CS202, Fall 2024

Homework 2 - AVL Trees and Heap

Due: 23:59, 06/11/2024

Before you start your homework, please read the following instructions carefully:

FAILURE TO FULFIL ANY OF THE FOLLOWING REQUIREMENTS WILL RESULT IN A GRADE SCORE OF 0 (zero) WITHOUT ANY CHANCE OF REDEMPTION.

- See the course page for any late submission policies and Honor Code for Assignments.
- Upload your solutions in a single ZIP archive using the Moodle submission form. Name the file as studentID_name_surname_hw2.zip.
- Your ZIP archive should contain **only** the following files:
  - **studentID_name_surname_hw2.pdf**, the file containing the answers to Questions 5 and 6.
  - In this assignment, you must have separate interface and implementation files (i.e., separate .h
  - and .cpp files) for your class. Your class name MUST BE **Heap** and your file names MUST BE
  - **Heap.h** and **Heap.cpp**. You have to implement **least5.cpp**, **game.cpp**, **prefixsubarrays.cpp** and **subarrays.cpp** to solve and test subtasks.Note that you may write additional class(es) in your solution.
  - Your .cpp, .h , .pdf files, and **Makefile** , which produces executables for each subtask (No Makefile results in 50% deduction in points).
  - Do not forget to put your name, student id, and section number in all of these files. Comment your implementation well. Add a header (see below) to the beginning of each file:

```
/**
 * Title: AVL Trees & Heap
 * Author: Name & Surname
 * ID: 12345678
 * Section : 1
 * Homework : 2
 * Description : description of your code
 */
```

  - Do not add unnecessary files, such as the auxiliary files generated from your preferred IDE.
  - Please do not use **Turkish** letters in your file and folder names.
- Your code must compile.
- Your code must be complete.
- You ARE NOT ALLOWED to use any data structure or algorithm-related function from the C++ standard template library (STL) or any other external libraries.
- We will test your code using the **Google test library**. Therefore, the output message for each operation in Questions 1-4 MUST match the format shown in the output of the example code.
- Your code must run on the **dijkstra.cs.bilkent.edu.tr** server.
- For any questions related to the homework, contact your TA: *ziya.ozgul@bilkent.edu.tr* or you can ask your questions on forum.

Question 1 - 20 points

You are given an empty min-heap, and there are consecutive actions. Action can be adding an element to the heap, removing the top of the heap, and getting the least 5 elements of the heap. However, you are not allowed to remove any element from the heap during querying. You are expected to return at least(smallest) 5 elements in asymptotically O(1) time. Sample input and output are given below. The first line represents n: number of actions. Adding an element is denoted by “a value”, removing is denoted with “r” with no parameter, and getting the least 5 elements is denoted with “g”.

```
Sample Input
10
a 1
g
a 2
a 6
a 8
a 10
g
a 4
g
r
g
Sample Output
-1
1
2
6
8
10
1
2
4
6
8
2
4
6
8
```

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./least5 <input_filename> <output_filename>
```

Assuming that you have an executable called “least5”, this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 2 - 20 points

Bobô and Holosko are playing a card game with a stack of cards numbered from 1 to 2 N. They randomly distribute the cards such that both of them have N cards, all cards are assigned a unique number. In each round, one card from Bobô's and Holosko's deck is picked. Players can pick any pair of cards on each deck. If the number on Bobô's deck is bigger than the one picked from Holosko's deck, Bobô gets 1 point; otherwise, Holosko gets 1 point. Bobô starts the game. This pattern continues, with Bobô picking the cards on odd turns and Holosko picking on even turns. No card can be picked up twice. Assuming both of them play optimally, you are asked to calculate the score of Bobô and Holosko given a specific card distribution. Your program must output the final score.

Sample Input

```
4
1 3 5 7
2 4 6 8
```

Sample Output

```
2 2
```

Constraints: N <= 1000000, and your code must work in O(N * log N) time to get a full credit.

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./game <input_filename> <output_filename>
```

Assuming that you have an executable called "game1", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 3 - 20 points

Ziya is very interested in arrays and has several array questions to solve. One of them is related to subarrays. A subarray is defined as any consecutive part of an array. For this problem, he only cares about subarrays starting from the first index of the array. For example, if we have an array with N elements, he is only interested in [1,1], [1,2], [1,3], [1,2,3,4] .. and so on. He has two arrays A and B, with N elements, and he selects subarrays of his interest. He wants to permute the elements of A (not B) so that at least in M indices A[i] > B[i]. However, such a solution is not possible for every subarray. Therefore, he wants to find the minimum length of subarrays, A[1:L] and B[1:L], such that after permuting A[1:L], there are at least M indices such that A[i] > B[i] where 1<=i<=L. In the first line of input, N and M are given, respectively. In the second line, the contents of A are given. In the third line, the contents of B are given. You are to output the value of L.

Sample Input

6 3
2 4 10 6 1 11
3 5 8 9 7 12

Sample Output

4

Explanation:

For L = 4, A can be permuted in a way that A[1:4] = { 4, 6, 2, 10}. So, 4 > 3, 6 > 5, 10 > 9; and M=3.

Constraints: N <= 1000000, M <= N, your solution must work in O((N * log N * log M) to get full credit.

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./prefixsubarray <input_filename> <output_filename>
```

Assuming that you have an executable called "prefixsubarray", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 4 - 20 points

Ziya is still very much interested in arrays and has several array questions to solve. One of them is related to subarrays. A subarray is defined as any consecutive part of an array. He has two arrays A and B, with N elements, he selects subarrays of his interest, denoted as A[left:right], B[left:right]. This subarray is called as "good" subarray if there exists a subset of A with M elements and a subset of B with K elements such that any element of subset of A is greater than any element of subset B. Ziya wants to find the length of the shortest good subarray. In the first input line, N, M, and K are given respectively. In the second line, A is given. In the third line, B is given.

Sample Input

6 3 2
2 1 10 16 4 11
3 5 8 9 7 12

Sample Output

4

Explanation:

For subarray [3:6] , subset of A = { 10,11, 16}, subset of B = {8,9} , all elements of subset A is greater than all elements of B. Length of subarray is 5-2+ 1 = 4.
Constraints: N <= 1000000, M, K <= N, your solution must work in O((N * log N) to get full credit.

The name of the input file will be provided as a command line argument to your program. Thus, your program should run using two command line arguments. Thus, the application interface is simple and given as follows:

```
username@dijkstra:~>./subarray <input_filename> <output_filename>
```

Assuming that you have an executable called "subarray", this command calls the executable with one command line argument, the name of the file from which your program reads the number of strings and strings.

Note that outputfile name is optional.

Question 5 - 5 points

In Question 1, you get the least 5 elements without removing the top of the heap. If you are asked to get least k elements, what will be the time complexity ? Is the approach you use in Question 1 still applicable for larger k values ?

Question 6 - 15 points

What have you learned from this homework ? How did you solve the questions? Explain your solutions for each question. Explain the data structures you used and why you use them. Discuss the time complexity of your solutions. You are expected to make your explanations separately for each question.

Important

- This homework requires you to use one of the common heap operations : decrease key or increase key. Please learn this carefully.

- You are expected to solve the questions using either any kind of binary search tree or AVL trees you learned in class or heap.

- Think carefully about the questions.

- Executable names are very important. You must use executable names given above.

- Please write your makefiles carefully

- Test your code on dijkstra.