UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 5

Deadline: Friday 8 April 2022, 11:59

PLAGIARISM POLICY

UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

Objectives

The aim of this practical is to learn how to implement different types of binary heaps.

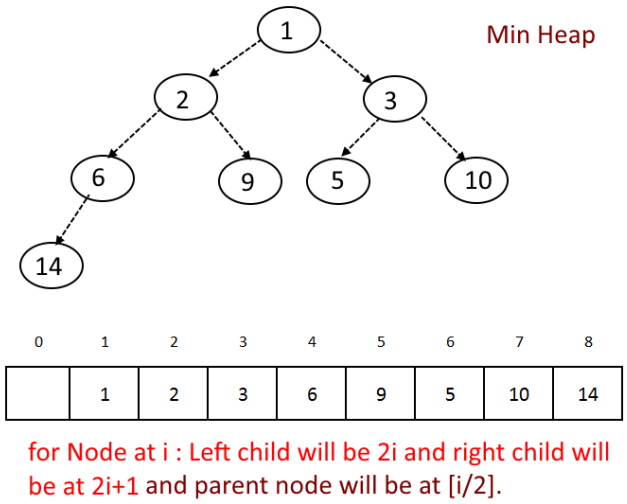
Instructions

Complete the tasks below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a compressed archive before the deadline. Please comment your name **and** student number in at the top of each file.

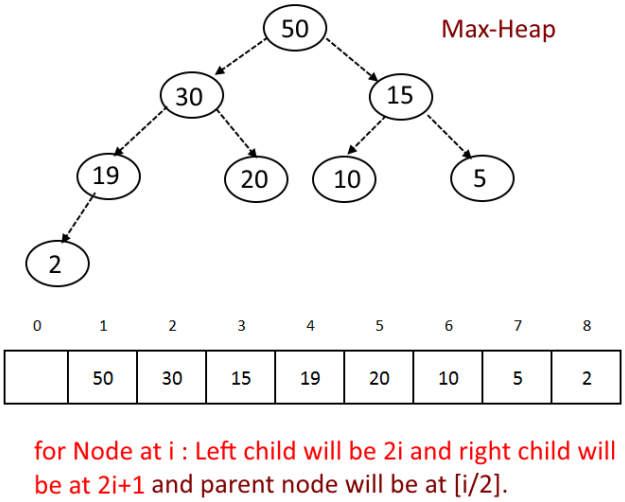
Task 1: Heaps [42]

Binary heaps are a particular type of binary tree with the following two properties:

1a. The value of each node is smaller than or equal to the values stored in either of its children. This makes the tree then a min-heap.



1b. The value of each node is greater than or equal to the values stored in either of its children. This makes the tree then a max-heap.



2. The tree is perfectly balanced, and the leaves in the last level are all in the leftmost positions.

As described in section 6.9.2 of the textbook and illustrated above, a heap can be implemented using an array to store its elements. You need to implement a simple array based heap with a number of operations. The heap should be created using Williams' top-down method. You have been given partially implemented heap classes to use for that purpose. Your task is to implement the following methods according to the given specification:

Heap.java

`boolean isEmpty()`

This function should determine whether the heap is empty or not. It returns true if the heap is empty and false otherwise.

MinHeap.java

`void insert(T elem)`

Insert the given element `elem` into the heap. If the element is already in the heap, insert it again as duplicates are allowed. Restore the heap property after insertion.

`T removeMin()`

Remove the minimum element from the heap and return it. Restore the heap property after removal. If the heap is empty, return null.

`void delete(T elem)`

Delete the given element `elem` from the heap. Restore the heap property after deletion. If the element `elem` is not in the heap, do nothing and return.

MaxHeap.java

`void insert(T elem)`

Insert the given element `elem` into the heap. If the element is already in the heap, insert it again as duplicates are allowed. Restore the heap property after insertion.

`T removeMax()`

Remove the maximum element from the heap and return it. Restore the heap property after removal. If the heap is empty, return null.

`void delete(T elem)`

Delete the given element `elem` from the heap. Restore the heap property after deletion. If the element `elem` is not in the heap, do nothing and return.

Only implement the methods listed above. You may use your own helper functions, such as methods for moving up or moving down an element, to assist in implementing the specification. However, you may not modify any of the given method signatures. Also do not modify any of the other code that you were given for this task.

Submission

You need to submit your source files on the Assignment website `https://ff.cs.up.ac.za/`. All tasks need to be implemented (or at least stubbed) before submission. Place **Heap.java, MinHeap.java and MaxHeap.java** files in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 4 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 5* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.