Assignment 5

Erik Norlin, CID: norliner

September 29, 2023

Problem 9

a)

A test strategy for this problem with a minimal worst case number of tests is to use a divide and conquer approach similar to binary search. Split the set S in half and test one subset, repeat this procedure for every subset that turns out to be positive. Continue doing this until the special object has been found. This can be described using the recursive formula

$$T(n) = T(n/2) + 1$$

where $T(n)$ is the total number of tests, n the total number of objects, $n/2$ means that the subset is splitted in half, and the constant "1" means that one test is performed. If the subset that is tested shows to be positive the object is in that subset, otherwise the object is in the other subset. Hence, only one test is needed to be performed for every split. Generally, since the number of subsets increases exponentially in the power of two for each split, the number of times we can split n into two subsets until each subset contains only one item is therefore $log_2(n)$. This number of splits is also true in this case when subsets that turns out to be negative are neglected, and hence results in this number of needed tests.

b)

The algorithm given in (a) is optimal because it runs in $O(log(n))$ which is optimal for comparison based search for large enough n, in a sorted array that is. However, since the algorithm does not compare a multitude of different values, only binary representations of positive and negative tests, a sorted array does not need to be considered in this case for it to be optimal, the algorithm runs in $O(log(n))$ either way. Alternatively, it is possible to test every object but this would result in a running time in $O(n)$, more expensive than the suggested algorithm, because the object we are looking for could potentially be the last object we test, resulting in n tests.

c)

The claimed strategy is optimal because if the subsets contain $\tilde{j}$ objects in each of them and $\tilde{j} < k$ we potentially end up performing more tests until we start searching as in (a). This means that we would perform more tests in $O(n)$ than if $\tilde{j} = k$. We want to start searching as in (a) as soon as possible because this algorithm runs in $O(log(n))$ which is faster than a linear running time (for large enough n). This becomes more obvious as the size of the subsets we choose becomes significantly smaller than k. Say that $\tilde{j} = 1$, the algorithm would never begin to search as in (a) and hence completely run in $O(n)$. Whereas if $k = n$ the algorithm would always run in $O(log(n))$. It is therefore better to maximize the number of objects in each subset,

i.e., let $\tilde{j} = k$, until there are less than $2k$ objects left and start searching as in (a) from there, which the claimed formula implies.

d)

If we have more objects to test then it comes naturally that we might have to perform more tests in total. Say that $k = 4$, $i = 4$, and $j = 8$, then $T_k(i)$ becomes 2 tests, whereas $T_k(j)$ becomes 3 tests. On the other hand if $i = j$ it is obvious that they have the same worst case number of tests. Thus, $T_k(i) \leq T_k(j)$.

e)

We prove optimality of the the proposed algorithm by contradiction with the help of the results from (c). Assume there is one optimal solution such that each subset contains maximum $\tilde{j}$ objects and that $\tilde{j} \ll k$. Performing the suggested recursive formula, subsets with the size of $\tilde{j}$ are tested until there are less than $2\tilde{j}$ objects left, the algorithm then starts to search as in (a). This is the optimal solution because it minimizes the amount of tests to perform in $O(n)$ before starting searching in $O(log(n))$. This is a contradiction because $\tilde{j} \ll k$ so the amount of tests to perform in $O(n)$ is larger than if $\tilde{j} = k$.