Kevin Lopez

CIS477

10/17/22

HW3

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.

1) The algorithm to search through this circular array is to take the first 2 elements and add them and then take the next two and add them. If the first list total is greater than the second list total then you have found the end of the circular array and enter the first list and find the max of the two elements which is on the left. It is done in O(Log(n))

a) First two elements get added up and stored into variable: firstList
b) Second two elements get added up and stored into second variable: secondList
c) If (firstList > secondList):
    i) First element of firstList is max

2) O(n) where n is the total length of both lists because in order to merge them using the merge function, they have to iterate through each first element of each list and compare and then add one of them to the list.
    a) T(n) = running time of merge
    b) T(n) = 1 when lists are empty or hold only one element
    c) T(n) = a + b + … z, where each variable holds the length of each list
    d) T(n) = nm, where n is number of lists and m is length of lists
    e) T(n) = O(n)
3) To do this algorithm, you can start a binary search algorithm to check if A[i]/2 = i, then if it A[i] < i, you call binary search on the left side of A[i/2] only and the opposite side for A[i] > i. The running time will be O(logn) because it is splitting up the tree in half each time and searching based on that.
    a) If A[i/2] == i
        i) Return i
    b) If A[i/2] < i:
        i) Run A[i/2] on left half where original i/2 is the end of the list
    c) Else:
        i) Run A[i/2] on right half where original i/2 is at the beginning of the list
4) This will create a hashtable with [0:9 ; 1: N/a , 2:2 ; 3: 3,12 ; 4:4 ; 5: N/A, 6: 6,15, 7: 7,15 ; 8: N/A]
    a) For 0:
        i) 9 mod 9 = 0
    b) For 1: None
    c) For 2:
        i) 2 mod 9 = 2
    d) For 3:
        i) 3 mod 9 = 3
        ii) 12 mod 3 = 3
    e) For 4:
        i) 4 mod 9 = 4
    f) For 5 = none
    g) For 6:
        i) 6 mod 9 = 6
        ii) 15 mod 9 = 6
    h) For 7:
        i) 7 mod 9 = 7
        ii) 15 mod 9 = 7

    i) For 8:
        i) 8 mod 9 = 8
5) The total memory required is O(n+nlogn) where n is the length of the list
    a) N is the original array and takes up space on the stack
    b) The first time it calls mergesort it splits up the array into 2^n which can be written as log(n)
6) You merge it based on how many arrays there are and their lengths
    a) O(klogn) because it does k iterations of merging arrays and merges them in log n time because it splits the array in log time complexity
    b) To do this you call merge at the end but extend the if statement to include all k arrays instead of merging them separately.