MAC TR-121

ON LOWER BOUNDS FOR SELECTION PROBLEMS

Foong Frances Yao

This research was supported by the National Science Foundation under research grant GJ-34671.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

This empty page was substituted for a blank page in the original document.

TABLE OF CONTENTS

CHAPTER 1 PRELIMINARIES

1.1 Introduction

1.2 Tree Algorithm

1.3 The Concept of a Strategy

1.4 Crucial vs. Noncrucial Comparisons

CHAPTER 2 SELECTION OF THE THIRD BEST PLAYER

2.1 Introduction

2.2 Weight Function and Basic Strategy

2.3 Nearly Exact Bounds for i=3

2.3.1 Main Theorem

2.3.2 Proof of Lemma 1

CHAPTER 3 GENERAL LOWER BOUNDS FOR SMALL i

3.1 Introduction

3.2 Main Theorem

3.3 Proof of Main Theorem

3.4 Proof of Lemma 2

3.4.1 Auxiliary Propositions

3.4.2 The Inductive Proof

CHAPTER 4 FINDING A MEDIOCRE PLAYER

4.1 Introduction

4.2 Properties of $S(i,j,n)$

4.3 $S(1,j,n) = V_2(j+2)$

4.4 Connections with Median Computation

This empty page was substituted for a blank page in the original document.

ABSTRACT

Let $V_i(n)$ be the minimum number of binary comparisons that are required to determine the i-th largest of n elements drawn from a totally ordered set. In this thesis we use adversary strategies to prove lower bounds on $V_i(n)$. For i=3, our lower bounds determine $V_3(n)$ precisely for infinitely many values of n, and determine $V_3(n)$ to within 2 for all n. For a general fixed i, our lower bound has the asymptotic form $n + (i-1)\log n - O(\log(\log^* n))$ where $\log^* n$ is a very slowly growing function. As a result, the asymptotic behavior of $V_i(n)$ is determined to within $O(\log(\log^* n))$. A more general problem is raised in which one wants to find an element which is (i,j)-mediocre, i.e., smaller than at least i elements and greater than at least j elements. For i=1, it is shown that the best algorithm is to select the i+1st largest of any subset of i+j+1 elements. It is an interesting question whether for general i this procedure is also optimal for finding an (i,j)-mediocre element. An affirmative answer to this question would imply $V_{n/2}(n) \leq 3n$.

CHAPTER 1

PRELIMINARIES

1.1 Introduction

In this thesis we are interested in the problem of determining the i-th largest element of a totally ordered set of n objects. We shall concentrate on selection methods in which only pairwise comparisons are allowed.

The history of this problem dates back to 1883 when Lewis Carroll in an essay[2] pointed out that the usual playing procedures of a "knockout" tennis tournament would in most cases select the wrong second and third best players. Therefore in the essay he set out to devise a plan for finding the true second and third best players. There are of course many ways to achieve this goal, and a question that naturally arises is "In how few matches (assuming transitivity and antisymmetry) between n players can one decide the i-th best player?" Let $V_i(n)$ be the answer to this problem. The other closely related problem is to select in order the first, second, ..., and the i-th best players, and the minimum number of matches required here we denote by $W_i(n)$.

Some obvious properties of V and W are:

(i) $V_i(n) \leq W_i(n)$

(ii) $V_i(n) = V_{n-i+1}(n)$, $W_i(n) = W_{n-i+1}(n)$ (by symmetry)

(iii) $W_i(n) \geq \lceil \log(i!) \rceil$, an information theoretic lower bound.

(All of our logarithms are taken to the base 2.)

(iv) $V_1(n) = W_1(n) = n - 1$ (cf. also Sec. 1.4)

In 1932 Schreier[7] gave an algorithm for finding the first and second best of n players that requires at most $n - 2 + \lceil \log n \rceil$ matches. This construction was generalized by Kislitsyn[3] in 1964 to show that

$$W_i(n) \leq n - i + \sum_{k=0}^{i-2} \lceil \log(n-k) \rceil \quad \text{for all } i. \tag{1}$$

His algorithm uses first i stages of "tree selection" sort. (cf. Knuth[4], Sec. 5.2.3) We first set up a knockout tournament of n players, and determine the best player in n-1 matches. We then "output" the champion and select the best of the remaining players. After repeating this procedure i times, we will have found the first, second, ..., and the i-th best players. Note that after the initial tournament is set up, at each later stage only a portion of the tournament needs to be reconstructed in order to find the new champion. One can in fact show that the first i stages require at most the number of matches as shown on the right hand side of (1).

If only the i-th best player is desired, Hadian and Sobel[5] proved that

$$V_i(n) \leq n - i + (i-1)\lceil \log(n-i+2) \rceil \quad \text{for all } i. \tag{2}$$

Their construction is as follows: One sets aside a set S of i-2 players and perform a knockout tournament on the remaining n-i+2 guys. (This requires n-i+1 matches.) The champion of this tournament ranks higher than n-i+1 players, and hence is too good to be the i-th best player. We replace him by a player from S. One rebuilds part of the tournament as needed to determine the new champion (which requires at most $\lceil \log(n-i+2) \rceil$ matches). Again, we replace the champion after finding him. After i-1 passes, we have successfully eliminated i-1 players that are overqualified. Now the champion of the remaining n-i+1 players in the tournament must be the true i-th best player. (The last step takes $\lceil \log(n-i+2) \rceil - 1$ matches.) Summing the matches, we get Equ.(2).

For small i, both of the algorithms described above seem to be rather efficient. However, they require O(nlogn) comparisons when i = O(n). For a while it was not clear whether any efficient algorithms might exist in the case of large i. Finally, in 1971 Blum, Floyd, Pratt, Rivest, and Tarjan[1] proposed a uniform procedure which can find the i-th largest of n elements in 5.73n comparisons for all i.

A very challenging task in the theoretical study of algorithms is to prove lower bounds on the complexity of a given problem. Until recently little was known about lower bounds in the tennis tournament problem. The first step was made in 1964 by Kislitsyn who showed, in

the same paper cited above, that $n - 2 + \lceil \log n \rceil$ is in fact the minimum number of comparisons required by any algorithm to find the largest and the second largest elements. Therefore, Schreier's construction gives an optimal algorithm. In 1971 Blum et al[1] proved that $n + i + \lceil \log n \rceil - 4$ is a lower bound for selecting the i-th largest element where $i \leq n/2$. Recently this result was further improved by Pratt[6] to give

$$V_i(n) \geq n + 2i - \log n \quad \text{for} \quad i \leq n/3,$$

and

$$V_i(n) \geq (3n+i)/2 - \log n \quad \text{for} \quad n/3 \leq i \leq n/2. \tag{3}$$

Impressive as this bound is, it still leaves a considerable gap between upper and lower bounds when i is small. As a matter of fact, the asymptotic behavior of the Hadian-Sobel upper bound (2) is $\sim n + (i-1)\log n$ while that of the previous lower bound is $\sim n + \log n$. It is therefore an intriquing problem to determine what the true asymptotic behavior of the optimal algorithm is.

In this thesis, we present lower bounds for the selection of the i-th largest element and the selection of the largest i elements (in order). For i small relative to n, the gap between upper and lower bounds for either problem now has the leading term $(i-1)(i-3)\log(\log^* n)$ where $\log^* n$ is a very slowly growing function. (cf. its definition in Sec.3.2) In the case of $i = 3$, we have a tighter lower bound which actually determines $V_3(n)$ precisely for infinitely many values of n ; and determines $V_3(n)$ and $W_3(n)$ to within 1 or 2 for all

1.2 Tree Algorithm

Let $x_1, \ldots, x_n$ be n distinct numbers. To select the i-th largest of $x_1, \ldots, x_n$ is to determine x_r, $1 \le r \le n$, such that

$$\{ x_k \mid x_k > x_r \} \ge i - 1 \quad \text{and} \quad \{ x_k \mid x_k < x_r \} \ge n - i .$$

We shall write $F_i(x_1, \ldots, x_n) = r$ to denote that x_r is the i-th largest of $x_1, \ldots, x_n$. To sort $x_1, \ldots, x_n$, then, is to compute $F_i(x_1, \ldots, x_n)$ simultaneously for $1 \le i \le n$ — or equivalently, to determine a permutation $p(1)p(2)\cdots p(n)$ such that $x_{p(1)} > x_{p(2)} > \cdots > x_{p(n)}$. We write $G(x_1, \ldots, x_n) = p$ for the permutation p which satisfies this condition.

It is quite obvious that if $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are two sequences each consisting of n distinct numbers, with the property that $(\forall j)(\forall k)[\, x_j < x_k \Leftrightarrow y_j < y_k \,]$, then we must have $F_i(x_1, \ldots, x_n) = F_i(x_1, \ldots, x_n)$ for all i, and $G(x_1, \ldots, x_n) = G(y_1, \ldots, y_n)$. In other words, for operations such as sorting or selecting the i-th largest element, the answer depends solely on the ordering relation between the input elements. Hence, for this class of problems, it is natural to study computing techniques that are based entirely on pairwise comparisons between the input items.

An algorithm that satisfies the above constraint can be represented by a binary tree structure, such as that shown in Figure 1, and

will be called a tree algorithm. Each internal node contains two indices "i:j", denoting a comparison of the i-th input x_i versus the j-th input x_j. For simplicity, we shall always assume that all inputs are distinct, so that each comparison may have two outcomes. The left subtree of this node then represents the subsequent comparisons to be made if $x_i < x_j$, and the right subtree describes the succeeding moves when $x_i > x_j$. Since we will be interested in minimizing the number

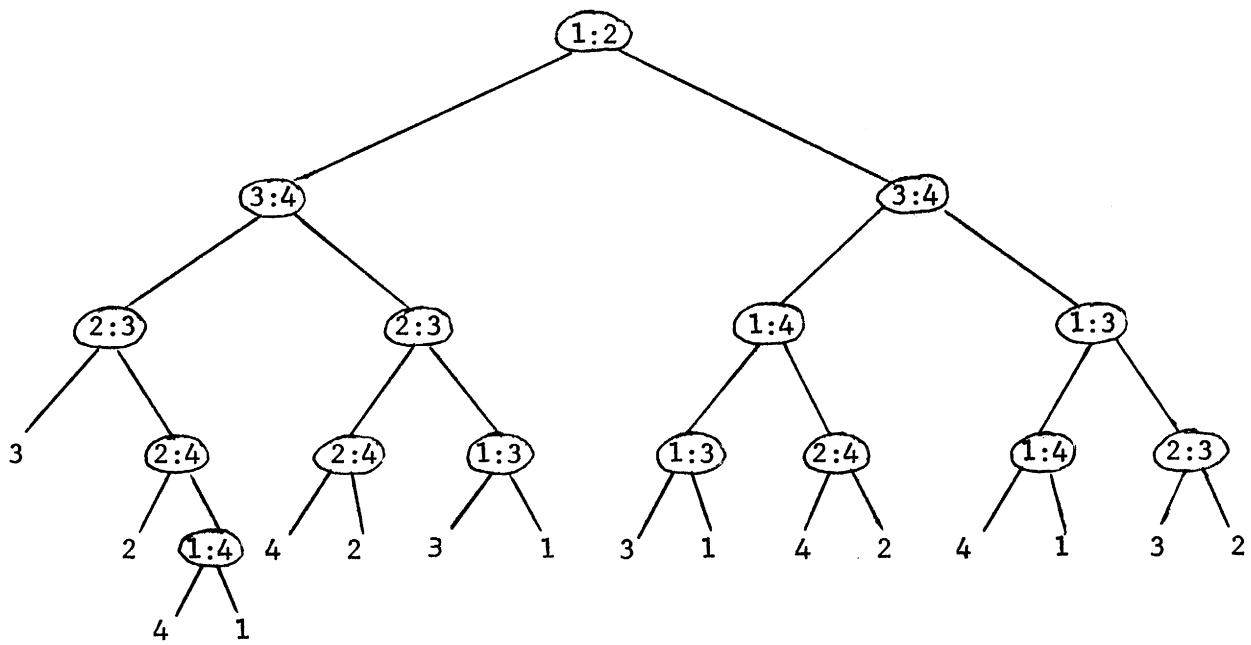


Figure 1 A tree algorithm for selecting the second largest of four elements

of comparisons needed, we may assume that no redundant comparisons occur in a tree algorithm. Thus the left and right subtrees of an internal node must both be accessible, and hence all external nodes are reachable from the root. If this algorithm is to compute the i-th largest of n items, then each external node of the tree shall contain an index r denoting the fact that $F_i(x_1, \ldots, x_n) = r$ has been established as a result of the comparisons made along the path from the root to this node. Similarly, for sorting algorithms each external node shall contain a permutation which equals $G(x_1, \ldots, x_n)$.

As a complexity measure, we shall define the <u>cost</u> of a tree algorithm to be the maximal path length of its comparison tree. Thus, the functions $V_i(n)$ and $W_i(n)$ that are informally defined in the last section can be expressed as

$$V_i(n) = \min \{\text{cost of } A \mid A \text{ is a tree algorithm that computes } F_i(x_1, \ldots, x_n)\}$$

$$W_i(n) = \min \{\text{cost of } A \mid A \text{ is a tree algorithm that computes } F_j(x_1, \ldots, x_n) \text{ simultaneously for } 1 \le j \le i\}$$

1.3 The Concept of a Strategy

The cost of a tree algorithm can be considered from a game-theoretic point of view. This approach, as we shall see, enables us to obtain tight lower bounds on the minimum number of comparisons required to do various selections.

For example, we may look at the computation of the i-th largest of n items as a two-person game in the following way. There is player A who makes the opening move, and whose move shall always be an inquiry of the form "Is x_i less than x_j?". The adversary B, in his turns, shall give a reply of either "yes" or "no" to the question that A just asked. The game ends when A can successfully determine the i-th largest element. The payoff of B in this game is defined to be the total number of questions asked by A.

Now, recall in game theory, a player's "strategy" is simply a complete specification of what he would do in each situation that might arise in the play of the game. In the game described above, any strategy that A may employ corresponds to a tree algorithm for selecting the i-th largest element, and vice versa. Once A has selected a strategy (i.e., a tree algorithm is given), the choice of a strategy on B's part will then completely determine the path that the computation is going to follow in the comparison tree of this algorithm. Therefore, if we can show that for any strategy A may use, B can

always find an "adversary strategy" that quarantees a payoff of at least $L_i(n)$ for B, then we in fact will have proved that

$$V_i(n) \geq L_i(n).$$

Notice that, as far as proving lower bounds for tree algorithms is concerned, adversary B has the privilege of consulting the strategy which A has chosen while he makes up his own. However, all the adversary strategies that are considered in the following chapters do have uniform prescriptions, i.e., decisions are solely based on previous moves and do not depend on the knowledge of A's strategy.

In giving descriptions of strategies and situations that might arise in a computation, it is convenient to use the terminology of tennis tournaments. After all, this is where the selection problem originated! So, let us regard each input item as a tennis player. A comparison between x_i and x_j then becomes a "match" of x_i versus x_j. If the outcome is $x_i < x_j$, we say "x_i is defeated by x_j"; if $x_i > x_j$, we say "x_i beats x_j", etc. (Here we have to assume that tennis skill satisfies transitivity, i.e., if x_i beats x_j and x_j beats x_k, then x_i would beat x_k.) A sequence of matches, each tagged with its outcome, is called a <u>tournament</u>, usually denoted by T. The <u>length</u> of a tournament T, written as $|T|$, is the total number of matches T contains. Thus the payoff of adversary B corresponds to the length of the resulting tournament. In connection with a tournament T, we shall let <u>time t</u> refer to the point when the t-th

[redacted]

1.4 Crucial vs. Noncrucial Comparisons

With respect to a tournament T, let us write $x_j < x_k$ if in the final ranking x_j is determined to be dominated by x_k. We write $x_j \leq x_k$ if either $x_j < x_k$ or $j = k$. It is easy to see that

1) if $x_k < x_r$, then in some match x_k must be defeated by an x_j such that $x_j \leq x_r$.

2) if $x_r < x_k$, then in some match x_k must beat an x_j such that $x_r \leq x_j$.

Now, if T selects the i-th best player, there must be a player x_r such that

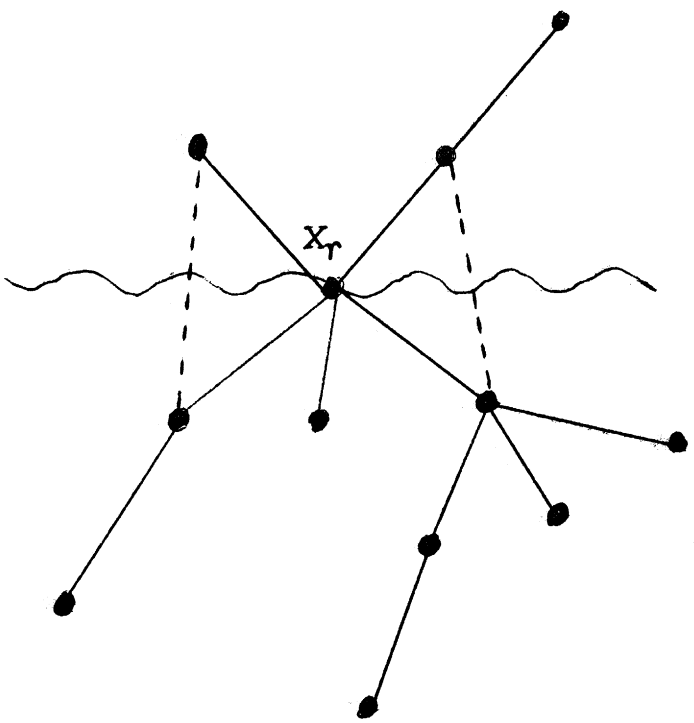
$$|\{x_k \mid x_k < x_r\}| = n - i \quad \text{and} \quad |\{x_k \mid x_r < x_k\}| = i - 1.$$

Hence for each player other than x_r we must be able to isolate a match of either type 1) or 2), which we call a <u>crucial</u> match. Thus T must contain n-1 crucial matches. Alternatively, if we look at the Hasse diagram that represents the partial ordering of the players as established by T, n-1 crucial matches are those required to form the links of a spanning tree on the n nodes, with x_r looking like a "bottleneck". (cf. Figure 2) Thus we know that

$$V_i(n) \geq n - 1 \quad \text{for all } i.$$

Incidentally, this proves $V_1(n) = n - 1$ since it is easy to see that $V_1(n) \leq n - 1$. For general i, in order to improve this estimate on

the lower bound of $V_i(n)$, the adversary must try to entrap A into making a large number of comparisons that are <u>noncrucial</u>.



<u>Figure 2</u> Crucial matches (solid lines) and noncrucial matches (dotted lines) for selecting x_r as the fourth best of 12 players.

CHAPTER 2

SELECTION OF THE THIRD BEST PLAYER

2.1 Introduction

The adversary approach, as introduced in Sec. 1.3, was used by Knuth (Sec. 5.3.3 of [4]) in proving Kislitsyn's theorem that $n - 2 + \lceil \log n \rceil$ comparisons are indeed necessary for finding the second largest of n items. In this chapter, we present Basic Strategy which is a slight modification of the strategy Knuth used. We shall first use Basic Strategy to reconstruct Knuth's proof, and then proceed to establish lower bounds for $V_3(n)$ and $W_3(n)$. Lemma 1 describes a specific situation that must arise in any computation controlled by Basic Strategy. By taking advantage of this situation, one can create a large number of noncrucial comparisons, and lower bounds for $V_3(n)$ and $W_3(n)$ are obtained thereby. As a consequence, $V_3(n)$ is determined precisely for infinitely many values of n, and to within 1 or 2 for general n. $W_3(n)$ is also determined to within 1 or 2 for all n.

2.2 Weight function and Basic Strategy

In Sec. 1.3 we introduced the notion of using strategies to establish lower bounds for selection problems. The Basic Strategy, to be defined below, shall play an essential role in the forming of our final strategies for the adversary. It is a modification of the strategy Knuth used in proving Kislitsyn's lower bound on $V_2(n)$. First we have to define a weight function:

Definition 1 With respect to a tournament T, $Q_t(x)$ where x is any player and $0 \le t \le |T|$, is a positive integer known as x's weight at time t, and is defined recursively as follows:

Q1. $Q_0(x) = 1$ for all x.

Q2. If x inflicted y's first loss in the t-th match of T, then $Q_t(x) = Q_{t-1}(x) + Q_{t-1}(y)$, and $Q_t(z) = Q_{t-1}(z)$ for all z such that $z \neq x$.

Q3. If the loser of the t-th match was previously beaten, then $Q_t(x) = Q_{t-1}(x)$ for all x.

We can now present

Definition 2 Basic Strategy

Assume x plays y in the t-th match of T:

BS1. If x is yet undefeated and y is not, then let x win.

BS2. If both x and y are undefeated and $Q_{t-1}(x) > Q_{t-1}(y)$, then let x win.

BS3. In other circumstances, let the outcome be arbitrary.

Definition 3 A tournament is said to be BS-ruled if the outcomes of the successive matches are decided by Basic Strategy.

The following are some central effects of our Basic Strategy.

Fact 1 In a BS-ruled tournament, once a player is defeated, his weight does not increase from then on.

Proof By the definition of $Q_t(x)$, a player's weight increases only when he makes a first loss. From BS1, a player who has been defeated cannot inflict any first loss, therefore his weight shall stay the same.

Fact 2 In a BS-ruled tournament, a player's weight is at most 2^k after he makes k first defeats.

Proof A player's weight increases only with every first defeat he makes. By BS2 and Q2, his weight at most doubles when he does make a first defeat. Since the initial weight is 1, Fact 2 is proved by induction on k.

In the definitions of weight and Basic Strategy, only a player's first defeat is relevant. Indeed, we can represent the players' weights in a BS-ruled tournament T by a tree structure, called the first-defeat-tree (abbr. FDT) of T. Every player is represented by a node in the FDT, and y is a son of x iff x inflicted y's first loss in T. Thus the FDT may be a disjoint union of several rooted trees, with every root representing an undefeated player. (cf. Figure 3) In view of Fact 1, we shall let Q(x) denote the maximum

value of x's weight in a BS-ruled tournament. Then we have

Fact 3 Given a BS-ruled tournament T, let D(x) be the set of x's descendants (including x itself) in the FDT of T. Then $Q(x) = |D(x)|$.

Proof Argue by induction on the number of first defeats x makes. Use Q2 and Fact 1. (Details omitted).

Thus, by Fact 3, Q(x) corresponds to the total number of nodes (or the actual "weight") of the subtree rooted at x in the FDT. For example, in Figure 3 we have Q(x)=10 and Q(y)=4.

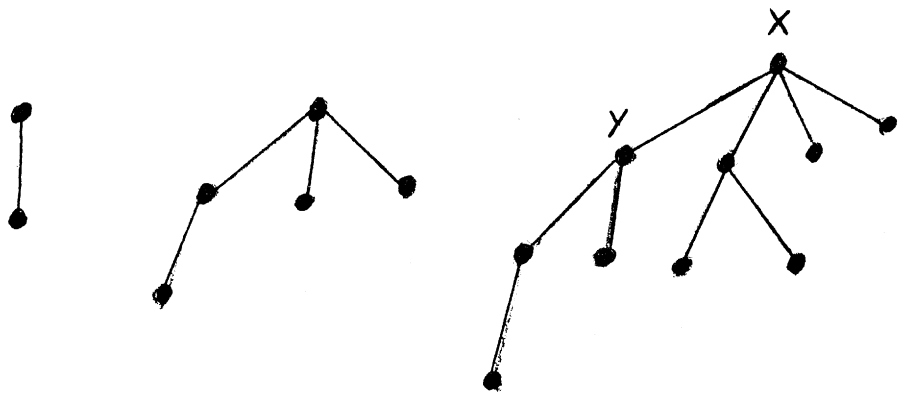


Figure 3 First-defeat-tree of a BS-ruled tournament

It is now a simple matter to prove

Theorem (Kislitsyn) $V_2(n) \geq n - 2 + \lceil \log n \rceil$.

Proof Given any tree algorithm for selecting the second best player, we apply the Basic Strategy to it. Note that the champion is also determined in the resulting tournament T (he is the singular player dominating the selected second best), call him x_c . Since he must be the only undefeated player in T, in the FDT of T he is hence the unique root. By Fact 3, we have $Q(x_c) = n$. Therefore x_c must have made at least $\lceil \log n \rceil$ first defeats in T. However, at most one of these could be a crucial match (cf. Sec. 1.4) as far as selecting the second best player is concerned. Since there must be $n - 2$ other crucial matches, we conclude that $T \geq n - 2 + \lceil \log n \rceil$. This proves that $V_2(n) \geq n - 2 + \lceil \log n \rceil$.

2.3 Nearly Exact Bounds for i=3

In this section, we proceed to prove fairly tight lower bounds for $V_3(n)$ (the minimum number of comparisons for selecting the third largest element), and $W_3(n)$ (that for selecting the first, second, and third largest elements in order).

2.3.1 Main Theorem

The main result of this section is the following theorem:

Theorem 1 $V_3(n) \geq H(n)$, where for $k \geq 1$,

$$H(n) = \begin{cases} n - 3 + 2\lceil \log(n-1) \rceil & \text{if } n = 2^k + 1 \\ n - 4 + 2\lceil \log(n-1) \rceil & \text{if } 3 \cdot 2^{k-1} < n \leq 2^{k+1} \\ n - 5 + 2\lceil \log(n-1) \rceil & \text{if } 2^k + 1 < n \leq 3 \cdot 2^{k-1} \end{cases}$$

By comparing $H(n)$ with Hadian and Sobel's upper bound (cf. Sec. 1.1) $V_3(n) \leq n - 3 + 2\lceil \log(n-1) \rceil$ we see that $V_3(n)$ is determined precisely if $n = 2^k + 1$; and is determined to within 1 or 2 for general n . Since $W_3(n) \geq V_3(n) \geq H(n)$, comparing $H(n)$ with Kislitsyn's upper bound $W_3(n) \leq n - 3 + \lceil \log n \rceil + \lceil \log(n-1) \rceil$ shows that $W_3(n)$ is determined to within 1 or 2 for all n .

The following lemma is the key to the proof of Theorem 1. To simplify notation, let us define

$$h(n) = H(n) - n + 1 .$$

It may be helpful to keep in mind that $h(n)$ is an estimate of the

number of noncrucial (i.e., wasted) comparisons that the adversary is aiming to create.

Lemma 1 Any BS-ruled tournament that selects the third best of n players must reach a point in time t when one of the following two situations arises:

(1) There exist 3 undefeated players A, B, and C; A and B together have made at least $h(n)$ first defeats so far.

(2) There exist 2 undefeated players A' and B' who together have made at least $h(n) + 1$ first defeats so far.

We first show how Theorem 1 can be proved by using Lemma 1. The proof of Lemma 1 will be given in Sec. 2.3.2.

Proof of Theorem 1 We present a strategy which, when applied to any algorithm for selecting the third best player, will result in a tournament that contains at least $H(n)$ matches. The strategy has two phases:

Phase I Basic Strategy

Follow Basic Strategy until either situation (1) or situation (2) of Lemma 1 occurs. At this point t, switch to

Phase II Clear Strategy

CS1. As a follow-up strategy for situation (1), let A, B, C always win when they play other players. Among A, B, C we shall assign the order $A > C$, $B > C$. In other cases we do not care.

CS2. As a follow-up strategy for situation (2), let A', B' always win when they play other players. In other cases, we do not care.

Now, in the case of situation (1) followed up by CS1, player C will necessarily be selected as the third best player, dominated by A and B. But then, none of the $h(n)$ matches as mentioned in situation (1) is a crucial match as far as selecting the third best player is concerned. Hence the entire tournament must contain at least $n - 1 + h(n) = H(n)$ matches.

Similarly, in the case of situation (2) followed up by CS2, A' and B' necessarily become the two top winners of the tournament. Among those $h(n) + 1$ first defeats made by A' and B' before time t, one could be the first defeat of the third best player. However, this is the only case where those $h(n) + 1$ matches may contain a crucial one. Since there ought to be $n - 2$ other crucial matches, we see that the length of the tournament is at least $n - 2 + (h(n) + 1)$, which equals $H(n)$. Theorem 1 is thus shown to follow from Lemma 1.

2.3.2 Proof of Lemma 1

It is clear that any BS-ruled tournament that selects the third best player must fall into one of the following two classes:

(i) Those in which there is only one undefeated player left at the end.

(ii) Those in which there are two undefeated players left at the end.

For the proof of Lemma 1, we need only consider case (i). Indeed, suppose the lemma is proved in this case. Let T be a tournament of class (ii), so that two undefeated players say x and y are found at time $|T|$. If $Q_{|T|}(y) \leq Q_{|T|}(x)$, we may extend T by letting x play and defeat y in an additional match. Call the resulting tournament T'. Clearly T' has been following Basic Strategy, and moreover is in class (i), so the lemma is true for T'. However, the particular time t in T' as characterized by the lemma must satisfy $t \leq |T|$, since at time $|T|+1$ there is but one undefeated player x, not satisfying either assertion (1) or (2). Hence the lemma must be in fact true for T.

Let us now look at the first-defeat-tree of a tournament of class (i). (cf. Figure 4) The only undefeated player, namely the champion, is denoted by x_c. Let $x_0, x_1, \ldots, x_s$ be the sons of x_c arranged in the order their first defeats by x_c took place. Thus if t_j, $0 \leq j \leq s$, is the time x_c inflicted x_j's first loss, then $t_0 < t_1 < \cdots < t_s$. For $0 \leq j \leq s$, let d_j denote the number of first defeats made by x_j. (By BS1, they must all take place before time t_j.)

<u>Claim</u> Let $M = \max \{ j + d_j \mid 0 \leq j \leq s \}$

and $\ell = \min \{ j \mid 0 \leq j \leq s, j + d_j = M \}$, then

(I) if $\ell < s$, we must have $M \geq h(n)$.

(II) if $\ell = s$, we must have $M \geq h(n) + 1$.

Note that once Claim is proved, Lemma 1 will follow immediately. For, in case (I) is true, we can choose t to be the time $t_\ell - 1$.

At this moment, x_c, x_ℓ and $x_{\ell+1}$ (note $\ell+1 \le s$) are all undefeated since $t_\ell - 1 < t_\ell < t_{\ell+1}$. By this time x_c has made ℓ first defeats (namely on $x_0, x_1, \ldots, x_{\ell-1}$), and x_ℓ has made d_ℓ. Since by (I) $\ell + d_\ell = M \ge h(n)$, assertion (1) of Lemma 1 is fulfilled if we choose A, B, C to be x_c, x_ℓ, and $x_{\ell+1}$. On the other hand, if (II) of Claim is true, we can choose t to be $t_s - 1$. At this time x_c and x_s are undefeated, and have made a total of $s + d_s = M \ge h(n) + 1$ first defeats. Thus choosing A' and B' to be x_c and x_s will satisfy assertion (2) of Lemma 1.

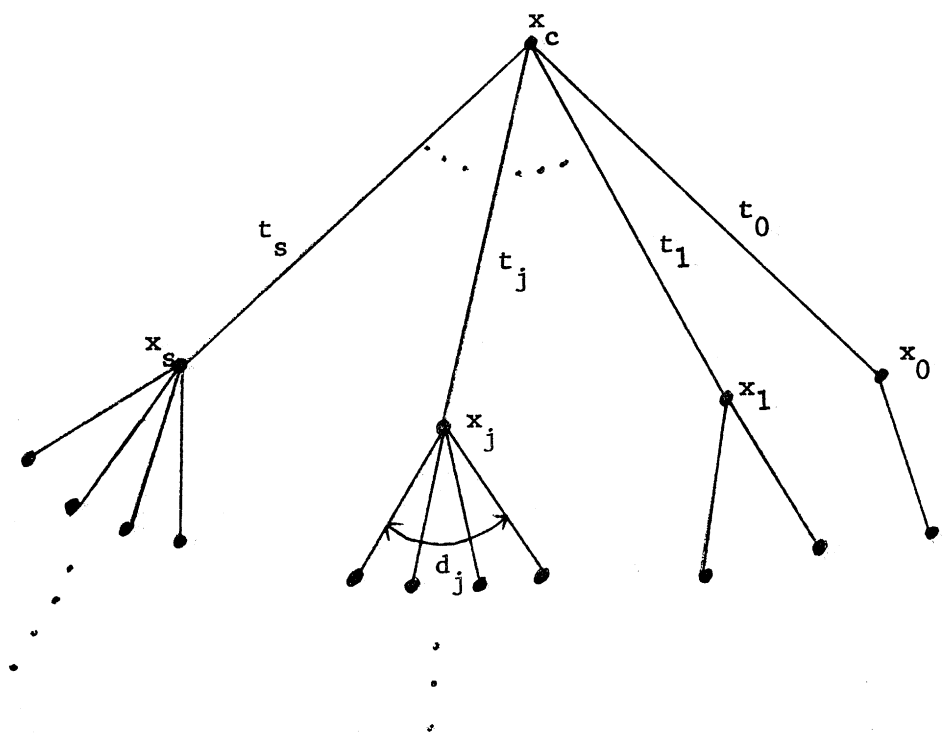


Figure 4 FDT with a single root

So we have reduced the proof of Lemma 1 to the verification of cases (I) and (II) of Claim.

Proof of Claim

(I) Assume $\ell < s$.

By the definition of M, we have $j + d_j \le M$ for all $0 \le j \le s$. In particular,

$$d_{\lceil M/2 \rceil} \le M - \lceil M/2 \rceil = \lfloor M/2 \rfloor$$

$$d_{\lceil M/2 \rceil + 1} \le M - (\lceil M/2 \rceil + 1) = \lfloor M/2 \rfloor - 1$$

$$\vdots$$

$$d_s \le M - s$$

By Fact 2, $Q(x_j) \le 2^{d_j}$ for all $0 \le j \le s$. Hence we have

$$Q(x_{\lceil M/2 \rceil}) \le 2^{\lfloor M/2 \rfloor}$$

$$Q(x_{\lceil M/2 \rceil + 1}) \le 2^{\lfloor M/2 \rfloor - 1}$$

$$\vdots$$

$$Q(x_s) \le 2^{M-s}$$

Summing the weights, we get

$$\sum_{j=\lceil M/2 \rceil}^{s} Q(x_j) \le 2^{\lfloor M/2 \rfloor} + 2^{\lfloor M/2 \rfloor - 1} + \cdots + 2^1 + 2^0$$

$$= 2^{\lfloor M/2 \rfloor + 1} - 1 \qquad (4)$$

Also, x_c has only made j first defeats prior to time t_j, hence by Fact 2 of Sec. 2.2 has weight at most 2^j before he plays x_j. This implies that $Q(x_j) \le 2^j$, for otherwise x_j would not lose to x_c by BS2 of Basic Strategy. In particular, this is true for x_j such that

$0 \le j < \lceil M/2 \rceil$. Therefore,

$$\sum_{j=0}^{\lceil M/2 \rceil - 1} Q(x_j) \le 2^0 + 2^1 + \cdots + 2^{\lceil M/2 \rceil - 1} = 2^{\lceil M/2 \rceil} - 1 \qquad (5)$$

On the other hand, since in the first-defeat-tree the union of the subtrees rooted at x_j, $0 \le j \le s$, contains every node except x_c, by Fact 3 of Sec. 2.2 we have

$$\sum_{j=0}^{s} Q(x_j) = n - 1 \qquad (6)$$

Now, from Equs. (4), (5) and (6), we obtain

$$2^{\lceil M/2 \rceil} + 2^{\lfloor M/2 \rfloor + 1} - 2 \ge n - 1$$

Solving this inequality for M and compare with the definition of $h(n)$, we conclude that

$$M \ge h(n) .$$

(II) Assume $\ell = s$.

By the definition of ℓ, we have $M = s + d_s$ but $j + d_j \not\le M$ for all $j < s$. Hence,

$$d_{\lceil M/2 \rceil} < M - \lceil M/2 \rceil = \lfloor M/2 \rfloor$$

$$d_{\lceil M/2 \rceil + 1} < \lfloor M/2 \rfloor - 1$$

$$\vdots$$

$$d_{s-1} < M - s + 1$$

$$d_s = M - s$$

Again, by Fact 2 we have

$$\sum_{j=\lceil M/2 \rceil}^{s} Q(x_j) \le 2^{\lfloor M/2 \rfloor - 1} + \cdots + 2^{M-s} + 2^{M-s} = 2^{\lfloor M/2 \rfloor} \qquad (7)$$

It then follows from Eqns. (5), (6), (7) that

$$2^{\lceil M/2 \rceil} + 2^{\lfloor M/2 \rfloor} - 1 \geq n - 1$$

As a result, we get

$$M \geq h(n) + 1 .$$

Thus, Claim has been proved, and the proof of Lemma 1 is now complete.

CHAPTER 3

GENERAL LOWER BOUNDS FOR SMALL i

3.1 Introduction

In this chapter, we shall generalize the techniques of Chapter 2 to deduce lower bounds on $V_i(n)$ and $W_i(n)$ for small i. Although we appeal to the same scheme of using a two-phase strategy, the analysis involved for $i > 3$ is considerably more difficult. The major effort is contained in the proof of Lemma 2, which is a generalization of the analysis we did on "first-defeat-trees" in Lemma 1 of Sec.2.3.

For any fixed i, our lower bound for $V_i(n)$ and $W_i(n)$ has the asymptotic behavior $n + (i-1)\log n - O(\log(\log^* n))$. By comparing it with upper bounds as given by Equs.(1) and (2) of Sec. 1.1, we see that the asymptotic behaviors of both $V_i(n)$ and $W_i(n)$ are determined to within a term of order $O(\log(\log^* n))$.

3.2 Main Theorem

We shall derive lower bounds on $V_i(n)$ and $W_i(n)$ for $i > 3$. by induction on i, based on the result established for $i = 3$ in the previous chapter. The resulting formulas for both of our lower bounds involve a quantity which is dependent on i as well as on n. This quantity, formally defined below as $h_i(n)$, is fairly close to $\log(n)$ when i is small relative to n.

Notation For $n \geq 2$, let $\log^* n$ denote the largest integer k such that

$$n \geq 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} k .$$

Thus, $\log^* 2 = \log^* 3 = 1$, $\log^* 4 = \log^* 5 = \cdots = \log^* 15 = 2$, and so forth.

Definition 4 For any $i \geq 3$, define $h_i(n)$ by the following formula:

$$\left.\begin{aligned} h_3(n) &= \log n - 2; \\ \text{for } i > 3,\ h_i(n) &= \log n - (i-3)\log(\log^* n) - 2\log((i-1)!) - (i-1) . \end{aligned}\right\} (8)$$

Our main results of this chapter are contained in the following theorem:

Theorem 2 For any n and i such that $n > i \geq 3$, we have

$$V_i(n) \geq n - i + (i-1)(h_i(n) - 1) \tag{9}$$

$$W_i(n) \geq n - 1 + (i-1)h_1(n) \tag{10}$$

Remark 1 If we define $h_1(n) = h_2(n) = \lceil \log n \rceil$, then the lower bounds as given by Theorem 2 also hold when i=1, 2. In fact, they are close to the precise bounds that are known in these cases:

$$V_1(n) = W_1(n) = n - 1$$
$$V_2(n) = W_2(n) = n - 2 + \lceil \log n \rceil \qquad \text{(Kislitsyn's Theorem)}$$

Remark 2 For i=3, Theorem 2 gives the following lower bounds:

$$V_3(n) \geq n - 3 + 2(\log n - 3)$$
$$= n - 9 + 2\log n ,$$
$$W_3(n) \geq n - 3 + 2(\log n - 2)$$
$$= n - 7 + 2\log n .$$

These are slightly worse than the lower bounds we actually proved for i=3 in Theorem 1 of Chapter 2. The reason is of course that here we would rather use a varied (and slightly weakened) form of Lemma 1, if only it could serve more conveniently as a basis for inductive arguments. (cf. Lemma 2 and its proof in Sec.3.4.2)

Remark 3 Compare the lower bounds of Theorem 2 with the upper bounds of Hadian-Sobel and Kislitsyn respectively. From (8), we have

$$h_i(n) > \lceil \log n \rceil - (i-3)\log(\log^* n) - 2\log((i-1)!) - i$$

since $\log + 1 > \lceil \log n \rceil$. Hence by comparing Equs. (9), (10) with

(1) and (2) of Sec.1.1, we see that the gap between upper and lower bounds on $V_i(n)$ and $W_i(n)$ is now less than

$$(i-1)(i-3)\log(\log^* n) + (i-1)[2\log((i-1)!) + i + 1] .$$

As a consequence, for any fixed $i > 3$, the asymptotic behaviors of $V_i(n)$ and $W_i(n)$ are determined to within

$$(i-1)(i-3)\cdot\log(\log^* n) + O(1) .$$

Remark 4 For large n and i, since

$$\log((i-)!) \approx \log(i!) \approx i(\log i) ,$$

we see from the definition of $h_i(n)$ in (8) that

$$h_i(n) \approx \log n - i(2\log i + \log(\log^* n)).$$

Hence $h_i(n)$ approaches zero at $i \approx (\log n)/(2\log(\log n))$.

Remark 5 When $h_i(n)$ gradually approaches zero or even becomes negative as i grows larger and larger, formula (9) is superseded by Pratt's lower bound as given by Equ.(3) of Sec.1.1:

$$V_i(n) \geq n + 2i - \log n \qquad \text{for } i \leq n/3 ,$$
$$V_i(n) \geq (3n+i)/2 - \log n \qquad \text{for } n/3 \leq i \leq n/2 .$$

Also, (10) may be replaced by the following lower bound on $W_i(n)$:

Theorem 3 For any n and i such that $n \geq i$,

$$W_i(n) \geq n - i + \lceil \log(i!) \rceil .$$

Proof This is purely by information-theoretical arguments. Suppose in a tournament $A_1, \ldots, A_i$ (unordered) are known to be the best i players, and the order of the remaining $n-i$ players is fixed. Then, just in order to completely determine the ranking of these i top players, $\lceil \log(i!) \rceil$ matches between them could be necessary in the worst case. However, for each of the remaining $n-i$ players, there still must be a match in the tournament in which he lost to someone whose ranking is no higher than the i-th. Hence these $n-i$ "crucial" matches are completely disjoint from those $\lceil \log(i!) \rceil$ mentioned before, and this tournament contains a total of $n - i + \lceil \log(i!) \rceil$ matches at least.

3.3 Proof of Main Theorem

The proof of Theorem 2 depends on the following lemma the way Theorem 1 of Sec. 2.3 depends on Lemma 1. Indeed, Lemma 2 generalizes the analysis of the first-defeat-tree as was done in Lemma 1. Here a "championship" tournament is a tournament which determines, among other things, the best player of all.

Lemma 2 Let T be a BS-ruled championship tournament of n players. For any i such that $i \geq 3$, we assert that there must exist a time t in T when the following two statements are true:

$(i)_i$ There are $\leq i-1$ undefeated players who have made a total of $(i-1)h_i(n)$ first defeats at least.

$(2)_i$ The champion himself has made at least $h_i(n)$ first defeats.

(The second statement serves mainly as an induction hypothesis that is needed for the proof of Lemma 2.)

As in Sec. 2.3, let us first assume that the lemma holds and prove Theorem 2. The proof of Lemma 2 will be given in Sec. 3.4 .

Proof of Theorem 2

Given an algorithm for selecting the i-th best player, we can find the champion by using at most $V_1(i-1) = i-2$ additional matches

to determine the best among the top i-1 players. This means that if $V_i'(n)$ is the minimum number of comparisons for selecting the largest and the i-th largest element, then

$$V_i(n) \geq V_i'(n) - (i - 2)$$

and

$$W_i(n) \geq V_i'(n) \ .$$

Thus, to prove Theorem 2, it suffices to show that

$$V_i'(n) \geq n - i + (i-1)h_i(n) \ .$$

Given an algorithm for selecting the champion and the i-th best player, we first apply Basic Strategy to it. By Lemma 2 there must be a crucial moment t when there are j ($1 \leq j \leq i-1$) undefeated players $A_1, \ldots, A_j$ who have made a total of $(i-1)h_i(n)$ first defeats at least. As soon as this occurs, we shall switch to a Clear Strategy. Under the latter strategy, $A_1, \ldots, A_j$ shall always win when they play any of the remaining n - j guys; in other circumstances the outcome may be arbitrary. Now, $A_1, \ldots, A_j$ necessarily turn out to be the j highest ranking players of the tournament. However, as far as determining the i-th best player is concerned, $V_1(n-i+1) = n - i$ matches have to be played between the n-i+1 low ranking players in order to determine the best among them. Since these players are completedly disjoint from the j highest ranking players, we see that the tournament must contain no fewer than $n - i + (i-1)h_i(n)$ matches all together.

3.4 Proof of Lemma 2

3.4.1 Auxiliary Propositions

Lemma 2 will be proved by induction on i. In trying to proceed from say $i=q$ to $i=q+1$, the problem of delicate "timing" in the tournament turns out to be of vital importance. Propositions 1 and 2 below are two instances of how one deals with this problem in the proof of Lemma 2.

As in the proof of Lemma 1, we consider the first-defeat-tree of T, the given BS-ruled championship tournament. Let $x_0, \ldots, x_s$ be those players whose first defeats are made by x_c, the champion, at time $t_1, \ldots, t_s$, respectively with $t_0 < t_1 < \cdots < t_s$. Also let d_j be the number of first defeats made by x_j, $0 \le j \le s$. (cf. Figure 4 of Sec. 2.3.2) Furthermore, for any x_j, $0 \le j \le s$, let T_j be the "subtournament" of T consisting of those matches (in the order they occur in T) represented by the edges of the subtree, rooted at x_j, of the FDT. Let T_j' denote the subtournament consisting of those matches of $T_1, \ldots, T_{j-1}$ together with the first defeats of $x_1, \ldots, x_{j-1}$ (also in the order they occur in T). (cf. Figure 5) One can easily see that T_j and T_j', when considered as tournaments themselves, are both BS-ruled championship tournaments. Within T, both subtournaments T_j and T_j' are completed before time t_j.

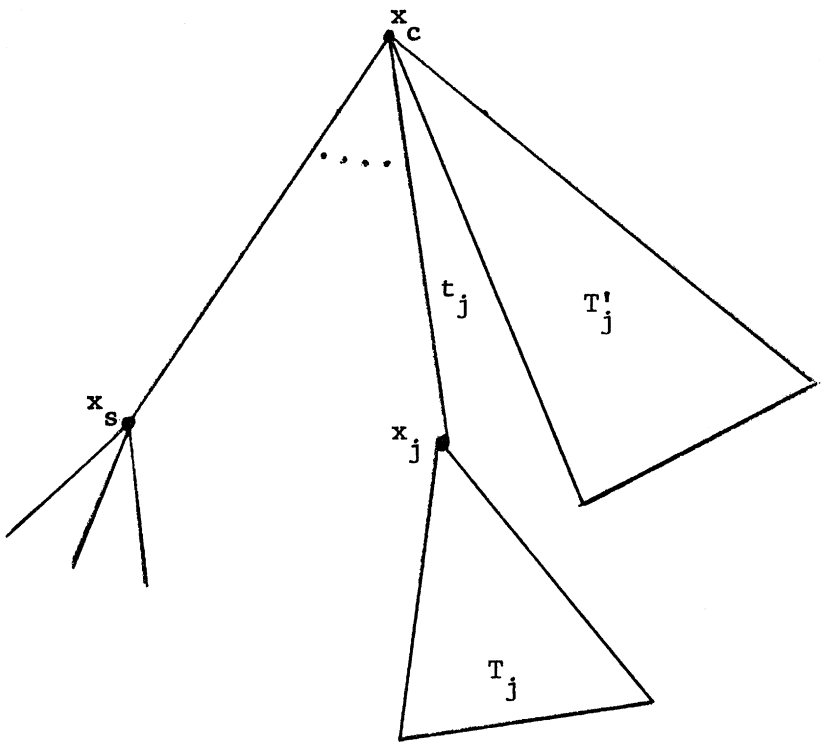


Figure 5 Subtournaments T_j and T'_j

In terms of the notations defined in the last paragraph, we have

Proposition 1 Suppose Lemma 2 is true for $i = q \geq 3$. If for a given BS-ruled championship tournament T (of n players), there exist j and m satisfying the following conditions, then Lemma 2 is also true in T when $i = q+1$:

(i) $0 \leq j \leq s$ and $m \leq n$,

(ii) $Q(x_j) \geq m$ and $h_q(m) \geq h_{q+1}(n)$.

Proof Since x_c defeated x_j at time t_j, by Basic Strategy we must have

$$Q_{t_j-1}(x_c) \geq Q_{t_j-1}(x_j) .$$

Since

$$Q_{t_j-1}(x_j) = Q(x_j) \geq m ,$$

this means that both tournaments T_j and T_j' involve at least m players.

By our induction hypothesis, there exists a time t in T_j when

$(1)_q$ some $\leq q-1$ undefeated players in T_j have made a total of $(q-1)h_q(m)$ first defeats at least;

$(2)_q$ x_j (the champion of T_j) by this time has made at least $h_q(m)$ first defeats.

Similarly, there exists a time t' in T_j' when

$(1)_q'$ —— same as statement $(1)_q$, only T_j is replaced by T_j' ;

$(2)_q'$ —— same as statement $(2)_q$, only x_j is replaced by x_c .

Now, if $t' < t$, then $(2)_q'$ is also true at time t. Hence at time t, by $(1)_q$ and $(2)_q'$, there are $\leq q$ undefeated players who have made a total of at least $q \cdot h_q(m)$ first defeats. Since by assumption $h_q(m) \geq h_{q+1}(n)$, this together with statement $(2)_q'$ above show that both assertions of Lemma 2 are true in T for $i=q+1$.

On the other hand, if $t<t'$, we will choose the time to be t'. Then statement $(1)'_q$ and $(2)_q$ above show that at time t' some $\leq q$ undefeated players have made a total of at least $q \cdot h_q(m) \geq q \cdot h_{q+1}(n)$ first defeats. Again, this together with $(2)'_q$ show that Lemma 2 is satisfied for $i=q+1$.

Proposition 2 Suppose Lemma 2 is true for $i = q \geq 3$. If for a given BS-ruled championship tournament T of n players, there exists an m and q indices named $\alpha, j_1, \ldots, j_{q-1}$, that satisfy the following conditions, then Lemma 2 is also true in T when $i=q+1$:

(i) $m \leq n$, $0 \leq \alpha \leq j_1 < j_2 < \ldots < j_{q-1} \leq s$ and $\alpha \geq \log n$;

(ii) $Q(x_{j_\ell}) \geq m$ for $1 \leq \ell \leq q-1$, and $\alpha + (q-1)h_q(m) \geq q \cdot h_{q+1}(n)$.

(cf. Figure 6)

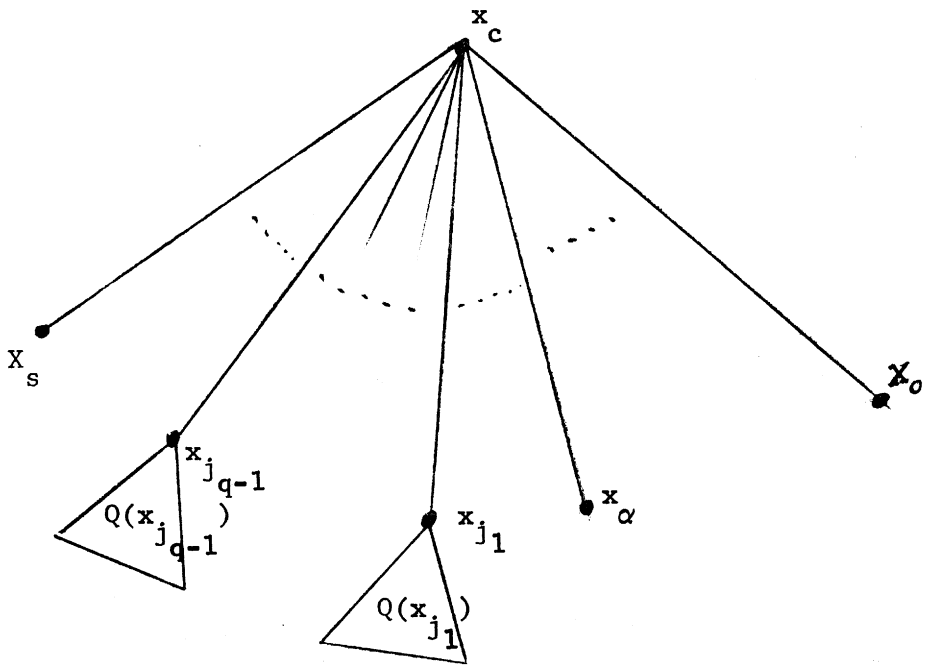


Figure 6 A tournament of Proposition 2

Proof Since $Q(x_{j_\ell}) \geq m$ for $1 \leq \ell \leq q-1$, this means that each of the $q-1$ subtournaments $T_{j_1}, \ldots, T_{j_{q-1}}$ involves at least m players.

Applying the induction hypotheses, we see that for $1 \leq \ell \leq q-1$, there exists time t'_ℓ, $t'_\ell < t_{j_\ell}$, such that at time t'_ℓ the following two statements are true:

$(1)^\ell_q$ Some $\leq q-1$ undefeated players in T_{j_ℓ} have made a total of $(q-1)h_q(m)$ first defeats at least.

$(2)^\ell_q$ x_{j_ℓ} himself has made at least $h_q(m)$ first defeats.

There are two cases:

(I) $t'_\ell < t_\alpha - 1$ for all $1 \leq \ell \leq q-1$.

If this is the case, then at time $t_\alpha - 1$, x_c has made α first defeats (namely on $x_0, \ldots, x_{\alpha-1}$). Also, since $t_{j_\ell} \geq t_\alpha > t_\alpha - 1$, x_{j_ℓ} are all undefeated at time $t_\alpha - 1$. Hence, by $(2)^\ell_q$, we know that at time $t_\alpha - 1$, x_c and x_ℓ for $1 \leq \ell \leq q-1$ are q undefeated players who have made a total of $\alpha + (q-1)h_q(m)$ first defeats. Since by assumption

$$\alpha \geq \log n, \quad \text{hence} \quad \alpha > h_{q+1}(n),$$

and

$$\alpha + (q-1)h_q(m) \geq q \cdot h_{q+1}(n),$$

we see that Lemma 2 is true for this tournament when $i=q+1$.

(II) $t'_\ell \geq t_\alpha - 1$ for some ℓ such that $1 \leq \ell \leq q-1$.

For this particular ℓ, x_c at time t'_ℓ has again made at least α first defeats. And by statement $(1)^\ell_q$, at time t'_ℓ there are $\leq q-1$

undefeated players who together with x_c have made at least $\alpha + (q-1)h_q(m) \geq q \cdot h_q(n)$ first defeats. Hence again Lemma 2 is true when $i=q+1$ for this tournament. Thus the proof of Proposition 2 has been completed.

3.4.2 The Inductive Proof

We are now ready to prove Lemma 2.

A) Basis of induction, $i=3$.

Let us look at Figure 4 of Sec. 2.3.2. Suppose Lemma 2 were false, we would have $j + d_j < 2\log n - 4$ for all j such that $\log n - 2 \leq j \leq s$. (If not so, we can choose the time to be $t_j - 1$ where $j \geq \log n - 2$ and $j + d_j \geq 2\log n - 4 = 2h_3(n)$. At this time $t_j - 1$, both statements $(1)_3$ and $(2)_3$ of Lemma 2 are satisfied.) Thus,

$$
\begin{aligned}
d_{\lceil \log n \rceil - 2} &< 2\log n - 4 - (\lceil \log n \rceil - 2) \leq \lceil \log n \rceil - 2 \\
d_{\lceil \log n \rceil - 1} &< \lceil \log n \rceil - 3 \\
&\vdots \\
d_s &< 2\lceil \log n \rceil - 4 - s
\end{aligned}
$$

From Fact 2 of Sec. 2.2, we have

$$
\begin{aligned}
\sum_{\lceil \log n \rceil - 2}^{s} Q(x_j) &\leq 2^{\lceil \log n \rceil - 3} + 2^{\lceil \log n \rceil - 4} + \cdots + 2^1 + 2^0 \\
&= 2^{\lceil \log n \rceil - 2} - 1 \qquad (11)
\end{aligned}
$$

(Note, if $\lceil \log n \rceil - 2 \leq 0$, then $h_i(n) \leq 0$ for all $i \geq 3$, and Lemma 2 is trivially true in this case. Hence we may assume $n > 4$.)

From (11)and Equs. (5), (6) of Sec. 2.3, we get

$$n - 1 = \sum_{j=0}^{\lceil \log n \rceil - 3} Q(x_j) + \sum_{j=\lceil \log n \rceil - 2}^{s} Q(x_j)$$

$$\leq 2^{\lceil \log n \rceil - 2} - 1 + 2^{\lceil \log n \rceil - 2} - 1$$

$$\leq n - 2$$

which is a contradiction. This proves Lemma 2 for $i=3$.

B) Suppose the lemma is true for $i=q\geq 3$, we will proceed to prove it for $i= q+1$.

If for the given tournament, $s \geq q \cdot \log n$, then at time say t_{s-q} , x_c and x_{s-q+1}, x_{s-q+2}, ..., x_{s-1} are q undefeated players where x_c alone has made $s-q+1$ first defeats (namely on x_0, ..., x_{s-q}). Since $s-q+1 > q \cdot \log n - q = q(\log n - 1) \geq q \cdot h_{q+1}(n)$, Lemma 2 is true for $i=q+1$. Hence from now on we shall assume $s < q \cdot \log n$.

Notation For $1 \leq k \leq \log^* n$, let $\log^{(k)} n$ denote $\underbrace{\log \cdots \log}_{k} n$, and i_k denote $\log n + (q-1)\log^{(k)} n$. Also, let $n' = n/(2\log^* n)$.

We shall divide the problem up into $\log^* n$ cases. Case k, for $1 \leq k \leq \log^* n - 1$, makes the assumption

$$\sum_{i_{k+1} \leq j < i_k} Q(x_j) \geq n' .$$

The last case, Case log*n , makes the assumption

$$\sum_{0\le j<2(q-1)+\log n} Q(x_j) > n/2 - 1 .$$

It is seen that these log*n cases do cover all possibilities since we always have

$$\sum_{i_2\le j<i_1} Q(x_j) + \sum_{i_3\le j<i_2} Q(x_j) + \cdots + \sum_{0\le j<\log n + 2(q-1)} Q(x_j)$$

$$> \sum_{0\le j\le s} Q(x_j) \qquad \text{(because } i_{\log^* n} < \log n + 2(q-1). \text{)}$$

$$= n - 1$$

$$\ge n/(2\log^* n) + \underbrace{n/(2\log^* n) + \cdots}_{\log^* n - 1} + (n/2 - 1) .$$

Now, Case k , $1\le k\le \log^* n - 1$, are proved in the same way.

Proof of Case k, $1\le k\le \log^* n - 1$

There are two possibilities:

(I) The maximum of $Q(x_j)$, $i_{k+1}\le j<i_k$, is $\ge n'/(q-1)$

If only we can show $h_q(n'/(q-1)) \ge h_{q+1}(n)$, then the desired result will follow from Proposition 1 of Sec. 3.4.1. Indeed,

$$h_q(n'/(q-1)) \ge \log n - \log(q-1) - 1 - \log(\log^* n) - (q-3)\log(\log^* n) - 2\log((q-1)!) - (q-1)$$

$$> \log n - (q-2)\log(\log^* n) - 2\log(q!) - q$$

$$= h_{q+1}(n) \ .$$

Therefore we are done in this case.

(II) The maximum of $Q(x_j)$, $i_{k+1} \le j < i_k$, is $< n'/(q-1)$.

In this case, let $Q(x_{j_1}), \ldots, Q(x_{j_{q-1}})$ be the $q-1$ largest weights among $Q(x_j)$, $j \in [i_{k+1}, i_k)$. (note $i_k - i_{k+1} \ge q-1$.)

Then we must have $Q(x_{j_\ell}) \ge n'/((q-1)(i_k - i_{k+1}))$ for all $1 \le \ell \le q-1$.

Denote this last fraction by m. If we can show that $i_{k+1} + (q-1)h_q(m) \ge q \cdot h_{q+1}(n)$, then the conclusion will follow from Proposition 2 of Sec. 3.4.1. Thus, since $i_k - i_{k+1} \le (q-1)\log^{(k)} n$, and $n' = n/(2\log^* n)$,

$$\begin{aligned} i_{k+1} + (q-1)h_q(m) &\ge \log n + (q-1)\log^{(k+1)} n + (q-1)(\log n - 2\log(q-1) \\ &\quad - \log^{(k+1)} n - 2\log((q-1)!) - q - (q-2)\log(\log^* n)) \\ &\ge q(\log n - (q-2)\log(\log^* n) - 2\log(q!) - q) \\ &= q \cdot h_{q+1}(n) \ . \end{aligned}$$

This completes the proof of Case k, $1 \le k \le \log^* n - 1$.

Proof of Case log*n

In this case we have the assumption

$$\sum_{0\le j<2(q-1)+\log n} Q(x_j) > n/2 - 1 . \qquad (12)$$

But by Equ.(5) of Sec.2.3 we have

$$\begin{aligned}\sum_{0\le j\le \lceil \log n\rceil -3} Q(x_j) &\le 2^0 + 2^1 + \cdots + 2^{\lceil \log n\rceil -3} \\ &\le 2^{\lceil \log n\rceil -2} - 1 \\ &= n/4 - 1 . \qquad (13)\end{aligned}$$

From Equs. (12) and (13), we get

$$\sum_{\log n-3<j<\log n+2(q-1)} Q(x_j) > n/4 .$$

Hence the maximum of $Q(x_j)$, $\log n-3 < j < \log n+2(q-1)$, must be greater than $n/8(q+1)$.

As in (I) of the proof of Case k, $k < \log^* n$, we need only show that $h_q(n/8(q+1)) \ge h_{q+1}(n)$, then the result will follow immediately from Proposition 1 of Sec.3.4.1. The calculation is straightforward:

$$\begin{aligned}h_q(n/8(q+1)) &\ge \log n - 2 - \log(2q+2) - (q-3)\log(\log^* n) - 2\log((q-1)!) - (q-1) \\ &\ge \log n - (q-2)\log(\log^* n) - 2\log(q!) - q \\ &= h_{q+1}(n).\end{aligned}$$

(We assume here $n > 4$ due to the remark following Equation (11).)
This proves Case log*n, and the proof of Lemma 2 is now complete.

CHAPTER 4

FINDING A MEDIOCRE PLAYER

4.1 Introduction

In the preceding chapters we have successfully extended Kislitsyn's lower bound to the case when, instead of finding the second best of n players, the i-th best is to be determined. We now turn our attention to another type of selection problems.

Given i and j with $i + j + 1 \leq n$, our objective now is to find a player who is neither among the i top players nor one of the j worst players. We shall say such a player is (i,j)-mediocre. Historically this problem is closely connected with the finding of the median element, whose starting point is usually the selection of an element that is not too close to either extreme.

Technically, the selection of the i+1st largest element is a special case of this "mediocre player" problem in which $n = i+j+1$. This connection suggests a way of finding an (i,j)-mediocre player. We simply pick i+j+1 elements arbitrarily and select the i+1st largest among them; it is obvious that this element satisfies the "(i,j)-mediocre" requirement. The question that naturally follows is "Is this the best algorithm ?" In Sec. 4.3 we will prove that this is the case when i=1. We will also show that if the answer to this

question is "yes" in general, then there is the surprising implication that the median can be computed by using no more than in comparisons. We will discuss this and some properties of the optimal algorithms in the following sections.

4.2 Properties of S(i,j,n)

For $i+j+1 \le n$, let $S(i,j,n)$ denote the minimum number of comparisons needed to find an (i,j)-mediocre element as defined in Sec.4.1. The following facts are easy to verify:

(i) $S(i,j,n') \le S(i,j,n)$ if $n' \ge n \ge i+j+1$.

This is true since from n' elements we can arbitrarily choose n elements and in $S(i,j,n)$ comparisons find a desired (i,j)-mediocre element.

(ii) There is an integer $n_0 \le 12(i+j+1)$ such that $S(i,j,n) = S(i,j,n_0)$ for all $n \ge n_0$.

Proof By the algorithm of Blum et al[1], it takes less than $6m$ comparisons to find the t-th largest of m elements for any $t \le m$. Since $S(i,j,i+j+1) = V_{i+1}(i+j+1)$, we thus have $S(i,j,n) \le S(i,j,i+j+1) \le 6(i+j+1)$ for all $n \ge i+j+1$ by (i). Since an optimal algorithm never consults more than $12(i+j+1)$ elements, we must have

$$S(i,j,n) \ge S(i,j,12(i+j+1)) \quad \text{for} \quad n \ge 12(i+j+1).$$

Thus, for fixed i, j, the function S is non-increasing in n and reaches a constant before n is $12(i+j+1)$.

As a result, we have

Theorem 4 For $n \geq i+j+1$, $S(i,j,n)$ is independent of n if and only if selecting (optimally) the $i+1$st largest element of an arbitrary subset of $i+j+1$ elements is an optimal procedure for finding an (i,j)-mediocre element in n elements.

Proof If $S(i,j,n)$ is independent of n, then

$$S(i,j,n) = S(i,j,i+j+1) = V_{i+1}(i+j+1) .$$

This shows that the proposed algorithm is optimal. On the other hand, if the algorithm described in the theorem is optimal, we will have

$$S(i,j,n) = V_{i+1}(i+j+1) \quad \text{for all} \quad n \geq i+j+1$$

which shows that $S(i,j,n)$ is independent of n.

4.3 $S(1,j,n) = V_2(j+2)$

Because of Theorem 4 of the last section, it becomes an intriquing question whether $S(i,j,n)$ is independent of n or not. In this section we will show that $S(1,j,n)$ *is* independent of n. This lends support to the conjecture that $S(i,j,n)$ depends only on i and j. We will discuss some implications of this conjecture in the next section.

Theorem 5 $S(1,j,n) = V_2(j+2) = j + \lceil \log(j+2) \rceil$

Proof It suffices to show that $S(1,j,n) \geq j + \lceil \log(j+2) \rceil$. For any given algorithm that finds a $(1,j)$-mediocre player, we shall apply Basic Strategy (cf. Sec. 2.2) to it and call the resulting tournament T. Suppose x_m is the $(1,j)$-mediocre player found in T, define

$$A = \{x \mid \text{either } x = x_m \text{ or } x \text{ is determined to be inferior to } x_m \text{ by T}\}.$$

By the definition of "$(1,j)$-mediocre" x_m must be defeated at least once in T; hence none of the guys in A is an undefeated player. Also, let

$$B = \{y \mid y \text{ is undefeated at the end of T and has weight} > 1\}.$$

Thus, if $B = \{y_1, \ldots, y_r\}$, then $y_1, \ldots, y_r$ are those players corresponding to the roots that have at least one descendant (besides

itself) in the first-defeat-tree of T. Certainly $A \cap B = \phi$.

Claim If for $1 \le \ell \le r$, y_ℓ has weight m_ℓ, then we must have

$$\sum_{\ell=1}^{r} \lceil \log(m_\ell) \rceil \ge \lceil \log(j+2) \rceil .$$

Proof of Claim Suppose for $1 \le \ell \le r$, we have $2^{k_\ell} < m_\ell \le 2^{k_\ell+1}$.

Then,

$$\begin{aligned} \sum_{\ell=1}^{r} \lceil \log(m_\ell) \rceil &= (k_1 + 1) + \cdots + (k_r + 1) \\ &= r + (k_1 + \cdots + k_r) . \end{aligned} \tag{14}$$

On the other hand,

$$\begin{aligned} m_1 + \cdots + m_r &\le 2^{k_1+1} + \cdots + 2^{k_r+1} \\ &\le 2^{k_1+1} \cdots 2^{k_r+1} \\ &= 2^r \cdot 2^{(k_1 + \cdots + k_r)} . \end{aligned}$$

Hence

$$\lceil \log(m_1 + \cdots + m_r) \rceil \le r + (k_1 + \cdots + k_r). \tag{15}$$

However, since every player in A must be the descendant of some y_ℓ, $1 \le \ell \le r$, we have

$$m_1 + \cdots + m_r \ge r + |A| \ge 1 + |A| . \tag{16}$$

Since by the definition of (i,j)-mediocre we must have $|A| \ge j+1$, Equ.(16) becomes

$$m_1 + \cdots + m_r \ge j + 2 . \tag{17}$$

The Claim then follows from Equs. (14), (15) and (17).

Now that Claim has been proved, Theorem 5 follows quite easily. Indeed, from Claim and Fact 2 of Sec. 2.2, we know that $y_1, \ldots, y_r$ have played at least $\lceil \log(j+2) \rceil$ matches all together. But from the definition of A, it follows that at least $|A| - 1$ matches have been played between the players of A (one "crucial" match for each player in A other than x_m). Since A and B are disjoint, we see that this tournament must contain at least

$$\lceil \log(j+2) \rceil + |A| - 1 \geq \lceil \log(j+2) \rceil + j$$

matches. Thus the proof of Theorem 5 is complete.

4.4 Connections with Median Computation

In the last section we proved that $S(1,j,n)$ is independent of n. This by no means implies $S(i,j,n)$ is independent of n for other i as well. However, if this should indeed be true, it would have the following interesting consequence.

Proposition If $S(i,j,n)$ is independent of n for all i,j, then one can find the median of n elements by using no more than 3n comparisons.

Proof Let $M(n)$ be the minimum number of comparisons for finding the median of n elements. Then, for a given n, to find an $(n/2, n/2)$-mediocre element among $3n/2$ elements one can proceed in two ways:

1) Pick any $n+1$ elements and find their median. This requires $M(n+1)$ comparisons in the worst case.
2) Divide the $3n/2$ elements into $n/2$ triplets and sort each triplet. Then take the central elements of all the triplets and find their median. This element is easily seen to be a desired $(n/2, n/2)$-mediocre element, and this method requires $3\cdot(n/2) + M(n/2)$ comparisons in the worst case.

Now, if $S(i,j,n)$ is independent of n then the first method is optimal by Theorem 4. Therefore we have

$$M(n+1) \le 3n/2 + M(n/2),$$

it can be proved by induction that $M(n) \le 3n$. Of course we have ignored many fine details in the above analysis. Taking those into account may constitute a logn term in the upper bound for $M(n)$.

BIBLIOGRAPHY

[1] Blum, Floyd, Pratt, Rivest, and Tarjan
"Linear Time Bounds for Median Computations", Proceedings of fourth Annual ACM Symposium on Theory of Computing, May,1972

[2] Carroll, St. James's Gazette, August 1, 1883, pp. 5-6

[3] Kislitsyn, "On the Selection of the k-th Element of an Ordered Set by Pairwase Comparisons" Sibirskii Mat. Zhurnal 5, 1964

[4] Knuth, The Art of Computer Programming, vol. 3 Addison-Wesley

[5] Hadian and Sobel, "Selecting the t-th Largest Using Binary Errorless Comparisons" Technical Report 121, Department of Statistics, University of Minnesota May, 1969

[6] Pratt and Yao, "On Lower Bounds for Computing The i-th Largest Element", Proceedings of the Fourteenth Symposium on Switching and Automata Theory, 1973

[7] Schreier, Mathesis Polska 7 (1932), pp. 154-160

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Michael Fischer for his patient guidance throughout my graduate study. His help and encouragement is most important to my work. I also want to thank Professor Albert Meyer whose lectures are always a source of inspiration. I am also indebted to Professor Joel Spencer for many useful suggestions concerning this thesis.

I wish to express my gratitute to M.I.T. and Project MAC for the financial support I received during my graduate study.

Finally, I wish to thank my husband, Andrew, for his delightful companionship and warm understanding during the preparation of this thesis.

This empty page was substituted for a blank page in the original document.

CS-TR Scanning Project
Document Control Form

Date : 2/29/96

Report # LCS-TR-121

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 60 (65-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☒ Other: BIBLIOGRAPHIC DATA SHEET

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description : Page Number:
IMAGE MAP: (1-60) UNM'ED TITLE, BLANK, TABLE OF CONTENTS, BLANK PAGES 1-55, UNM'ED BLANK PAGE (61-65) SCANCONTROL, BIBLIO. DATA SHEET, TRGTS(3)

Scanning Agent Signoff:
Date Received: 2/29/96 Date Scanned: 3/14/96 Date Returned: 3/21/96

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 3/14/1996

M.I.T. Libraries
Document Services