Name, SID, Date ....

In Class Assignment 24: Max-Priority Queue Data Structure

Benjamin Sanders, MS November 25, 2020

1 Introduction

You may work in groups of up to two or three students. Write all student names at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

....
....
....
....

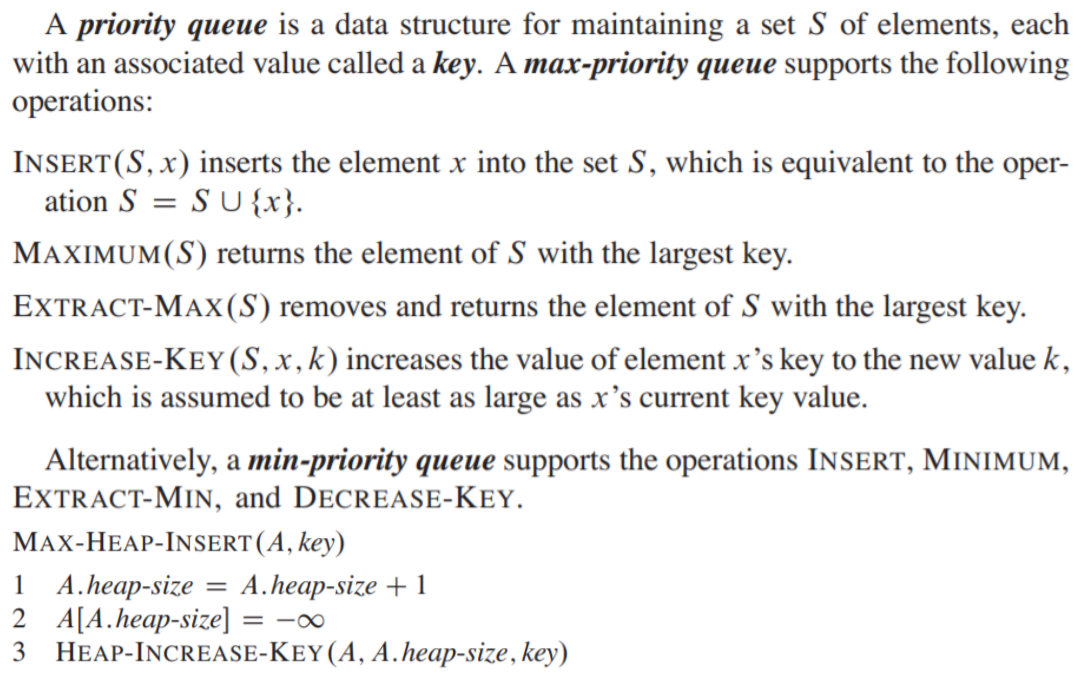
2 Assignment Description

2.1 Big Picture

This is used for scheduling jobs on a computer processor and is often implemented in hardware.

2.2 Algorithm Implementation

Implement the following algorithms in Java, using the Vector data structure for any 1-D array, 2-D array, or linear algebra purposes.



A ***priority queue*** is a data structure for maintaining a set S of elements, each with an associated value called a ***key***. A ***max-priority queue*** supports the following operations:

INSERT(S, x) inserts the element x into the set S, which is equivalent to the operation $S = S \cup \{x\}$.

MAXIMUM(S) returns the element of S with the largest key.

EXTRACT-MAX(S) removes and returns the element of S with the largest key.

INCREASE-KEY(S, x, k) increases the value of element x's key to the new value k, which is assumed to be at least as large as x's current key value.

Alternatively, a ***min-priority queue*** supports the operations INSERT, MINIMUM, EXTRACT-MIN, and DECREASE-KEY.

```
MAX-HEAP-INSERT($A$, $key$)
1  $A.heap\text{-}size = A.heap\text{-}size + 1$
2  $A[A.heap\text{-}size] = -\infty$
3  HEAP-INCREASE-KEY($A$, $A.heap\text{-}size$, $key$)
```

Name, SID, Date ....

```
HEAP-MAXIMUM($A$)
    return $A[1]$
```

```
HEAP-EXTRACT-MAX($A$)
    if $A.heap\text{-}size < 1$
        error "heap underflow"
    $max = A[1]$
    $A[1] = A[A.heap\text{-}size]$
    $A.heap\text{-}size = A.heap\text{-}size - 1$
    MAX-HEAPIFY($A, 1$)
    return $max$
```

```
HEAP-INCREASE-KEY($A, i, key$)
    if $key < A[i]$
        error "new key is smaller than current key"
    $A[i] = key$
    while $i > 1$ and $A[\text{PARENT}(i)] < A[i]$
        exchange $A[i]$ with $A[\text{PARENT}(i)]$
        $i = \text{PARENT}(i)$
```

```
MAX-HEAPIFY($A, i$)
    $l = \text{LEFT}(i)$
    $r = \text{RIGHT}(i)$
    if $l \le A.heap\text{-}size$ and $A[l] > A[i]$
        $largest = l$
    else $largest = i$
    if $r \le A.heap\text{-}size$ and $A[r] > A[largest]$
        $largest = r$
    if $largest \ne i$
        exchange $A[i]$ with $A[largest]$
        MAX-HEAPIFY($A, largest$)
```

```
PARENT($i$)
    return $\lfloor i/2 \rfloor$
```

```
LEFT($i$)
    return $2i$
```

```
RIGHT($i$)
    return $2i + 1$
```

2.3 Time Complexity Analysis

Where n is the number of data points to be stored, analyze the time complexity of the given algorithms with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n^2)$ in the space below.

....

Name, SID, Date ....

2.4 Space Complexity Analysis

Where n is the number of data points to be stored, analyze the space complexity of the given algorithms with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n \cdot log(n))$ in the space below.

....

2.5 New Algorithms Design and Implementation

In the space below, design algorithms that implement a min-priority queue. Compose HEAP-MINIMUM, HEAP-EXTRACT-MIN, HEAP-DECREASE-KEY, and MIN-HEAP-INSERT. Use pseudocode written in a style similar to the given algorithms, and implement it in Java. You may use as many additional pages as necessary for this purpose.

3 What to Turn In

Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.