**Project Report**

**Group members:**

* **Behjet Ansari (ba04079)**
* **Laiba Fatima Khan (lk04067)**

**Instructor: Dr. Shahid Hussain**

**Course: Data Structures II**

**Date: 29th April, 2019.**

**Requirement:**

Use two different data structures to solve a real-world problem and compare the efficiency of data structures.

**Proposal:**

We chose a practical business-related example of patients being tackled in a hospital according to their injury levels. We are going to implement the working and solution using ***Priority Queue*** and ***Array-based List*** data structures. The scenario below further explains our idea in detail.

**Scenario:**

Patients are coming in at a steady rate in the hospital and there is only one doctor available. There are patients of various levels of injuries so to ensure optimum care for every patient, they are prioritized according to their injury levels. For example the first patient who walks in is served immediately. A person with a pneumonia comes in next and requires a check-up. You add him to the queue and he waits in line for the doctor to become available. Next, a man with a severe injury comes through the door. He is assigned a higher priority because he has a higher medical liability. So the man with the cold is bumped down in line. Similarly, as more patients stand in the queue, their priority is decided and they shift places in the queue accordingly.

**Approach:**

We have designed our program with priorities 0 to 4, 0 being the highest injury level, 4 being the lowest.

The data we enter is sorted, as patients are given increasing ID numbers as they come. For the Array-based List, we have 5 lists nested inside a list; the index of each representing priorities/injury levels. As patients come, they will be added into their respective nested list according to their priority.

**Theoretical Analysis:**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Priority\_Queue** | **Array\_Based List** |
| **Add()** | **O(n)** | **O(1)** |
| **Remove()** | **O(1)** | **O(1)** |

**Empirical Analysis On Machine 1:**

**Built-in Priority Queue Implementation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Add Inputs** | **Remove Inputs** | **Add() Time**  **m-s** | **Remove() Time**  **m-s** |
| **100** | **5** | **0** | **0** |
| **1000** | **10** | **4** | **0** |
| **5000** | **50** | **24** | **0** |
| **10000** | **100** | **56** | **0** |
| **50000** | **500** | **260** | **0** |
| **100000** | **1000** | **528** | **0** |

**Array Based List Implementation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Add Inputs** | **Remove Inputs** | **Add() Time**  **m-s** | **Remove() Time**  **m-s** |
| **100** | **5** | **0** | **0** |
| **1000** | **10** | **8** | **0** |
| **5000** | **50** | **92** | **0** |
| **10000** | **100** | **340** | **0** |
| **50000** | **500** | **8516** | **4** |
| **100000** | **1000** | **35356** | **16** |

**Empirical Analysis On Machine 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Add Inputs** | **Remove Inputs** | **Add() Time**  **m-s** | **Remove() Time**  **m-s** |
| **100** | **5** | **0** | **0** |
| **500** | **10** | **1** | **0** |
| **1000** | **50** | **4** | **0** |
| **5000** | **100** | **16** | **0** |
| **10,000** | **500** | **36** | **1** |
| **50,000** | **1000** | **166** | **0** |

**More Efficient of the two:**

**Why was it more efficient?**