

**Technical Report: Patients or Patience**

**Authors: James Price**

**Date: DATE**

**Wordcount: NUMBER Words**

**Pagecount: NUMBER Pages**

**Confidential: NO / YES – INTERNAL ONLY/YES – X DEPARTMENT ONLY**

Contents

[Executive Summary 2](#_Toc38738165)

[Introduction 1](#_Toc38738166)

[Theory 1](#_Toc38738167)

[Implementation 1](#_Toc38738168)

[Conclusions 2](#_Toc38738169)

[References 2](#_Toc38738170)

[Appendix A 2](#_Toc38738171)

# Executive Summary

# Introduction

Hosppital A&E departments can receive wide array of different cases being presented on a day to day basis, with some cases being more urgent than others. Hospitals also have a time constraint in regards to getting patients senn on time. In the given brief, the time constraint was 4 hours and if a patient is not seen to in under that amount of time the hospital is fined £10,000. As mentioned before, some cases are more urgent than others and as a result they require a higher priority to ensure they are seen as soon as possible, so the program requires that cases can be queued in order of FIFO; however that can be overridden by the priority is it is high enough.

# Theory

A statistics paper for the NHS (Baker, 2020) states that in 2019 the average attendances per day for all Type 1 A&E departments across the country was at 44,366 and assuming an even distribution of attendances across all 132 NHS trusts operating a Type 1 A&E department (What’s going on with A&E waiting times?, 2020), there would be aproximatly 336 admitances per day. Given this information, it is clear that the solution implemented for priority queuing would need to be able to operate fast with large quatities of data; consequently, the data structure implemented would need to be able to operate with the fastest time complexity. This is why a Binary heap will be implemented.

Binary (Max) heaps are based on the binary tree with two extra constraints:

1. The parent node must have an equal or higher value than its children
2. The heap is held as a mostly complete binary tree meaning all nodes have two children except for the second deepest layer and leaf nodes

Elements in a list can be treated as nodes of a binary heap using the current index of the selected element and simple formulae to calculate the parent and child nodes (if any):

When I = index of particular node:

Parent node = (i-1)//2

Left child node = 2i+1

Right child node = 2i+2

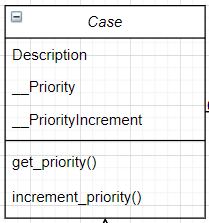
A binary heap would be ideal for the task because of it’s worst case time complexities, which would be especially idea for large quatities of nodes as binary heap has the following worst case time complexities:

|  |  |
| --- | --- |
| **Operation** | **Time complexity (BigO)** |
| Insert | O(log n) |
| Delete | O(log n) |
| Heapify | O(log n) |
| Peek at root | O(1) |

# Implementation

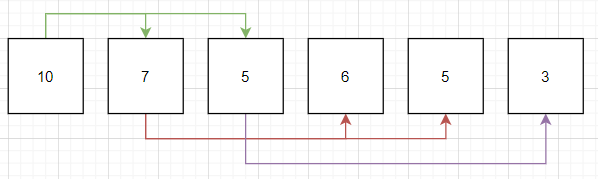
## Cases

In order to conform with the brief, the cases that are stored in the queue must have a description and a priority, but the priority must increment by a predetermined amount every 10 minutes; therefore, for the implementation of a case class there would need to be a case description, priority and a priority increase amount defined. In order to protect the priority and increment values, they are made private to the class and there are proceedures implemented to utilise and return the private values.



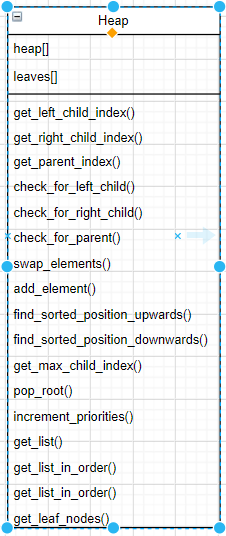
## Heap

The Heap class holds all of the case elements and has all of the algorithms for adding, removing, sorting and checking the root value. The actual values are stored in the program with an array but the operations that are done to calculate the index positions and manipulate data, **treats the array like a heap**. Here is an example array of numbers and how the parent-child relationships would connect those elements:

****

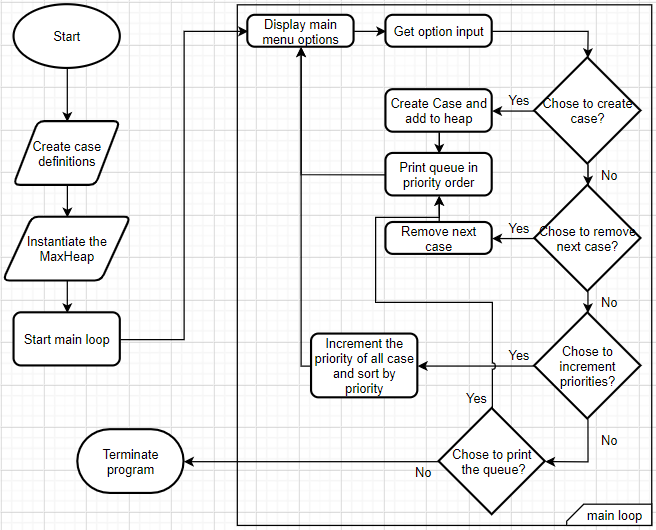
The value 10 (at index 0) will have children 7 and 5 (at indecies 1 and 2 respectively) which was calculated using the formulae **leftChild = 2i+1 and rightChild = 2i+2**.

Below is a list of proceedures and attributes present in the MaxHeap class:



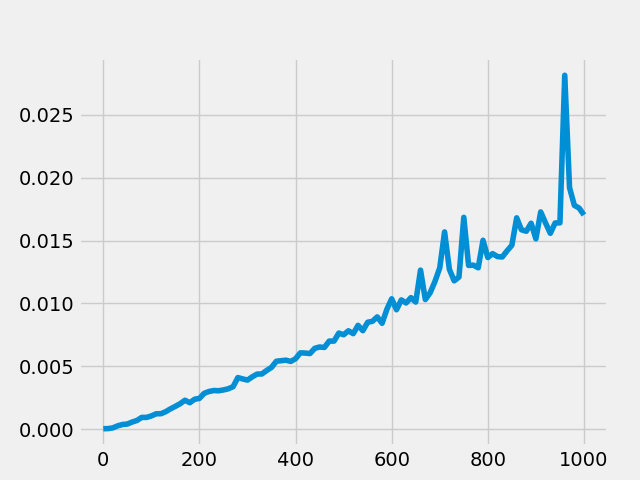
## Main

The main class handles the operation of the MaxHeap class and takes the user input to traverse menus and delete or create cases to be handled in the heap. A simple flowchart below shows the operation of the main class:



# Conclusions

Firstly the process of adding elements to the heap and sorting them into the correct position has the worst case time complexity of O(n). The operation to append a case to the heap is O(1) and the operation of sorting that case into the correct position is O(n) and this is seen in Figure 2.1 as that has a linear increase in time taken to add cases to the heap and sort it into place.

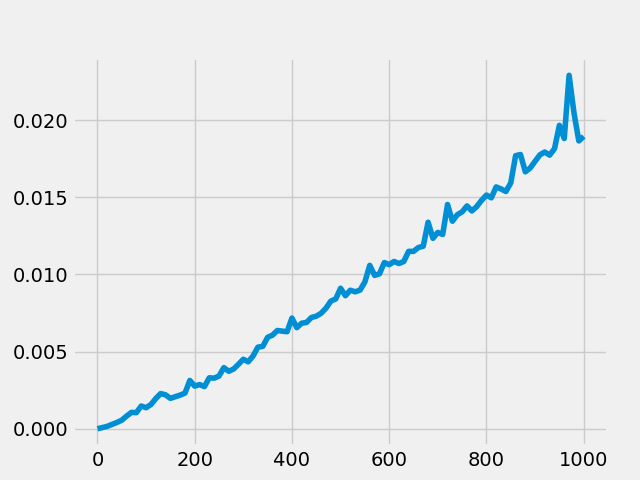


The time taken to add and sort a number of elements into place (seconds)

The number of elements being added to the heap

Figure 2.1: The time taken for a given number of cases to be added to the heap and sorted into the correct place

Secondly the process of removing the next case in the queue would have the same time complexity as the complexity of removing the case would be O(1) and the sorting of the elements would take O(n), just like the adding of elements. Figure 2.2 shows this trend with the removal of 1000 elements taking aproximatly 0.018 seconds (18ms).



The number of elements

The time taken to remove a given number of elements (seconds)

Figure 2.2: The time taken for a given number of elements to be removed from the heap.

# References

Baker, C., 2020. *NHS Key Statistics. England, February 2020*. [ebook] pp.4, 5. Available at: <https://researchbriefings.files.parliament.uk/documents/CBP-7281/CBP-7281.pdf> [Accessed 25 April 2020].

The King's Fund. 2020. *What’S Going On With A&E Waiting Times?*. [online] Available at: <https://www.kingsfund.org.uk/projects/urgent-emergency-care/urgent-and-emergency-care-mythbusters> [Accessed 25 April 2020].

# Appendix A