Data Structures and Algorithms
Spring 2019
Assignment 3

IT University Innopolis

Contents

1 About this homework

1.1 Coding part

For practical (coding) part you have to provide you program as a single Java class file that contains all necessary definitions and a `main` method.

The program should read from standard input and write to standard output unless mentioned otherwise.

1.1.1 Preliminary tests

For some coding parts a CodeForces or CodeTest system will be used for preliminary tests. You will have to specify a submission number in those systems as a proof that your solution is correct. Only correct solutions will be assessed further by TAs.

1.1.2 Code style and documentation

The source code should contain adequate internal documentation in the form of comments. Internal documentation should explain how the code has been tested, e.g. the various scenarios addressed in the test cases.

Do not forget to include your name in the internal documentation.

All important exceptions should be handled properly.

Bonus points might be awarded for elegant solutions, great documentation and the coverage of test cases. However, these bonus points will only be able to cancel the effects of penalties.

1.2 Theoretical part

Solutions to theoretical exercises have to be elaborate and clear. Write all solutions in a single document and make it a PDF for submitting.

Do not forget to include your name in the document.

1.3 Submission

You are required to submit your solutions to all parts via Moodle.

For this assignment you will need to submit: - Java class file(s) for the coding parts; - submission number or a link to submission in CodeForces as a proof of correctness; - a PDF file with solutions to theoretical parts.

Submit files as a single archive with your name and group number. E.g. `BS18-00_Ivanov_Ivan.zip`.

1.4 Plagiarism

Plagiarism will not be tolerated and a plagiarised solutions will be heavily penalised for all parties involved. Remember that you learn nothing when you copy someone else's work which defeats the purpose of the exercise!

You are allowed to collaborate on general ideas with other students as well as consult books and Internet resources. However, be sure to credit all the sources you use to make it clear what part of your solution comes from elsewhere.

2 Coding part

2.1 Mergeable Heap ADT (25 points)

Implement Mergeable Heap ADT consisting of the following operations:

- `insert(k, v)` — insert value `v` with key `k`;
- `max()` — get the key-value pair with maximum key;
- `removeMax()` — remove maximum key-value pair from the heap;
- `merge(h)` — merge another heap `h` with the present one, incorporating all entries into the current one while emptying `h`.

Provide an implementation for Mergeable Heap ADT. Prove that your implementation achieves $\mathcal{O}(\log n)$ performance for all operations, where n denotes the size of the heap (number of entries).

2.2 TODO-list (25 points)

Implement a program that manages a TODO-list of a certain user. Such a list is in fact a priority queue of tasks. A task consists of a name (a short description) and priority value (an integer). The functionalities of such a system include:

- Given a list of tasks with priorities, create a new TODO-list;
- Add a new prioritized task to an existing TODO-list;
- Solve the task with the highest priority from the TODO-list (in other words, remove the task from the list);
- Merge two TODO-lists.

For implementing this system, you should necessarily use one of the following concepts: binary heaps, binomial tree or binomial heaps. Note that you can also use your mergeable heap from the first part of this assignment.

2.2.1 Input and output

The input starts with a number N, followed N command lines. Each command can be one of:

- create new task: `[date](dd.mm.yy) [task_name](1 word - String) [priority_value](Integer)`
- delete task with the highest priority from the list: `DO [date](dd.mm.yy)`

Output should contain final TODO list for each date (in ascending order) and also the combined TODO-list (for all dates). In each TODO-list tasks should be ordered by priority (from highest to lowest).

2.2.2 Examples

Standard Input	Standard output
6 10.04.19 DSA_Read_Leacture 15 10.04.19 DSA_Read_Tutorial 10 10.04.19 DSA_Assigment 25 12.04.19 DSA_Visist_Lab 20 12.04.19 DSA_Do_Lab_Task 10 DO 12.04.19	TODOList 10.04.09 DSA_Assigment DSA_Read_Leacture DSA_Read_Tutorial TODOList 12.04.09 DSA_Do_Lab_Task TODOList DSA_Assigment DSA_Read_Leacture DSA_Read_Tutorial DSA_Do_Lab_Task

6 11.04.19 Motivation_Letter 15 12.04.19 Read_article 10 12.04.19 Presentation 25 12.04.19 Solution_Analyses 25 DO 12.04.19 DO 11.04.19	TODOList 11.04.19 TODOList 12.04.09 Solution_Analyses Read_article TODOList Solution_Analyses Read_article

3 Theoretical part

3.1 Binary Heaps (20 points)

Answer the following questions about binary heaps:

a. Provide the *binary min-heap* resulting from inserting the following entries (in this order): **76, 21, 9, 69, 15, 33, 12, 7**.

   Remember that you should start with an initially empty binary min-heap. You are *required* to show the intermediate steps and circle your final heap to have the question graded.

b. Show the binary min-heap resulting from running two `removeMin()` on the final heap produced in the previous step. You are *required* to show the intermediate steps and circle your final heap to have the question graded.

c. What is the maximum height of a binary heap with n keys? Your answer should be followed by your reasoning.

3.2 Binomial trees and binomial heaps (20 points)

Answer the following questions about binomial heaps:

a. This question concerns the algorithm for union (or merge) of two binomial heaps (as presented in the tutorial). Explain why in case 2 (when x is the first of a sequence of three binomial trees with the same degree) we march x instead of merging it to the next tree. Your answer should be succinct, clear, and take at most half of a page. You may use illustrations if necessary.

b. Is the following statement TRUE or FALSE? Your answer should be followed by your reasoning. A binomial tree of height d has at least 2^d nodes, assuming that the depth of the root node is zero.

3.3 Graph representation (10 points)

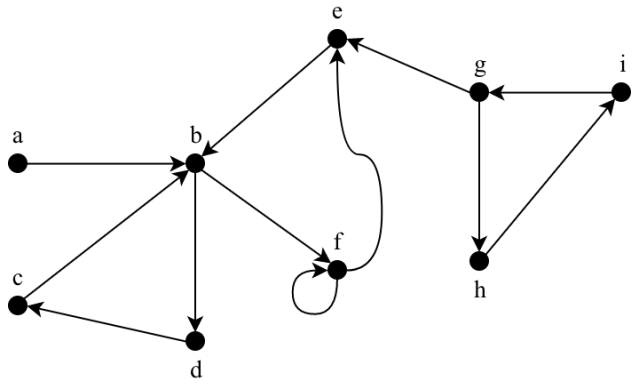

Figure 1: Graph Γ.

Given directed graph Γ, answer the following questions:

- Show a table with the in-degree, out-degree and degree of c and f.
- Is Γ strongly connected? Justify your answer. How many strongly connected components does Γ have?
- Show the adjacency matrix of Γ with the vertices ordered alphabetically.

Prove that any undirected graph has an even number of vertices of odd degree by using the Handshaking Theorem. Your twofold answer should necessarily include a proof by contradiction and also a proof by induction.