Institut for Matematik og Datalogi
Syddansk Universitet, Odense

17. March 2025

Project: Part I

DSK814

This project consists of three parts: each part has its own deadline, so that the work is spread over the entire semester. The deadline for part I is *Friday, April 04 at 11:59 p.m.* The three parts I/II/III are not equal in size, but are approximately 15/25/60 in size.

The project must be completed in groups of two or three.

Goal

The overall goal of the project is training the transfer of the course's knowledge of algorithms and data structures to programming. The project and the written exam complement each other, and the project is not intended as preparation for the written exam. The specific goal of part I is to implement the data structure *priority queue*. The priority queue will be used again later in Part III of the project.

Task

In short, the task is to transfer the textbook's pseudo-code for priority queues to a Python program and then test it—which includes sorting numbers.

Requirements

Your program should be called `PQHeap.py` and it must implement the data structure *Heap*. The heap must consist of a list `A` of numbers (i.e. use a list instead of the book's array). The program should contain the following functions:

- `extractMin(A)`: Remove the item with the least priority from the priority queue `A` and return it.
- `insert(A,e)`: Insert the element `e` in the priority queue `A`.
- `createEmptyPQ()`: Return a new, empty priority queue (i.e., an empty list).

You must base the implementation on the pseudocode in Cormen et al. Chapter 6, on pages 175, 165 and 162, as well as on the summary of the pseudocode from page 176, which can be found further down in this project description [in the 3rd edition the corresponding page numbers are 163, 154 and 152, as well as 164].

For simplicity, assume that `extractMin(A)` is only called on a non-empty priority queue. It is left to the user of the priority queue to ensure this, for example by keeping track of the number of elements in the priority queue. In addition to `extractMin(A)`, `insert(A,e)`, and `createEmptyPQ()`, you will need to implement functions for internal use within the program.

From Pseudocode to Implementation

1. In this project, you will create a *min*-heap structure, while the book formulates its pseudo-code for a *max*-heap structure. Therefore, remember to reverse all inequalities between elements in the pseudocode (but not inequalities between indices).
2. Since the list can get bigger and smaller using the methods `append(x)` and `pop()`, the heap-size value does not need to be maintained, but can be found via the function `len(A)`.
3. The book's pseudocode indexes arrays starting with 1, while Python starts with 0. Thus, the formulas for left child, right child, and parent need to be adjusted to the following:
   - $\textsc{Left}(i) = 2i + 1$
   - $\textsc{Right}(i) = 2i + 2$
   - $\textsc{Parent}(i) = \lfloor (i - 1)/2 \rfloor$

4. This also means that the first index (root) is 0 (not 1), and that the last index (last leaf) is `len(A)-1` (not the heap size). In the pseudocode, applications of indices and comparisons between indices should be adjusted accordingly.

5. In the pseudocode on page 175 [page 163], the `if` in lines 1-2 is omitted due to the assumption that the priority queue is not empty.

6. On page 176 [page 164], the two parts of pseudocode on the page can be combined, since we don't need `increaseKey`. This gives the following variant of the pseudocode for `insert`. This variant will be used in the project.

Algorithm: MAX-HEAP-INSERT(A, key)

A.heap-size = A.heap-size + 1
$i = A$.heap-size
$A[i] = key$
while $i > 1$ *and* $A[\text{PARENT}(i)] < A[i]$ **do**
    exchange $A[i]$ with $A[\text{PARENT}(i)]$
    $i = \text{PARENT}(i)$

Note that the above pseudo-code uses the same conventions as the one in the book and therefore needs to be adjusted as well.

Test

Test your code thoroughly! This includes testing it with the provided program `PQSort` that sorts by repeatedly making `insert`'s in a priority queue, followed by repeated `extractMin`'s. The program `PQSort` reads from standard input (via the file object `sys.stdin`), which by default is the keyboard, and writes to standard output, which by default is the screen. Input to `PQSort` is a sequence of characters representing integers, separated by newlines (i.e. one number per line). The program outputs the numbers in sorted order, separated by newlines. As an example, `PQSort` is called like this in a command prompt:

```
python PQSort.py
34
645
```

```
-45
1
34
0
Control-D
```

(Control-D indicates end of data under Linux and Mac, under Windows use Control-Z instead) and then gives the following output on the screen:

```
-45
0
1
34
34
645
```

By redirecting standard input and output, you can apply the same program to files in a command prompt like this:

```
python PQSort.py < inputfile > outputfile
```

Before submission, you must test sorting all the provided data files in this way. An important reason why you should try the above method (with redirection in a command prompt) is that the programs must be able to be tested automatically after submission.

Formalities

You only need to submit your Python source file `PQHeap.py`. It must contain the names and SDU logins of the group members.

The file must be submitted electronically in `itslearning` in the Project folder under the Resources tab. The submission module is also inserted into a `itslearning` plan in the course.

During the submission, you must declare the group by stating the names of all members. You only need to submit once per group. Please note that

during the submission, you can create temporary "drafts", but you can only submit once.

Submit by:

Friday, April 04 at 11:59 p.m.

Please note that submitting someone else's code or text, whether copied from fellow students, from the internet, or in other ways, is exam cheating, and will be treated very seriously according to current SDU rules. You will also not learn anything. In short: only people whose names are mentioned in the submitted file may have contributed to the code.