Evan La Fleur
Partners: Ian Guy(m/w),
Michael Robertson(t/th)

Homework 3

CS 350
October 23, 2021

Section 5.1

Problem 1

a. Write pseudo-code for a divide and conquer algorithm for finding the position of the largest element in an array of n numbers.
b. What will be your algorithm's output for arrays with several elements of Write pseudo-code for a divide-and-conquer algorithm for finding the position of the largest element in an array of n numbers.the largest value?
c. Set up and solve a recurrence relation for the number of key comparisons made by your algorithm.
d. How does this algorithm compare with the brute-force algorithm for this problem?

a.

```
int curr = 0;
for (int i = 1; i < array.len; ++i)
{
  if (array[i] > array[curr])
    curr = i;
}
return curr;
```

b. If there are several elements of the largest value, then it will return just 1 of them.

c. .
$C(n) = C(n/2) + C(n/2) + 1$ for $n > 1, c(1) = 0$
$C(2^k) = 2C(2^{k-1}) + 1$
$2[2C(2^{k-2}) + 1] + 1 = 2^2C(2^{k-2} + 2 + 1$
$2^2[2C(2^{k-3}) + 1] + 2 + 1 = 2^3C(2^{k-3}) + 2^2 + 2 + 1$
$2^kC(2^{2k-k}) + 2^k - 1... + 1 = 2^k - 1 = n - 1$

d. It is the same as the brute force version listed above, as it requires the same amount of comparisons.

Problem 5

Find the order of growth for solutions of the following recurrences.
$a.T(n) = 4T(n/2) + n,$ *T(1)=1*

b.$T(n) = 4T(n/2) + n^2$, *T(1)=1*
c.$T(n) = 4T(n/2) + n^3$, *T(1)=1*

a. a = 4, b = 2, d = 1 which leads to $\theta(n^{log_b a}) = \theta(n^2)$ according to the master theorem.

b. a = 4, b = 2, d = 2. This would be $a = b^d$, which leads to $\theta(n^d logn) = \theta(n^2 logn)$ according to the master theorem.

c. a = 4, b = 2, d = 3. This would be $a < b^d$, which leads to $\theta(n^d) = \theta(n^3)$ according to the master theorem.

Section 5.2

Problem 5

For the version of quicksort given in this section:
A. Are arrays made up of all equal elements the worst-case input, the best- case input, or neither?
B. Are strictly decreasing arrays the worst-case input, the best-case input, or neither?

a. If it is assumed that A[s] is in the middle of the array, then it would be neither worst or best case because then it would always be the L array would be 0 (L) < s-1 indices, and R array would be s+1 < R. If A[s] is not in assumed to be in the middle, then it would be the best case input if the arrays were made up of all equal elements.

b. If this was a strictly decreasing array, then this would be a worst case input because there would be a lot of comparisons and switching that would lead to the array being properly sorted.

Problem 10

Implement quicksort in the language of your choice. Run your program on a sample of inputs to verify the theoretical assertions about the algorithm's efficiency.
See attached documents for this question

Section 7.1

Problem 1

Is it possible to exchange numeric values of two variables,say,u and v, without using any extra storage?
In python, there is a way to do this. You could then put $u, v = v, u$ and then the two variables would exchange the numeric value that they hold. This was the reason Python was chosen for problem 10 above. By being able to easily swap variables we are able to generate a more efficient piece of code.

Problem 4

Is the distribution-counting algorithm stable?
It seems so. It should take the larger values and then put them at the end, then fill in the values from the right to the left.