COMP 3270

Homework 3

100 points

**Please submit using Canvas by 11:59PM on Thursday, July 14th, 2022**

Instructions:

1. This is an individual assignment. You should do your own work. Any evidence of copying will result in a zero grade and additional penalties/actions.
2. Late submissions **will not** be accepted unless prior permission has been granted or there is a valid and verifiable excuse.
3. Think carefully; formulate your answers, and then write them out concisely using English, logic, mathematics and pseudocode (no programming language syntax).
4. Type your final answers in this Word document.
5. Don’t turn in handwritten answers with scribbling, cross-outs, erasures, etc. If an answer is unreadable, it will earn zero points. **Neatly and cleanly handwritten submissions are acceptable**.

**1. (5 points)** Heapsort

Show the array A after the algorithm Min-Heap-Insert(A, 6) operates on the Min Heap implemented in array A=[6, 8, 9, 10, 12, 16, 15, 13, 14, 19, 18, 17]. In order to solve this problem you have to do some of the thinking assignment on the Ch.6 lecture slides. But you do not have to submit your solutions to those thinking assignments. Use your solutions to determine the answer to this question and provide the array A below.

**A= [6,8,6,10,12,9,15,13,14,19,18,17,16]**

**2. (5 points)** Let A be a collection of objects. Describe an efficient O(nlgn) algorithm for converting A into a set. That is, remove all duplicates from A.

**We can use a heap sort algorithm to first sort the collection of objects, which would be of nlgn time complexity. Then we can search through the sorted list to find all duplicates, which would be of n time complexity. This can be done by seeing if the current index is equal to the previous index in each iteration. If it is equal, it is removed and the index increases. This would require nlgn + n calls, which would be approximately O(nlgn).**

**3****. (5 points)** Given a sequence of numbers, S, the mode is the value that appears the most number of times in this sequence. Give an efficient O(nlgn) algorithm to compute the mode for a sequence of n numbers.

**We can use a heap sort algorithm to first sort the collection of objects, which would be of nlgn time complexity. Then we can search through the sorted list and count all the same integers that appear consecutively, which would be of n time complexity. This can be done by increasing the index and determining if that value in the array was equal to the previous. If not, we can create a new variable for the new number that appears in the array. This would require nlgn + n calls, which would be approximately O(nlgn).**

**4. (5 points)** Show that any comparison-based sorting algorithm can be made to be stable, without affecting the asymptotic running time of this algorithm. Hint: Change the way elements are compared with each other.

**You can make any comparison-based sorting algorithm stable by changing the comparison operation so that the comparison between two values also considers position as a factor for equal keys. For example, there would be a separate comparison operation for 1 and 1 where the position will be considered, but not 3 and 4.**

**5. (22 points)** Quicksort

(a) (6 points)

Quicksort can be modified to obtain an elegant and efficient linear (O(n)) algorithm QuickSelect for the selection problem.

Quickselect(A, p, r, k)

{p & r – starting and ending indexes; to find k-th smallest number in non-empty array A; 1≤k≤(r-p+1)}

1 if p=r then return A[p]

else

2 q=Partition(A,p,r) {Partition is the algorithm discussed in class}

3 pivotDistance=q-p+1

4 if k=pivotDistance then

5 return A[q]

6 else if k<pivotDistance then

7 return Quickselect(A,p,q─1,k)

else

8 return Quickselect(A,q+1,r, k-pivotDistance)

Draw the recursion tree of this algorithm for inputs A=[10, 3, 9, 4, 8, 5, 7, 6], p=1, r=8, k=2. At each non-base case node show all of the following: (1) values of all parameters: input array A, p, r & k; (2) A after Partition. At each base case node show values of all parameters: input array A, p, r & k. Beside each downward arrow connecting a parent execution to a child recursive execution, show the value returned upwards by the child execution.

Text, letter

Description automatically generated

(b) (16 points). This algorithm has two base cases.

Explain what the first base case that the algorithm checks for is, in plain English:

**The first base case checks if the algorithm’s first and last element are the same, which can only be possible if the array contains one element. If this is the case, then the single element is returned.**

List the steps that the algorithm will execute if the input happens to be this base case: **1**

Complete the recurrence relation using actual constants:

T(first base case) = **\_\_\_\_\_\_\_\_\_\_\_(1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Explain what the second base case that the algorithm checks for is, in plain English:

**The second base case checks if the k, the number element smallest in the list, is equal to the pivot distance point. If this is the case, then the element at index q is returned.**

List the steps that the algorithm will execute if the input happens to be this base case**: 1, 2, 3, 4, 5**

Complete the recurrence relation using actual constants (assume complexity of Partition to be 20n):

T(second base case) = **\_\_\_\_\_\_\_\_\_\_\_20n+20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

List the steps that the algorithm will execute if the input is not a base case: **1, 2, 3, 4, 5, 6, (7 or 8)**

Complete the recurrence relation using actual constants (assume complexity of Partition to be 20n and the worst case input size for the recursive call):

T(n) = **\_\_\_\_\_\_T(n-1) +20n+20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

How will the above recurrence change if you instead assume the best case input size for the recursive call):

T(n) = **\_\_\_\_\_\_T((n-1)/2) +20n+20\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6. (10 points)** Counting Sort

Show the B and C arrays after Counting Sort finishes on the array A [19, 6, 10, 7, 16, 17, 13, 14, 12, 9] if the input range is 0-19.

Text, letter

Description automatically generated

**C = [0,0,0,0,0,0,1,2,2,3,4,4,5,6,7,7,8,9,9,10]**

**B = [6,7,9,10,12,13,14,16,17,19]**

**7. (5 points)** Radix Sort

If Radix Sort is applied to the array of numbers [4567, 3210, 2345, 4321, 5678], show how these numbers will get rearranged after each of the four passes of the algorithm.

**3210 3210 3210 2345**

**4321 4321 4321 3210**

**2345 🡪 2345🡪 2345🡪 4321**

**4567 4567 4567 4567**

**5678 5678 5678 5678**

**8. (12 points)** Bucket Sort

Consider the algorithm in the lecture slides. If length(A)=15 then list the range of input numbers that will go to each of the buckets 0…14.

**Bucket0:**

**Bucket1:**

**Bucket2:**

**Bucket3:**

**Bucket4:**

**Bucket5:**

**Bucket6:**

**Bucket7:**

**Bucket8:**

**Bucket9:**

**Bucket10:**

**Bucket11:**

**Bucket12:**

**Bucket13:**

**Bucket14:**

Now generalize your answer. If length(A)=n then list the range of input numbers that will go to buckets 0,1,…(n-2), (n-1).

**Bucket0:**

**Bucket1:**

**Bucket(n-2):**

**Bucket(n-1):**

**9.** **(2 points)** Is the bucket-sort algorithm in-place? Why or why not?

**The bucket-sort algorithm is not in-place because it relies on storing data in “buckets”, which take extra memory spaces.**

**10.** **(3 points)** Suppose we are given a sequence of n elements, each of which is an integer in the range [0,n2-1]. Describe a simple method for sorting in O(n) time. Hint: Think of alternate ways of viewing the elements.

**We can use radix sort since it is a linear sorting algorithm. You can first represent each number as a “d”-digit number with base n. The conversion of each element into base-n, which would be a time complexity of O(n). The radix sort would be O(d\*n), which is of O(n) time complexity.**

**11. (10** points**)** Disjoint Set

Assume a Disjoint Set data structure has initially 20 data items with each in its own disjoint set (one-node tree). Show the final result (only show the array P for parts a, b & c below; no need to draw the trees) of the following sequence of unions (the parameters of the unions specified in this question are data elements; so assume that the find operation without path compression is applied to the parameters to determine the sets to be merged): union(16,17), union(18,16), union(19,18), union(20,19), union(3,4), union(3,5), union(3,6), union(3,10), union(3,11), union(3,12), union(3,13), union(14,15), union(14,3), union(1,2), union(1,7), union(8,9), union(1,8), union(1,3), union(1,20) when the unions are:

1. Performed arbitrarily. Make the second tree the child of the root of the first tree.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 0 | 1 | 14 | 3 | 3 | 3 | 1 | 1 | 8 | 3 | 3 | 3 | 3 | 1 | 14 | 18 | 16 | 19 | 20 | 1 |

b. Performed by height. If trees have same height, make the 2nd tree the child of the root of the 1st tree.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| -3 | 1 | 14 | 3 | 3 | 3 | 1 | 1 | 8 | 3 | 3 | 3 | 3 | 1 | 14 | 1 | 16 | 16 | 16 | 16 |

c. Performed by size. If trees have the same size, make the second tree the child of the root of the first tree**. Is the size of 3 all the children or just the ones directly connected to 3?**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 3 | 1 | -11 | 3 | 3 | 3 | 1 | 1 | 8 | 3 | 3 | 3 | 3 | 3 | 14 | 3 | 16 | 16 | 16 | 16 |

d. For the solution to part a, perform a find with path compression on the deepest node and show the array P after find finishes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 0 | 1 | 1 | 3 | 3 | 3 | 1 | 1 | 8 | 3 | 3 | 3 | 3 | 0 | 14 | 1 | 1 | 1 | 1 | 1 |

**12. (16 points)** Binomial Queue

First show the Binomial Queue that results from merging the two BQs below. Then show the result of an Extract Max operation on the merged BQ. There may be more than one correct answer.

Diagram

Description automatically generated

Extract Max operation:

Diagram

Description automatically generated