CS 170 Homework 2

Due **2025/2/23, at 10:00 am (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write “none”.

Solution: I worked on this homework with the following collaborators:

- none,which is only me,Sillycheese

2 Median of Medians

(a) (2 points) Let us see an example of QuickSelect in action. Suppose you always pick the first element as the pivot. Compute QuickSelect(A, 6) for the following array:

Solution: [78,13,97,45,48,26,85,100,78] k=4

[13,45,48,26] k=4

[45,48,26] k=3

[48] k=1

(b) (2 points) Consider the array

shuffled into some arbitrary order. What is the worst-case runtime of QuickSelect(A, n/2) in terms of n? Construct a sequence of ‘bad’ pivot choices that achieves this worst-case runtime.

Solution:

$$\frac{n(n+1)}{2} = O(n^2)$$

(c) (3 points) Let p be the pivot chosen by DeterministicSelect on A. Show that at least 3n/10 elements in A are less than or equal to p, and that at least 3n/10 elements are greater than or equal p.

Solution: firstly,at least half of medians array is less than p.so it is $n/10$. then go back to every median’s subarray and add them.It is $\frac{2n+n}{10}$(also need to add itself). so at least $\frac{3n}{10}$ elements in A are Less than or equal to p.

Same to the greater issue.

(d) In this problem, we will show that the worst-case runtime of DeterministicSelect(A, k) using the ‘Median of Medians’ strategy is O(n).

i. Find a recurrence relation for the time complexity of the algorithm, T (n).

Solution:

$$T(n) <= T(n/5) + T(7n/10) + O(n)$$

$T(n/5)$ is to find the medians in each subarray.

$T(7n/10)$ is the at most size of partition size,which is used to recursive call to the DC

$O(n)$ is just build array and partition time.

ii. Use the recurrence relation to show that, for some sufficently large c 0, the inequality T (n) c · n always holds.

Solution:

$$T(n) \leq c(n/5) + c(7n/10) + O(n) \leq c$$

(e) We are playing a variant of The Resistance, a board game where there are n players, s of which are spies. In this variant, in every round, we choose a subset of players to go on a mission. A mission succeeds if the subset of the players does not contain a spy, but fails if at least one spy goes on the mission. After a mission completes, we only know its outcome and not which of the players on the mission were spies.

(a) (2 points) If there is one spy (s = 1), come up with a strategy that identifies the spy in O(log(n)) missions. You do not need to prove that your strategy works.

Solution: partition n/2 each time.until we found that spy. it looks like a BS algorithm.

(b) (2 points) In the general case, when there are s spies, consider evenly splitting the n players into x disjoint groups (containing n/x players each), and send each group on a mission. At least how many of these x missions must succeed, in terms of x and s?

Solution: x-s

(c) (8 points) Come up with a strategy that identifies all the spies in O(s log(n/s)) missions.

Solution: we can partition 2s groups.at least s groups will succeed.then we can recursively deal with those which failed. so the $O(n) = s\log(n/s)$.

we need to find s groups each time,and we also need to deal with recursive tree,which is $O(log(n/s))$

This content is protected and may not be shared, uploaded, or distributed.