CIS 477 (Fall 2022) Disclosure Sheet

Name: Kevin Lopez

HW # 3

Yes No Did you consult with anyone on parts of this assignment, including other students, TAs, or the instructor?

Yes No Did you consult an outside source (such as an Internet forum or a book other than the course textbook) on parts of this assignment?

If you answered Yes to one or more questions, please give the details here:

By submitting this sheet through my Blackboard account, I assert that the information on this sheet is true.

This disclosure sheet was based on one originally designed by Profs. Royer and Older.

Kevin Lopez

CIS477

10/3/22

HW2

Problem 1: Prove that $n! = O(n^n)$. (Reminder, $n!$ is defined as $n \times (n-1) \times ... \times 1$).

1)

a) Claim: n! = O(n^n)

b) Direct Proof:

   i) For n=0, n! = 0

   ii) For n=1, n!=0

   iii) For n, n!= n*(n-1)*….*1

   iv) For n+1, (n+1)*n*(n-1)…*1

   v) This can be rewritten as (n+1)*n!

   vi) This can be rewritten as (n^2+n) *(n-1)!

   vii) This can be rewritten as (n^3-n)

   viii) We can see that the n is increasing as well as the exponent raised to it, so it is (n^{n-1}+c)

   ix) After this, We see the recurrence relation is showing O(n^n)

2)

Problem 2: Analyze the running time of the following algorithm (assume arrays are 0-indexed):

```
function FINDMIN(array X)
    if length(X) == 1:
        return X[0]
    else:
        X1 ← X[0 : length(X)/2]
        X2 ← X[length(X)/2 : length(X)]
        return min(FindMin(X1), FindMin(X2))

end function
```

a) T(n) = running time of FindMin for array size n

b) T(1) = C1

c) T(n) = C2 + T(n/2) + T(n/2) + O(1)

d) T(n) = 2T(n/2) + O(1)

e) A = 2, b = 2, d = 0

f) $\text{Log}_2 2 < 0$

g) O(n)

Problem 3: Suppose you are given the following list: $[6, 8, 7, 1, 2]$. Describe each step of the following sorting algorithms on this list: MergeSort, InsertionSort, BubbleSort, and QuickSort (use the first element as the pivot). For this problem, it is sufficient to show the array after each step of the sorting algorithm.

3)

a) Merge Sort

   i) Array1 = [6,8] , Array2 = [7,1,2] , mergedArray = []

ii) Array1 = [6,8] , Array2 = [7,2] , mergedArray = [1]
iii) Array1 = [6,8] , Array2 = [7] , mergedArray = [1,2]
iv) Array1 = [8] , Array2 = [7] , mergedArray = [1,2,6]
v) Array1 = [8] , Array2 = [] , mergedArray = [1,2,6,7]
vi) Array1 = [] , Array2 = [] , mergedArray = [1,2,6,7,8]

b) InsertionSort
  i) Array = [6,8,7,1,2]
  ii) Array = [6,7,8,1,2]
  iii) Array = [1,6,7,8,2]
  iv) Array = [1,2,6,7,8]

c) BubbleSort
  i) Array = [6,8,7,1,2]
  ii) Array = [6,7,8,1,2]
  iii) Array = [6,7,1,8,2]
  iv) Array = [6,7,1,2,8]
  v) Array = [6,7,1,2,8]
  vi) Array = [6,1,7,2,8]
  vii) Array = [6,1,2,7,8]
  viii) Array = [6,1,2,7,8]
  ix) Array = [1,6,2,7,8]
  x) Array = [1,2,6,7,8]

d) Quicksort
  i) Array = [6,8,7,1,2]
  ii) Array = [1,2,6,8,7]
  iii) Array = [1,2,6,7,8]

Problem 4: Write and solve a recurrence relation describing the worst-case running time of QuickSort as a function of the array size n. Assume that the first element is used as the pivot. Explain why your recurrence relation

4) is correct.

a) T(n) = worst case scenario for Quicksort for array size n
b) T(1) = C1
c) T(n) = T(n-1) + T(n-2)+…+T(n)
d) T(n) = (n-1) + (n-2) + … + n
e) T(n) = ((n-1)n)/2
f) T(n) = O(n^2)

Extra Credit: A binary tree is *full* if all vertices have either 0 or two children. Let B_n denote the number of full binary trees with n vertices.

1. What are the values of B_3, B_5, and B_7?
2. For general n, derive a recurrence relation for B_n. Include base cases. Hint: if you have a full binary tree with n vertices, what are the possible sizes of the left and right subtrees?

5)

a) The values of these are
   i) B_3=6
   ii) B_5=15
   iii) B_7=28
b) B(n) = running time of B for binary tree size n

c) $B(1) = C1$
d) $B(n) = 2B(n) + n$