remain) according to their arriving order, and are served group by group. A group of requests is served as follows [8]: (a) Select the direction with nearer farthest request and move the disk head to this farthest request processing the requests on its path. (b) Scan back to serve the remaining requests. If no requests remains, (b) is unnecessary. Thus, for each group, at most two sweeps are needed.

Consider the status after the ith turn, where $0 \leq i \leq m$, and assume that this serves a request belonging to the kth group. Let P be the disk head position after the $(k-1)$th group of requests is completed, and P' that after the ith transaction. Suppose that this is not the last of the kth group. Then F_i^{ns} is defined as the smaller of the distance from P to both ends of the disk, i.e., $F_i^{ns} = \min(P, Q-P)$. N_i^{ns} is defined as the number of requests having been served and D_i^{ns} is defined as the distance which the disk head has been moved in processing the kth group. Otherwise, $F_i^{ns} = \min(P', Q-P')$ and $N_i^{ns} = D_i^{ns} = 0$. N_0^{ns} and D_0^{ns} are also zero intuitively. The potential function of *N-StepSCAN* is then defined as

$$\Phi_i^{ns} = N_i^{ns} N/\alpha - D_i^{ns} + F_i^{ns},$$

where $\alpha = \min(N, W)$.

Before analyzing *N-StepsSCAN*, an important lemma concerning the F_i^{ns} is proved below.

LEMMA 4 (Property of the F_i^{ns}). *Consider the processing of a group of requests in N-StepSCAN. Let this group be the kth group of requests and end with the xth transaction. D_k is the total distance of the disk head moving in the k group. Then, $D_k + F_x^{ns} \leq Q + F_{x-1}^{ns}$.*

PROOF. Consider the case requiring only one sweep to serve this group of requests. Let P and P' be the positions of disk head just after the processing of the $(k-1)$th and the kth groups of requests. $D_k = |P - P'|$ and $F_x^{ns} = \min(P', Q-P')$ by definition. Then we only have to consider the following four cases:

[Case 1]. $P \geq P'$ and $P' \geq (Q - P')$.

In this case, $D_k + F_x^{ns} = P - P' + (Q - P') = P + (Q - 2P')$. Since $P' \geq (Q - P')$, $D_k + F_x^{ns} \leq P \leq Q$.

[Case 2]. $P \geq P'$ and $P' < (Q - P')$.

In this case, $D_k + F_x^{ns} = P - P' + P' = P$. Therefore, $D_k + F_x^{ns} \leq Q$.

[Case 3]. $P < P'$ and $P' \geq (Q - P')$.

Then, $D_k + F_x^{ns} = P' - P + (Q - P') = Q - P \leq Q$.

[Case 4]. $P < P'$ and $P' < (Q - P')$.

In this case, $D_k + F_x^{ns} = 2P' - P$. Since $P' < (Q - P')$, $D_k + F_x^{ns} \leq Q - P \leq Q$.

From the discussions above, we may conclude that $D_k + F_x^{ns} \leq Q$ if only one sweep is needed to process the group of requests.

Consider the case requiring two sweeps. Let $D_a + D_b = D_k$, and $D_a \leq F_{x-1}^{ns}$ since the distance between the disk head and the nearer farthest request is F_{x-1}^{ns}. On the other hand, the second sweep can be analyzed as it were from a single sweep, which we did in the above four cases. Therefore, $D_b + F_x^{ns} \leq Q$. Then

$D_k + F_x^{ns} = D_a + D_b + F_x^{ns} \leq F_{x-1}^{ns} + Q$ since $D_a \leq F_{x-1}^{ns}$ and $D_b + F_x^{ns} \leq Q$. ■

The amortized complexity of *N-StepSCAN* is derived in the following theorem.

THEOREM 3 (Amortized Complexity of *N-StepSCAN*). *The amortized complexity of N-StepSCAN is $(Q/m)[(m-1)/\alpha + 1]$.*

PROOF. Consider the ith serving, where $0 \le i \le m$. Suppose that the actual service time to service this request is t_i^{ns}, the amortized time is a_i^{ns}, and an upper bound of a_i^{ns} is A^{ns}. Consider the following two cases:

[Case 1]. This request is not the last served in the current group. In this case, $F_i^{ns} = F_{i-1}^{ns}$, $N_i^{ns} = N_{i-1}^{ns}$, and $D_i^{ns} = D_{i-1}^{ns} + t_i^{ns}$.

$$\begin{aligned}
\text{Therefore, } a_i^{ns} &= t_i^{ns} + \Phi_i^{ns} - \Phi_{i-1}^{ns} \\
&= t_i^{ns} + [N_i^{ns}\cdot Q/\alpha - D_i^{ns} + F_i^{ns}] - [N_{i-1}^{ns}\cdot Q/\alpha - D_{i-1}^{ns} + F_{i-1}^{ns}] \\
&= t_i^{ns} + [(N_{i-1}^{ns} + 1)\cdot Q/\alpha - (D_{i-1}^{ns} + t_i^{ns}) + F_{i-1}^{ns}] \\
&\quad - [N_{i-1}^{ns}\cdot Q/\alpha - D_{i-1}^{ns} + F_{i-1}^{ns}] \\
&= Q/\alpha.
\end{aligned}$$

[Case 2]. This request is the last served in the current group. In this case, $N_i^{ns} = 0$ and $D_i^{ns} = 0$.

$$\begin{aligned}
a_i^{ns} &= t_i^{ns} + \Phi_i^{ns} - \Phi_{i-1}^{ns} \\
&= t_i^{ns} + [N_i^{ns}\cdot Q/\alpha - D_i^{ns} + F_i^{ns}] - [N_{i-1}^{ns}\cdot Q/\alpha - D_{i-1}^{ns} + F_{i-1}^{ns}] \\
&= t_i^{ns} + [F_i^{ns}] - [N_{i-1}^{ns}\cdot Q/\alpha - D_{i-1}^{ns} + F_{i-1}^{ns}].
\end{aligned}$$

Consider the size of a group: If $N < W$, it is N; otherwise it is at least W. It is clear that the size of a group is at least $\alpha (= \min(N, W))$. Hence $N_{i-1}^{ns} \ge (\alpha - 1)$, and

$$\begin{aligned}
a_i^{ns} &\le t_i^{ns} + F_i^{ns} - [(\alpha-1)\cdot Q/\alpha - D_{i-1}^{ns} + F_{i-1}^{ns}] \\
&\le [(t_i^{ns} + D_{i-1}^{ns}) + F_i^{ns} + F_i^{ns} - Q - F_{i-1}^{ns}] + Q/\alpha.
\end{aligned}$$

Moreover, since $t_i^{ns} + D_{i-1}^{ns}$ is the total distance of the disk head moved in the process of the group, $(t_i^{ns} + D_{i-1}^{ns}) + F_i^{ns} \le Q + F_{i-1}^{ns}$ by Lemma 4. Therefore, $(t_i^{ns} + D_{i-1}^{ns}) + F_{i-1}^{ns} - Q - F_{i-1}^{ns} \le 0$ and $a_i^{ns} \le Q/\alpha$.

According to the discussions above, $A^{ns} = Q/\alpha$.

Let T^{ns} be the total service time of a sequence of m requests. Then

$$\begin{aligned}
T^{ns} &= \textstyle\sum_{i=1}^{m} t_i^{ns} \\
&\le mQ/\alpha + \Phi_0^{ns} - \Phi_m^{ns} \quad \text{since} \quad A^{ns} = Q/\alpha \\
&\le m\cdot Q/\alpha - \Phi_m^{ns} \quad\quad \text{since} \quad \Phi_0^{ns} = 0 \text{ (i.e., } N_0^{ns} = D_0^{ns} = F_0^{ns} = 0).
\end{aligned}$$

Moreover, if $N_m^{ns} = 0$, then $D_m^{ns} = 0$ and $\Phi_m^{ns} = F_m^{ns} \ge 0$. Otherwise, $D_m^{ns} \in [0, Q]$ and $\Phi_m^{ns} = N_m^{ns}\cdot Q/\alpha - D_m^{ns} + F_m^{ns} \ge Q/\alpha - Q + F_m^{ns} \ge Q/\alpha - Q$. Since $Q/\alpha - Q \le 0$, the minimum of Φ_m^{ns} is $Q/\alpha - Q$. Therefore,

$$T^{ns} \le m\cdot Q/\alpha - (Q/\alpha - Q) \le (m-1)\cdot Q/\alpha + Q.$$

To show that the upper bound $(m-1)\cdot Q/\alpha + Q$ cannot be tightened, consider the following case. Suppose $N = 5$ and the number of waiting requests is 4 at any time. Let $W = 4$. Then $\alpha = \min(N, W) = \min(5, 4) = 4$. Suppose a sequence of m requests which are respectively located on cylinders $((Q^4, 0^4)^{(m-1)/8}, Q)$ and $(m-1)$ is divisible by 8. By the scheduling of *N-StepSCAN*, this sequence of requests should also be processed as follows:

$$((Q^4, 0^4)^{(m-1)/8}, Q).$$

The total service time of this sequence is therefore $2Q(m-1)/8 + Q = (m-1)\cdot Q/\alpha + Q$, which is just equal to the upper bound of T^{ns} obtained. The same result is obtained if we choose $N \leq W$ in this case. Therefore, the amortized complexity of *N-StepSCAN* is $(Q/m)[(m-1)/\alpha + 1]$. ■

3. A lower bound of amortized complexity for the disk scheduling problem.

To explore how the best scheduling algorithm will behave, a lower bound of the amortized complexity of disk scheduling problem is derived in the following theorem.

THEOREM 4 (Lower bound of the disk scheduling problem). *The amortized complexity of any disk scheduling algorithm must be* $\geq Q/W$.

PROOF. To prove the amortized complexity is lower-bounded by Q/W, suppose that there is a scheduling algorithm with amortized complexity less than Q/W.

Consider a sequence of requests $((Q^W, 0^W)^k)$ for arbitrarily large k. Suppose the number of waiting requests in W at any time. This sequence should also be scheduled and processed by the sequence $((Q^W, 0^W)^k)$. The length of this sequence is $2Wk$. Since the disk head is initially placed on cylinder 0, the total service time of this sequence is $2Qk$. The average service time of this sequence is then Q/W which contradicts our previous assumption. ■

4. Comparisons.

Here comparisons are made among *SSTF*, *SCAN*, and *N-Step-SCAN* with respect to their amortized complexities. The results obtained in the previous section are listed in the first column of Table 1. Note that another famous disk scheduling algorithm *FCFS* is actually the 1-*StepSCAN*. Therefore the row 1-*StepSCAN* in Table 1 can be viewed as the complexities of *FCFS*.

It is difficult to compare these disk scheduling algorithms for arbitrary m from the first column of Table 1. However, as $m \to \infty$, the second column shows that both

Table 1. *Amortized complexities of disk scheduling algorithms.*

Method	Amortized Complexity	Amortized Complexity $(m \to \infty)$
SSTF	$(m+1) \cdot Q/2m$	$Q/2$
SCAN	$(m-1)/m \cdot Q/W + Q/m$	Q/W
1-*StepSCAN*(*FCFS*)	$(m-1)/m \cdot Q + Q/m$	Q
N-StepSCAN($N < W$)	$(m-1)/m \cdot Q/N + Q/m$	Q/N
N-StepSCAN($N \geq W$)	$(m-1)/m \cdot Q/W + Q/m$	Q/W
Lower Bound	Q/W	Q/W

SCAN and *N-StepSCAN* with $N \geq W$ are optimal in amortized complexity, but *N-StepSCAN* is not optimal when $N < W$. We conclude that *SCAN* is the best disk scheduling algorithm among *SSTF*, *SCAN*, and *N-StepSCAN* in amortized sense. The 1-*StepSCAN* (*FCFS*) is the worst among them.

5. Conclusion.

The performance of three representative disk scheduling algorithms, *SSTF*, *SCAN*, and *N-StepSCAN*, are analyzed in amortized sense. A lower bound of the amortized complexity for the disk scheduling problem is also derived. According to our analysis, *SCAN* is not only better than *SSTF* and *N-StepSCAN*, but is also an optimal algorithm. Various authors have studied the disk scheduling problem based on probability models [4, 6, 8] and concluded that the most acceptable performance is obtained from *SCAN*. Our result therefore supports their conclusion.

As far as the authors know this is the first paper performing amortized analysis on a set of practical algorithms for computer systems. We believe that we have opened a new field, namely, applying amortized analysis to many existing algorithms involving a long sequence of operations.

Acknowledgements.

The authors wish to express their thanks to the referee and to Arbee L. P. Chen, Judy C. R. Tseng, G. J. Hwang and J. Y. Tien whose valuable comments improved presentation and organization of this paper.

REFERENCES

1. J. L. Bentley and C. C. McGeoch, *Amortized analyses of self-organizing sequential heuristics*, Comm. ACM 28, No. 4 (1985), pp. 404–411.
2. T. S. Chen, W. P. Yang and R. C. T. Lee, *Amortized analyses of disk scheduling algorithms: SSTF and SCAN*, Proc. of National Computer Symposium, Taiwan, R.O.C., (1989), pp. 831–836.

3. T. S. Chen and W. P. Yang, *Amortized analysis for a continuum of disk scheduling V(R)*, Proc. of International Conference on Information and Systems, AMSE, Hangzhou, China, (1991), pp. 1277–1280.
4. E. G. Coffman, L. A. Klimko and B. Ryan, *Analysis of scanning policies for reducing disc seek times*, SIAM Journal on Computing, 1, (1972), pp. 269–279.
5. H. M. Deitel, *Operating Systems* (Second Edition), Addison-Wesley Publishing Co. Reading Mass., (1990).
6. P. J. Denning, *Effects of scheduling on file memory operations*, Proc. AFIPS, Montvale, N.J., (1967), pp. 9–21.
7. M. L. Fredman, R. Sedgewick, D. D. Sleator and R. E. Tarjan, *The pairing heap: a new form of self-adjusting heap*, Algorithmica No. 1 (1986), pp. 111–129.
8. H. Frank, *Analysis and optimization of disk storage devices for time sharing systems*, J. ACM 16, No. 4 (1969), pp. 602–620.
9. R. Geist and S. Daniel, *A continuum of disk scheduling algorithms*, ACM Trans. Computer Syst. 5, No. 1 (1987), pp. 77–92.
10. E. F. Grove, *The harmonic online k-server algorithm is competitive*, In Proc. 23rd ACM Symposium on Theory of Computing, (1991), pp. 260–266.
11. M. Hofri, *Disk scheduling FCFS vs. SSTF revisited*, Comm. ACM 23, No. 11 (1980), pp. 645–653.
12. R. C. T. Lee, R. C. Chang and S. S. Tseng, *Introduction to the Design and Analysis of Algorithms*, Prentice Hall, (1989).
13. M. Manasse, L. A. McGeoch and D. D. Sleator, *Competitive algorithms for on-line problems*, In Proc. 20th Annual ACM Symposium on Theory of Computing, (1988), pp. 322–333.
14. M. Manasse, L. A. McGeoch and D. D. Sleator, *Competitive algorithms for server problems*, Journal of Algorithms, 11 (1990), pp. 208–230.
15. B. M. E. Moret and H. D. Shapior, *Algorithms from P to NP, Volume I: Design and Efficiency*, The Benjamin/Cummings Publishing Company, Inc. Redwood City, CA 94065 (1991).
16. P. W. Purdom, Jr. and C. A. Brown, *The Analysis of Algorithms*, CBS College Publishing, (1985).
17. A. Silberschatz and J. L. Peterson, *Operating System Concepts*, Addison-Wesley Publishing, Reading Mass., (1988), pp. 263.
18. D. D. Sleator and R. E. Tarjan, *Self-adjusting binary search trees*, J. Assoc. Comput. Mach. 32, No. 3 (1985), pp. 652–686.
19. D. D. Sleator and R. E. Tarjan, *Self-adjusting heaps*, SIAM J. Comput. 15, No. 1 (1986), pp. 52–69.
20. R. E. Tarjan, *Data structure and network algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., (1983).
21. R. E. Tarjan, *Amortized computational complexity*, SIAM J. Alg. Disc. Math. 6, No. 2 (1985), 306–318.
22. T. J. Teorey and T. B. Pinkerton, *A comparative analysis of disk scheduling policies*, Comm. ACM 15, No. 3 (1972), pp. 177–194.
23. N. C. Wilhelm, *An anomaly in disk scheduling: a comparison of FCFS and SSTF seek scheduling using an empirical model for disk accesses*, Comm. ACM 19, No. 1 (1976), 13–17.
24. W. P. Yang and T. S. Chen, *The study of amortized complexity on disk system*, Technical Report, NSC79-0408-E009-01, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., (1989).