ECE608 CHAPTER 9 PROBLEMS

1) 9.1-1
2) 9.1-2
3) 9.3-1
4) 9.3-4
5) 9-2

ECE608 - Chapter 9 answers

(1) CLR 9.1-1

By comparing elements in tournament style as shown in Figure 1 (for an eight-element array $\langle 5, 4, 10, 8, 6, 3, 1, 9 \rangle$), we can find the minimum in $n - 1$ comparisons. Note that the second smallest element is among the elements that were compared with the minimum in the comparison tree. There are $\lceil \lg n \rceil$ elements that are compared with the smallest element (see the double circled elements in Figure 1). Now $\lceil \lg n \rceil - 1$ comparisons are required to find the smallest value out of these $\lceil \lg n \rceil$ values. Thus, the total number of comparisons is $n - 1 + \lceil \lg n \rceil - 1 = n + \lceil \lg n \rceil - 2$.

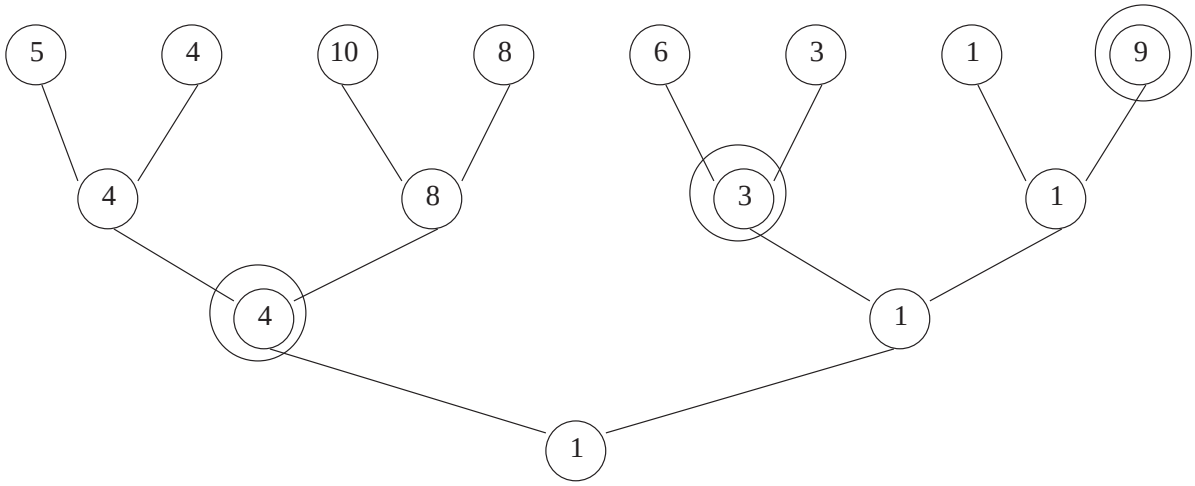

Figure 1: The algorithm for Problem CLR 9.1-1. The double circled elements were compared with the minimum and all are candidates for the second smallest element.

(2) CLR 9.1-2

When n is even we can form $n/2$ pairs. Compare the two numbers in each pair and place the winners in a LARGE bin and the losers in a SMALL bin. So far $n/2$ comparisons have been performed. The smallest number must lie in the SMALL array and the largest must be in the LARGE array. The size of both these new subarrays is $n/2$. Now we can run MINIMUM on SMALL to obtain the minimum number in $n/2-1$ comparisons and run MAXIMUM on LARGE to obtain the largest number in $n/2 - 1$ comparisons. Thus we have obtained the minimum and maximum numbers in $3n/2 - 2$ comparisons. As n is even $\lceil 3n/2 \rceil = 3n/2$.

When n is odd we can form $\lfloor n/2 \rfloor$ pairs with one element remaining unpaired. The $n-1$ numbers that are in pairs need $\lfloor n/2 \rfloor = \lceil n/2 \rceil - 1$ comparisons to be sorted into SMALL and LARGE bins. Each of these bins will have $\lfloor n/2 \rfloor$ numbers. Running MINIMUM on SMALL to obtain the minimum number requires $\lfloor n/2 \rfloor - 1 = \lceil n/2 \rceil - 2$ comparisons. Similarily $\lceil n/2 \rceil - 2$ comparions are required when MAXIMUM is run

on LARGE. So far $\lceil 3n/2 \rceil - 5$ comparions have been performed. Now we simply compare the one unpaired number against the maximum and minimum obtained so far (2 comparisons) to obtain the final maximum and minimum for the n numbers in a total of $\lceil 3n/2 \rceil - 3$ comparisons.

(3) CLR 9.3-1

(i) To analyze SELECT for groups of size 7, follow a similar analysis as that on p. 191, except the array now is divided into groups of 7; hence, the number of elements greater than x is at least $4(\lceil \frac{1}{2}\lceil \frac{n}{7} \rceil \rceil - 2) \geq \frac{2n}{7} - 8$. Hence, SELECT is called recursively on at most $\frac{5n}{7} + 8$ elements in step 5. The recurrence is:

$$T(n) \leq T(\lceil \frac{n}{7} \rceil) + T(\frac{5n}{7} + 8) + O(n)$$

We prove that $T(n) = O(n)$ using the substitution method with the assumption $T(k) \leq ck$ for $k < n$.

$$\begin{aligned} T(n) &\leq T(\lceil \tfrac{n}{7} \rceil) + T(\tfrac{5n}{7} + 8) + a\,n \\ &\leq c\lceil \tfrac{n}{7} \rceil + c(\tfrac{5n}{7} + 8) + a\,n \\ &\leq \frac{cn}{7} + c + \frac{5cn}{7} + 8c + a\,n \\ &\leq \frac{6cn}{7} + 9c + a\,n \\ &\leq cn - (\frac{cn}{7} - 9c - a\,n) \\ &\leq cn \end{aligned}$$

To make the last step valid, we must select c, a and n_0 so that $\frac{cn}{7} - 9c - a\,n \geq 0 \Rightarrow cn - 63c - 7\,a\,n \geq 0 \Rightarrow c(n-63) \geq 7\,a\,n$. When $n > 63$, it follows that $c \geq \frac{7\,a\,n}{n-63}$. If we pick $n \geq 126$, then $\frac{n}{n-63} \leq 2$; hence, picking $c \geq 14\,a$, it is clearly the case that $\frac{cn}{7} - 9c - a\,n \geq 0$. Therefore, the worst-case running time of SELECT on groups of size 7 is linear. In fact, any odd group size ≥ 5 works in linear time.

(ii) We can show that the running time for groups of 3 is not always linear by deriving a recurrence for a particular worst case that requires $\Theta(n \lg n)$ time. Consider a particular case in which n is divisible by six and each of the groups whose median is greater than the median of medians x contains exactly two elements greater than x. Hence, the number of elements greater than x is $2(\lceil \frac{1}{2}\lceil \frac{n}{3} \rceil \rceil - 1) + 1 = \frac{n}{3} - 1$ (note that one is subtracted in the first term to discount x's group, and one is added in at the end for the single

item contributed by x's group). Hence, the number of elements less than x is: $n - (\frac{n}{3} - 1) - 1 = \frac{2n}{3}$ elements, which is the partition chosen in this case, giving the following recurrence:

$$\begin{aligned} T(n) &= T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n) \\ &= \Theta(n \lg n) \qquad \triangleright \text{ by exercise 4.2-5} \end{aligned}$$

Hence, we have shown that there is at least one case for which the groups of 3 selection problem requires more than linear time.

(4) CLR 9.3-4

To have the i-1 smaller elemets without any more comparisons what we can do is to make two groups. Any time we make a comparision we put the larger element in one array and the smaller one in other. We do this for all comparisions and in the end one groups shall contain the i-1 smaller elements and the other will contain n-i larger elements.

(5) CLR 9.1-2

(a) If w_i is $1/n$ for all i, then the given expressions for x_k are:

$$\sum_{x_i < x_k} w_i < \frac{1}{2}$$

$$\sum_{x_i < x_k} \frac{1}{n} < \frac{1}{2}$$

and

$$\sum_{x_i > x_k} w_i \leq \frac{1}{2}$$

$$\sum_{x_i > x_k} \frac{1}{n} \leq \frac{1}{2}$$

The solution to x_k for the above inequalities are $k < \lceil n/2 \rceil$ and $k < \lfloor n/2 \rfloor$. And this corresponds to x_k being the median of the n numbers.

(b) Sort the n elements in $O(n \lg n)$ steps. Now for $i = i$ to n perform the summation of the w_i's and at each element compare this sum against $n/2$. Mark the first value of i for which the sum becomes greater than $n/2$ upon adding x_i. Thus we determine k that satisifes the two inequalities in $O(n)$ additions and comparisons at worst. The total running time being $O(n \log n)$.

(c) Design a routine WSELECT that takes an array of elements and a target weight w and returns the least element such that all the weights of the elements $\leq$ that particular element, have a total weight of w or more.

WSELECT operates as follows (following SELECT from the book):
1. Divide the input element into groups of 5 (with one group ≤ 5 elements)
2. Use insertion sort to sort each group and find its median
3. Use SELECT from the book to find the median of the medians x
4. Partition the input array around x
5. Compute the total weight t of the elements $\leq x$
6. Return x if $t - weight(x) < w \leq x$
7. Else, recursively invoke WSELECT on:
a. The elements $\leq x$ if $w < t$. Use the same target weight w
b. The elements $> x$ if $w > t$. USe target weight $w - t$

(d) We wish to minimize $\sum_{i=1}^{n} w_i|p - i|$. This sum is actually:

$$\sum_{i=1}^{p} w_i(p - i) + \sum_{i=p}^{n} w_i(i - p)$$

$$p\sum_{i=1}^{p} w_i - \sum_{i=1}^{p} w_i i + \sum_{i=p}^{n} w_i i - p\sum_{i=p}^{n} w_i$$

We can make the first and the fourth term cancel out each other if we choose p such that it divides the sum of w_i into two equal halves. The second and the third term are the sum of moments of each w_i about the centre p on either side. The total sum of moments is miminized by setting the centre p to the centre of gravity of the wieghts, which is exactly the weighted median of the numbers.