COMP160: Algorithms, **Homework 6**

- To obtain full credit, you must justify your answers. When describing an algorithm, do not forget to **state any assumptions that you make**, analyze its running time and explain why the algorithm is correct.
- Although not specifically stated, You can assume that we look for algorithms that are as fast as possible, and bounds as tight as possible.
- You may discuss these problems with others, but remember to write the answers on your own. In case of doubt, cite any source you used to do the assignment.
- Remember to submit each question in a separate page.

1. Suppose you are given the following set of intervals on the real number line:

   (30, 45), (75, 83), (15,55), (5, 10), (0, 10) (90, 95), and (50, 100)

   (a) Insert these intervals in an AVL tree using the left endpoint as the key. Draw the final tree

   (b) We want to augment the tree to answer the following query: given a new real number $q \in \mathbb{R}$, we want to know if there is any interval in the data structure that intersects q (in other words, the algorithm should return `yes` if there is an interval in the data structure that contains q, `no` otherwise). Indicate what information you will augment the tree with, and the steps of the algorithm.

   (c) Go back to the tree drawn in the first question and add the extra information that you annotated.

   (d) Can you modify the search algorithm so that it reports *all* intervals that contain q? What do you need to change? What is the new runtime?

   (e) Run your algorithm on the augmented tree from the previous question on the following query numbers. For each one describe your decisions and the status as you traverse nodes. Run the algorithm that reports all intersections (that is, your answer to question (d)). Does the answer of your algorithm make sense?

      i. 12

      ii. 53

2. We consider a variation of the classic Stack where you have three operations: Push, Pop, and Multipop. the first two operations are the classic ones from the stack where we add or remove a single element. The Multipop operation receives one integer k as parameter and removes in one operation k many elements from the stack.

   (a) Assume that the stack is implemented under the hood with a linked list. What is the worst-case runtime of a single Multipop operation?

(b) Give an informal (2-3 sentence) argument of why a sequence of n operations in any combination of PUSH, POP, and MULTIPOP should take $O(n)$ time.

(c) Now we will try to give a proof using the accountant's method. What virtual cost would you give to each operation (i.e. how many coins would you deposit)? What is the real cost of each operation?

(d) Justify that you never run out of coins.

(e) Assuming that your previous answers were correct, justify that any sequence of n operations in any combination of PUSH, POP, and MULTIPOP will take $O(n)$ time (use a few bullet points.)

(f) Say that we now add an additional operation called MULTIPUSH. This operation receives two parameters (an object o and an integer k) and simply pushes k many times into the stack object o. Does this change your amortized cost? Justify your answer either way with a couple of sentences. If it changes the total cost, give the new amortized cost of n operations.

3. Design a data structure to support the following two operations for a set S of integers:

   Insert(S,x) Inserts x into set S

   Delete-Half(S) deletes the largest $n/2$ (rounded up) elements from S

   We want a this data structure so that any sequence of m operations can be executed in $O(m)$ time (for simplicity, assume that the DS starts empty). Your data structure must also be able to report all elements currently remaining in S (in $O(n)$ time, where n is the number of remaining elements).

   (a) Describe what data structure would you use.
   HINT: it is a simple structure explained in Comp 15.

   (b) How would you handle INSERT(S,X)? What is the runtime of a single such operation?

(c) How would you handle and a DELETE-HALF(S)? What is the runtime of a single such operation?

(d) We will use the potential method to bound the amortized runtime. When using the potential function, it is hard to actually measure the exact runtime. Instead, we should measure something different (say, in the bit counter example we looked at number of bits changed). What should we use to assess runtime for this problem?

(e) Describe the potential function Φ that you want to use. What is the potential of the empty data structure?

(f) Show that the amortized cost of any operation $\hat{c}_i$ is constant.

Note: for this question you can assume any result explained in lectures (no need to prove it, just state it).

(g) Describe (in 3-6 bullet points) why the fact that $\hat{c}_i = O(1)$ implies our final goal (i.e., any sequence of m operations runs in $O(m)$ time).