Ryan Pollack

rep127

I have not discussed this exam with anyone except the professor and the TAs of CS 344. I have not used any online resources during the exam except for those accessible from the Canvass website.

1. Part 1



Layer Number of calls Problem size Non-recursive work

1 1 n

2 64 n/4

3 n/

… … … …

k n/

The non-recursive work on the kth level is

geometric series

1. Part 2

Considering,

Assume where some c>0

Base case: c = 1, so 1

So,

2.

I don’t know

3.

The T(n/13) part is derived from the fact that when we run the recursive SELECT method, we are running it in the 13 different chunks which fills the new array of size n/13 that holds the mini median from each of the 13 chunks. For each of these individual chunks, picking out the mini median is O(1) time and there is n/13 groups so T(n/13).

The O(n) is derived from partitioning around the chosen m. It goes one by one and compares each chunk median with the m and positions said element on the left side (lesser than m) or right side (greater than m) of m based on the comparison. (Dividing the array into 13 chunks and creating the secondary array of length n/13 also take O(n) time.)

The T(19n/26) is derived based from the worst-case of how far away from the center m can be. To prove this, we can start by saying by definition, half of the mini medians will be lesser than m (the medians of the medians) and half will be larger than m, so (n/13)/2 = n/26 for each lesser half (as well as larger half but we’re proving this with small). Looking into each lesser half, we know the median of that chunk is smaller than m. However, we also know since it’s the median of the chunk, at least half of the elements in there will also be smaller than m since half are smaller than the mini medium. In the lesser chunks with 13, that is the median + 6 other elements are bare minimum, so 7 \* (n/26) = 7n/26. With this same concept applied on the larger side, we know m can at most be 19n/26, where at least 7 elements are bigger in each larger chunk.

4.

Set closest = int\_Max

Set answer = 0

Initialize

For i=0 to n-1

.insert(B[i].value, B[i])

End for

For j=0 to n=1

Set diff = abs\_value(A[j] - .search(A[j].value))

If diff < closest

closest = diff

answer = A[j].value

End if

End for

Return answer

5.

ThresholdSum(A, T)

Set the ints value, largest, and count to 0

For i=0 to n-1 //deals with no solution

Value += A[i]

End for

If value T

Return “no solution”

End if

ThresholdSumRecursive(A, T)

For i=0 to n-1

If largest < A[i]

Set largest = A[i]

End if

End for

count++

result += largest

Remove A[i] that ended as largest from array

If(result > T)

Return count

Else

ThresholdSumRecursive(A, T)

End ifelse