

CSE 150: Problem Set #3

Problem 1

Construct an array of 7 integers such that, when quicksort runs on the array, every call to partition splits its input array exactly in half.

Now suppose you have two arrays, A and B , each of length n . Suppose also that, when quicksort runs on A , each call to partition splits its input array exactly in half, and likewise for B . Construct a new array C of size $2n + 1$ such that, when quicksort runs on C , each call to partition splits its input array in half.

Solution

Any array $[x_3, x_0, x_2, x_1, x_5, x_4, x_6]$ where all $x_i \in \mathbb{Z}$ and $x_i < x_{i+1}$.

Let $C[0]$ be $1 + \max A$. Now take A , rotate it left one item ($A = A[1 \dots n]$ with $A[0]$ appended after $A[n]$), and append that to C . Finally, add $C[0] - \min B$ to every element of B , and append that to C .

Grading

- 6 points for the array of seven integers
- 10 points for the constructed array:
 - 2 points for correctly identifying what $C[0]$ should be
 - 4 points for altering A and using it in C in the correct place
 - 4 points for altering B and using it in C in the correct place

Problem 2

Solve the following recurrence relations (in all cases, $T(1) = 0$). You only need to find $\Theta()$ solutions, not exact solutions.

- $T(n) = T(\lfloor n/2 \rfloor) + 1$
- $T(n) = 2T(\lfloor n/2 \rfloor) + 1$
- $T(n) = 4T(\lfloor n/2 \rfloor) + 1$
- $T(n) = T(n - 1) + 1$
- $T(n) = 2T(n - 1) + 1$
- $T(n) = T(\lfloor n/2 \rfloor) + n$
- $T(n) = 2T(\lfloor n/2 \rfloor) + n$ (yeah, I know we did one like this in class)
- $T(n) = 4T(\lfloor n/2 \rfloor) + n$

Solution

- $T(n) = T(\lfloor n/2 \rfloor) + 1 = \Theta(\log n)$
- $T(n) = 2T(\lfloor n/2 \rfloor) + 1 = \Theta(n)$
- $T(n) = 4T(\lfloor n/2 \rfloor) + 1 = \Theta(n^2)$
- $T(n) = T(n-1) + 1 = \Theta(n)$
- $T(n) = 2T(n-1) + 1 = \Theta(2^n)$
- $T(n) = T(\lfloor n/2 \rfloor) + n = \Theta(n)$
- $T(n) = 2T(\lfloor n/2 \rfloor) + n = \Theta(n \log n)$
- $T(n) = 4T(\lfloor n/2 \rfloor) + n = \Theta(n^2)$

Grading

- 3 points per problem. Partial credit sometimes given if you had some good-looking scratch work.

Problem 3

Pick three of the recurrences from Problem 2 and carefully prove, via induction, that your answer is correct. For convenience, you may assume that n is a power of two.

Solution

I'm not typing up 8 induction proofs, are you crazy? Come to office hours if you want to practice induction.

Grading

- 10 points per problem:
 - 2 points for a correct base case
 - 6 points for a correct induction hypothesis
 - 2 points for style

Problem 4

Use the partition algorithm to write a function **find-kth-smallest**(**A**, **n**, **k**) that, given an array, **A**, of **n** integers, returns the **k**th smallest integer in the array. In other words, your algorithm should return the integer that would be in position **k** after the array is sorted. A trivial solution to this problem is

```
procedure find-kth-smallest(A, n, k)  
  qsort(A, n)  
  return A[k]
```

This algorithm would have running time $O(n \log n)$. Your algorithm should be faster.

Solution

```
procedure find-kth-smallest( $A, n, k$ )
   $t :=$  partition( $A[1 \dots n - 1], n - 1, A[0]$ )
  if ( $k = t - 1$ )
    return  $A[0]$ 
  else
    if ( $k < t - 1$ )
      return find-kth-smallest( $A[1 \dots t - 1], t - 1, k$ )
    else
      return find-kth-smallest( $A[t \dots n - 1], n - t, k - t$ )
```

This algorithm does $O(n)$ work to divide the input in half (expected), and then runs the algorithm on one of those halves. Therefore, the recurrence relation is

$$T(n) = n + T\left(\frac{n}{2}\right)$$

which, when solved, is $O(n)$.

Grading

- 15 points for a correct algorithm (no points for copying the algorithm that was given)
- 10 points if that algorithm was $O(n)$
- 5 points for clarity / an explanation / a running-time analysis (NB: Few of you provided an analysis. How are we to know your algorithm is faster than the one given if you don't explain how it is?)

Example tabbing environment

In case you decide to write your homework in latex, here's an example of using the tabbing environment to typeset pseudo-code using the commands defined in the source of this latex file.

```
procedure find-kth-smallest( $A, n, k$ )
  qsort( $A, n$ )
  if foo
    hello
    for blah blah blah
      hello
  else
    goodbye
  for blah blah blah
    hello
  return  $A[k]$ 
```