EC330 Applied Algorithms and Data Structures for Engineers
Spring 2021

Homework 4

Out: March 16, 2021
Due: March 26, 2021

This homework has a written part and a programming part. Both are due at 11:59 pm on March 26. You should submit both parts on Gradescope.

This is an individual assignment. See course syllabus for policy on collaboration.

1. **Does it sort? [10 pt]**
   For the following algorithm:
   - **i.** Does the algorithm correctly sort (in ascending order) all arrays A of size n containing only positive integers, for $n \geq 8$?
   - **ii.** If the answer to the previous question is "yes", provide a short explanation and a worst-case (asymptotic) running time. If "no" give a small counterexample (i.e. an input array and the corresponding output array that is not sorted properly).

```
sortB(Array A[i..j]) // call sortB(A[0..n-1]) to sort A[0..n-1]
   if (i == j)
      return
   mid = (i + j)/2
   sortB(A[i..mid])
   sortB(A[mid+1..j])
   if (A[i] < A[mid+1])
      swap(A[i], A[mid+1])
   return A;
```

2. **Heap [10 pt]**
   Given eight numbers $\{n_1, n_2, \ldots, n_8\}$, show that you can construct a heap (either min-heap or max-heap) using eight comparisons between these numbers (e.g., comparing n_1 and n_2 would be one comparison).

3. **Median [10 pt]**
   Describe an efficient algorithm for finding the median of all numbers between *two sorted arrays each of size* n. Your algorithm should be asymptotically faster than $\boldsymbol{O}(n)$. Analyze the running time of your algorithm.

4. **Hashing [20 pt]**
   Given a sequence of inputs 631, 23, 33, 19, 44, 195, 64 and the hash function h(x) = x mod 10, show the resulting hash table for each of the following cases of conflict resolution.

(a) Chaining
(b) Linear probing with $h_i(x) = [h(x) + i] \bmod 10$, $i = 0, 1, 2, \ldots, 9$
(c) Quadratic probing with $h_i(x) = [h(x) + i^2 + i] \bmod 10$, $i = 0, 1, 2, \ldots, 9$
(d) Using a second hash function $h_2(x) = 7 - (x \bmod 7)$

5. Programming [50 pt]

a) In this problem, you are tasked with sorting a string in *increasing* order based on the number of occurrences of characters. If there is a tie, output them based on *alphabetical order*, e.g., 'a' before 'e'. You can assume that all the characters are lower-case letters (so a total of 26 possible types of characters). Below are some example inputs and the corresponding expected outputs.

Input1: "engineers"
Output2: "girsnneee"

Input2: "engineering"
Output2: "rggiieeennn"

Implement *sortByFreq* in *sorta.cpp*. You are allowed to use *only* the libraries included in *sort.h* (you may not need to use all of them). You are welcome to implement your own data structure or use built-in ones like *int[]*. You *cannot* use any of the built-in sort functions including those provided by the <*algorithm*> library and will *need to implement your own* (you should write this as a separate function and call it from *sortByFreq*). **[20 pt]**

In the *written* part of your submission, *describe your high-level algorithm* and then *state and justify the time and space complexity* of your algorithm (in terms of *n* which is the length of the input string). **[10 pt]**

Your code will be graded not only on correctness but also on efficiency. For efficiency consideration, you should expect long string inputs (i.e. large *n*). **[5 pt]**

b) Implement *sortedMode* in *sortb.cpp*. Given a *sorted* vector of integers, *sortedMode* should return the integer that occurs the most often in the vector (this integer is also known as the *mode*). **[10 pt]**

In the *written* part of your submission, *describe your high-level algorithm* and then *state and justify the time and space complexity* of your algorithm. **[5 pt]**

The fastest solution(s) (which needs to be substantially faster than the rest for large inputs) will receive **10 bonus points**.