HW - Week 10

Name: Sahand Sabour **Student ID:** 2020280401

Page 22:

2.1-3) the pseudo-code for this linear search would be as follows

LINEAR-SEARCH(A,v)

For i = 1 to A.length

    If A[i] == v

        Return i

Return NIL

At the beginning of each iteration, since the loop is still going, that means that element v has not been found in the array so far. Therefore, the loop invariant would be that at the start of each iteration, the sub-array A' = A[1..i-1] does not contain v.

Initialization:

v does not exists in the initial empty sub-array.

Maintenance:

At the start of each iteration, A[1..i-1] does not contain v, else we would have returned i; since the algorithm returns i when A[i]=v.

Termination:

There are two cases that cause the termination: first, if we observe v in A, we would return i and the loop is terminated; second, if we don't observe v in A and i becomes larger than the length of A, we return NIL and the loop is terminated.

Hence, the loop invariant fulfills the three necessary properties.

Page 39:

2.1) The answers for each part are provided respectively below:

(a) $T(n) = \Theta(k^2)*n/k = \Theta(k^2*n/k)= \Theta(nk)$

(b) The merging process is done for two sub-lists at a time, giving the worst case time for finishing all the merges for n/k sub-lists to $\Theta(\lg(n/k))$. Accordingly, the worst case time within each merge process, that is the worst-time to merge two sub-list is

Θ(n), giving the worst case time for the overall merge to be Θ(nlg(n/k)).

(c) As provided, merge sort runs in Θ(nlgn). Therefore, for Θ(nk+nlg(n/k)) to be equal to Θ(nlgn), we have that

$$\mathrm{nlgn} = \mathrm{nk} + \mathrm{nlg}\frac{\mathrm{n}}{\mathrm{k}} = \mathrm{nk} + \mathrm{nlgn} - \mathrm{nlgk}$$

Assuming that $\mathrm{k} > \mathrm{lgn}$, we would have Θ(nk+nlg(n/k))>Θ(nlgn) since k is growing faster than lgn. Therefore, lgn would be the upper bound for the value of k. Thus, we can set k=lgn, giving

$$\Theta(\mathrm{nk} + \mathrm{nlgn} - \mathrm{nlgk}) = \Theta(\mathrm{nlgn} + \mathrm{nlgn} - \mathrm{nlglgn}) = \Theta(2\mathrm{nlgn} - \mathrm{nlglgn}) = \Theta(\mathrm{nlgn})$$

(d) As stated, the running time for merge sort and insertion sort are respectively Θ(nlgn) and Θ(n^2). Therefore, we should find the largest k for which insertion sort runs faster than the merge sort. In practice, it is believed that this value would be obtained by experiment.

Page 223:

9.3-1) If we divide the n elements into 7 groups, we would find the median for each group, which would be index 4 of the sorted sub-group, and then recursively use the medians as the pivot. Accordingly, the input array around the selected median x is divided into two zones, namely S1 and S2, where the size is at least

$$4([\frac{1}{2}[\frac{\mathrm{n}}{7}]] - 2) \geq \frac{4\mathrm{n}}{14} - 8 = \frac{2\mathrm{n}}{7} - 8$$

Hence, the size of the sub-problem would be at most n-($\frac{2\mathrm{n}}{7} - 8) = \frac{5\mathrm{n}}{7} + 8$. Accordingly, we can obtain the following recurrence

$$\mathrm{T(n)} \leq \begin{cases} \mathrm{O}(1) & \text{if } \mathrm{n} < \mathrm{n}_0 \\ \mathrm{T}([\frac{\mathrm{n}}{7}]) + \mathrm{T}(\frac{5\mathrm{n}}{7} + 8) + \mathrm{O(n)} & \text{if } \mathrm{n} \geq \mathrm{n}_0 \end{cases}$$

Therefore, we can show that the running time is linear by substitution; that is, T(n)≤cn for a positive c given that $n \geq n_0$.

$$\mathrm{T(n)} \leq \mathrm{c}[\frac{\mathrm{n}}{7}] + \mathrm{c}\frac{5\mathrm{n}}{7} + 8\mathrm{c} + \mathrm{an}$$

$$\leq \mathrm{c}[\frac{\mathrm{n}}{7}] + \mathrm{c} + \mathrm{c}\frac{5\mathrm{n}}{7} + 8\mathrm{c} + \mathrm{an}$$

If we divide the n elements into 3 groups, we would find the median for each group, which would be index 2 of the sorted sub-group, and then recursively use the medians as the pivot. Accordingly, the input array around the selected median x is divided into two zones, namely S1 and S2, where the size is at least

$$2([\frac{1}{2}[\frac{n}{3}]] - 2) \geq \frac{2n}{6} - 4 = \frac{n}{3} - 4$$

Hence, the size of the sub-problem would be at most $n - (\frac{n}{3} - 4) = \frac{2n}{3} + 4$. Accordingly, we can obtain the following recurrence

$$T(n) \leq \begin{cases} O(1) & \text{if } n < n_0 \\ T([\frac{n}{3}] + T(\frac{2n}{3} + 4) + O(n) & \text{if } n \geq n_0 \end{cases}$$

Therefore, we try to show that the running time is linear by substitution; that is, $T(n) \leq cn$ for a positive c given that $n \geq n_0$.

$$T(n) \leq c[\frac{n}{3}] + c\frac{2n}{3} + 4c + an$$

$$\leq c[\frac{n}{3}] + c + c\frac{2n}{3} + 4c + an$$

$$= cn + 5c + an$$

Which is larger than cn. Hence, no positive value of c would fulfill the condition and therefore, the algorithm does not run in linear time if we use groups of 3.

9.3-7) Find the median in linear time O(n), find the distance for each value from median in linear time O(n), find k^{th} smallest number using SELECT in linear time O(n), and choose values whose distance to the median is less than or equal to k^{th} smallest number in linear time O(n).