Answers to Midterm 1.2
CIS 675 — Spring 1998

Problem 1. In LINEAR-SELECT-SORT, the total amount of work, in the worst case, is

$$(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}$$

because, to find the largest element of a sequence, we need $(n-1)$ comparisons, and likewise the next largest requires $(n-2)$ comparisons, etc.

(a) For convenience of presentation, let n be even. In this case, the largest element is found in $(n-1)$ comparisons, because $(\frac{n}{2}-1)$ comparisons are required to find the largest of the first sub-array, likewise $(\frac{n}{2}-1)$ comparisons to find the largest of the second sub-array, and 1 comparison to find the largest of the two. So, the total is $(\frac{n}{2}-1)+(\frac{n}{2}-1)+1=n-1$ comparisons.

In the worst case, the second and the third largest are found in $(\frac{n}{2}-2)+1=(\frac{n}{2}-1)$ comparisons only (find the next largest among the sub-array from which the largest was obtained, and compare with the other element). Similarly, the 4^{th} and 5^{th} are found in $(\frac{n}{2}-2)$ comparisons.

In general, the $(2i)^{th}$ and $(2i+1)^{th}$ largest will be obtained in $(\frac{n}{2}-i)$ comparisons.

Thus the total number of comparisons, in the worst case, would be

$$\begin{aligned}
T(n) &= (n-1)+2\sum_{i=1}^{\frac{n}{2}-1}(\frac{n}{2}-i)\\
&= (n-1)+2[1+2+\cdots+(\frac{n}{2}-1)]\\
&= (n-1)+2\frac{\frac{n}{2}(\frac{n}{2}-1)}{2}\\
&= (n-1)+\frac{n(n-2)}{4}\\
&= (n-1)+\frac{n^2}{4}-\frac{n}{2}\\
&= \frac{n^2}{4}+\frac{n}{2}-1.
\end{aligned}$$

Comparison with the LINEAR-SELECT-SORT shows that, for large n, there is a gain of $\frac{n^2}{4}$ terms.

(b) If we divide the initial array in 4 equal parts, then the performance analysis would follow a similar approach. For example, the largest element will be obtained in $(n-1)$ comparisons, thereafter next 4 largest elements will be obtained in $(\frac{n}{4}-2)+3=(\frac{n}{4}+1)$ comparisons and the next 4 largest elements in $(\frac{n}{4}-3)+3=(\frac{n}{4})$ comparisons, etc. Therefore the total amount of work would be

$$\begin{aligned}
T(n) &= (n-1)+4[(\frac{n}{4}+1)+(\frac{n}{4})+\cdots+(3)]+3 \\
&= (n-1)+4\frac{\frac{n}{4}(\frac{n}{4}+1)}{2}-12+3 \\
&= (n-1)+\frac{n(n+4)}{8}-9 \\
&= (n-1)+\frac{n^4}{8}+\frac{n}{2}-9 \\
&= \frac{n^2}{8}+3\frac{n}{2}-10.
\end{aligned}$$

where the last 3 in the first equation is to process the last three elements when the subsequences are of sizes 0, 1, 1, 1 respectively.

Problem 2. **(A)** The following algorithm works. For convenience of the presentation, assume $n = 4L$ for some integer L. If $n \neq 4L$ for any integer L then the last part of the list will contain fewer than 4 elements.

1. Compare X with $a[4], a[8], \cdots, a[4i], \ldots$ sequentially.
2. Suppose $a[4(i-1)] < X \leq a[4i]$. If $X = a[4i]$, we have found the correct index; Else compare X with $a[4i-1], a[4i-2], a[4i-3]$ successively.
3. If $X = a[4i-j]$, $j = 3, 2, 1$, then the correct index is known; Else, if $a[4i-j]X < a[4i-j-1]$ then X is not in the list A.

(B) The worst case is obtained when X is compared with $a[4], a[8], \cdots, a[4L]$ and then with $a[4L-1], a[4L-2], a[4L-3]$ before reaching a decision. Then the total number of comparisons would be $L+3 = (\frac{n}{4}+3)$.

(C) Suppose the search is successful. In a successful search, all $a[i]$'s are equally likely, $P(X = a[i]) = \frac{1}{n} = \frac{1}{4L}$.

If $X = a[4K]$, the number of comparisons is equal to K $K = 1, \cdots, L$. IF $X = a[4K-j]$, the number of comparisons is equal to $K+j$ where $j = 1, 2, 3$. Putting both of these equations together, we get, number of comparisons for $X = a[4K-j]$ is $K+j$ where $K = 1, \cdots, L$ and $j = 0, 1, 2, 3$. Then the average number of comparisons would be

$$\begin{aligned}
A(n) &= \frac{1}{n}\sum_{i=1}^{n}[\text{ number of comparisons if } X = a[i]] \\
&= \frac{1}{n}\sum_{K=1}^{L}\sum_{j=0}^{3}[\text{ number of comparisons if } X = a[4K-j]]
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n}\sum_{K=1}^{L}\sum_{j=0}^{3}(K+j) \\
&= \frac{1}{n}[\sum_{K=1}^{L}(4K+6)] \\
&= \frac{1}{4L}[4\frac{L(L+1)}{2}+6L] \\
&= \frac{L+1}{2}+\frac{3}{2} \\
&= \frac{n}{8}+2
\end{aligned}
$$

Suppose the search is unsuccessful. If $a[4K-j] < X < a[4K-j-1]$ where $K = 1, \cdots, L$ and $j = 1, 2, 3$ then, the number of comparisons required would be $K+j$. If $a[4(K-1)] < X < a[4K-3]$ then, the number of comparisons required would be $K+3$ (because by a previous comparison we know that $X > a[4(K-1)]$.)

Let $a[0] = -\infty$ and $a[n+1] = \infty$. Suppose all unsuccessful searches are equally likely, then the average number of comparisons is

$$
\begin{aligned}
&\frac{1}{n+1}\sum_{i=1}^{n}[\text{ number of comparisons if } a[1] < X < a[i+1]] \\
= &\frac{1}{n+1}\sum_{K=1}^{L}\sum_{j=0}^{3}[\text{ (number of comparisons if } a[4K-j] < X < a[4K-j-1])] + L
\end{aligned}
$$

where the last term L represents $X > a[n]$.

$$
\begin{aligned}
&= \frac{1}{n+1}\sum_{K=1}^{L}[(K+1)+(K+2)+(K+3)+(K+4)]+L \\
&= \frac{1}{n+1}\sum_{K=1}^{L}[(4K+10)]+L \\
&= \frac{1}{4L+1}[4\frac{L(L+1)}{2}+10L+L] \\
&= \frac{2L(L+1)}{4L+1}+\frac{11L}{4L+1} \\
&\approx \frac{2L(L+1)}{4L}+\frac{11L}{4L} \\
&\approx \frac{L}{2}+3 \\
&= \frac{n}{8}+3
\end{aligned}
$$

Then, the overall average $= \frac{1}{2}(\frac{n}{8}+2)+\frac{1}{2}(\frac{n}{8}+3) = \frac{n}{8}+\frac{5}{2}$.

Problem 3. This problem is solved by using the classical algorithm to find the k^{th} largest element. The algorithm we have discussed in class uses "partition" as the key step. The partition is repeated on the subsequences that contain the k^{th} largest element. Its analysis shows that the overall performance of the algorithm is $O(n)$. We adapt the algorithm as follows;

Using the first element as the pivotal element, partition the sequence of $k_1, k_2, \cdots, k_n$ into two subsequences such that

- all elements to the left of k_1 are smaller than k_1,
- all elements to the right of k_1 are larger than k_1,
- and k_1 is in its correct place.

Rename the sequence (only for the convenience of presentation, not in the algorithm) as $k_1^*, k_2^*, \cdots, k_n^*$ where $k_1 = k_j^*$ and their corresponding weights as $w_1^*, w_2^*, \cdots, w_n^*$. Find

$$p_1 = \sum_{i<j} w_i^*,$$

and

$$p_2 = \sum_{i>j} w_i^* = 1 - \sum_{i<j} w_i^* - w_j^*.$$

[In other words you do not have to add the weights in the right subsequence. (Preferably in the larger subsequence.)]

- **CASE 1.** If $p_1 < \frac{1}{2}$ and $p_2 < \frac{1}{2}$ then the weighted median is found, $k_j^*(= k_1)$.
- **CASE 2.** If $p_1 > \frac{1}{2}$ then the weighted median lies in the left subsequence.
- **CASE 3.** If $p_2 > \frac{1}{2}$ then the weighted median lies in the right subsequence.

[Both, CASE 2 and CASE 3 are essentially similar, so for clarity we consider CASE 2 only.]

In CASE 2: Apply the partition to the left subsequence, find the element of array k such that

- sum of the weights to the left of it is $< \frac{1}{2}$, and
- sum of the weights to the right of it is $< \frac{1}{2} - p_2 - w_j^*$.

The above procedure continues in a similar manner, after appropriately adjusting for the desired remaining weights. Details ignored.

Analysis: The average case behavior is linear, because the algorithm is essentially the usual "find" algorithm.