Programming Assignment #4

CSCE221 - Data Structure and Algorithms
Instructor: Mohsen Aznaveh
Texas A&M University

Due: Wednesday, April 20, 2020 11:59 pm

PA4 Instructions

- Basic task before starting assignment
  - Download PA4.zip from piazza and Unzip it.
  - Move PA4 folder to your CSCE-UIN local GitHub repository.
  - Go to terminal, see the new changes in your git repository by running ***git status*** command.
  - Add new changes i.e. a new PA4 folder, using command ***git add filename***
  - Make new commit of this change using ***git commit -m "commit message"***
  - Push the change using ***git push*** command.
- Once above tasks are done, you have basically pushed the PA4 code base without solution on your GitHub.
- Now start working on the assignment, complete it and push the completed assignment on GitHub along with **makefile** and **PDF** file which contains answers to few non-coding based questions and your explanations as part of this assignment.
- The PDF can be hand-written then scanned, or LaTeX, MS Word, what-ever you prefer. But only include the one and only one final PDF, not *docx, *tex, etc.

Important Notes

- Following actions will led to 0 points for this assignment:
  - Manually uploading your assignment using upload option on GitHub. This can be detected.
  - No makefile submission.
  - Using g++ *.cpp in your makefile.
  - Code plagiarism and cheating. It will be checked using MOSS.

Question 1

(70 points)

In the file `Sort.h` all the different sorting algorithms that we talked about is implemented. Using `q1.cpp` you can see if your implementations pass the tests. The tests worked based on your implementations and we will check your algorithms too.

a) Complete the method `isSorted`. This method reads the input vector and check if it is sorted. What is the complexity of this algorithm?(10 points)

b) complete the method `bubbleSort`. Use bubble sort algorithm to sort the input vector. What is the time complexity of this algorithm?(10 points)

c) Complete the method `selectionSort`. Use selection sort algorithm to sort the input vector. What is the time complexity of this algorithm?(20 points)

d) Complete the method `hybridSort`. Write a hybrid sort that split the input into 4 parts, sort each part using quick sort and then merge all the parts together. If input vector is smaller than 16 you can use any other sort. (20 points)

e) In `q1.cpp` you have the code for measuring execution time for some sorting algorithms. Add the code to measure the time for all of sorting algorithms that are in `Sort.h` then draw a chart that shows time complexity of different algorithms based on their running time. Some algorithms with $O(n^2)$ can take a long time to finish sorting.(20 points)

Question 2

(30 points) A Min Binary Heap is a complete binary tree with every node value lower than it's subtree node values. A complete binary tree can be represented using an array instead of creating a Node Structures with pointers using following rule -

- The root element will be at `arr[0]`.
- For `ith` node, i.e., `arr[i]`, the `arr[(i-1)/2]` is the parent node.
- For `ith` node, i.e., `arr[i]`, the `arr[(2*i)+1]` is the left child node.
- For `ith` node, i.e., `arr[i]`, the `arr[(2*i)+2]` is the right child node.

a) File `MinHeap.h` contains the class, attributes and functions of MinHeap and file `q2_1.cpp` is testing the code of `MinHeap.h`. Implement following functions in `MinHeap.h` file. (10 points)

- `void insertKey(int k)` - Insert a new element in the min heap.
- `int extractMin()` - Extract the minimum element from the min heap and reconstruct the min heap.

- `void decreaseKey(int i, int newVal)` - Decrease the element at index `i` to `newVal` and reconstruct the min heap.
- `void deleteKey(int i)` - Delete the element at index `i` and reconstruct the min heap.

`Sample output after executing q2_1.cpp - 2 4 1`

b) There is no programming needed for this part. Draw the tree in your PDF file. (6.2 from the book, 5 points):

- Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2, one at a time, into an initially empty binary heap.
- Show the result of using the linear-time algorithm to build a binary heap using the same input.

c) Given an array of `n` integers, find the `kth` smallest element from the array. In the file `q2_3.cpp` implement function named `int kthSmallest(int arr[], int n, int k)` that will return the `kth` smallest element from the array. (15 points)

Note: Time complexity of your algorithm should be less than `O(nlog(n))`. And `O(nk)` is greater than `O(nlog(n))`. Describe what is the complexity of the algorithm. **Hint:** Min Heap :)

`Sample input/output after executing q2_3.cpp -`

```
Enter the value of n: 5
Enter the value of k: 3
Enter n integers space separated: 5 4 3 1 2
Kth Smallest Integer: 3
```