Section: CPS3410

Group Member: Hu Hanxing, Fu Dongfang

IDs: 987793, 987757

Title: Project Document

**Instruction**

This is a project implementing priority queue in two different algorithm. Priority queue is to arrange a queue according to the priority of each element in queue. “Many computer algorithms have embedded in them a sub-algorithm called a priority queue which produces on demand an element of extreme priority among elements in the queue” (Johnson, 1981). The older version of priority queue is to sort the elements in an array according to the priority of each element. The new version is to store the elements into a minimum heap. Elements with lower priority will be put at the front positions of the queue.

**Pseudo-code**

1. **Old Version:**

**Insert**{

Check whether array need to be expended;

Insert value at bottom of array;

Use bubble sort to shift forward;

Size++;

}

**Delete\_Min**{

Check Empty;

Save queue[0]

Shift queue to queue[size-1];

Clear the last one

Size++;

Return queue[0]

}

**FindMin**{

Check Empty;

Return queue[0];

}

**isEmpty**{

Return size==0;

}

1. **New Version:**

**Insert**{

Check whether array need to be expended;

Insert value at bottom of array;

Adjust Heap;

Size++;

}

**Delete\_Min**{

Check Empty;

Swap queue[1] and queue[size];

Queue[size]=0;

Size--;

Adjust Heap;

}

**FindMin**{

Check Empty;

Return queue[1];

}

**isEmpty**{

return size <= 0;

}

**Evaluation**

**Big-O Summary Table**

|  |  |  |
| --- | --- | --- |
| Function | Time Complexity (Old) | Time Complexity (New) |
| insert | O (N) |  |
| delete\_min | O (N) |  |
| isEmpty | O (1) |  |
| findMin | O (1) |  |
| deleteMin | O (N) |  |

**Old Version:**

when inserting a new value, it will place the new one at the end of the array. Then, it will be bubble proper place. Since it need to be swapped at most N time, the time complexity will be . When deleting the minimum value, it will also shift all the rest value forward. Therefore, the time complexity of deleting will be as well.

**New Version:**

Every time when inserting, it will put the new value at the bottom of the array. Then it will adjust the heap to build minimum heap. Each time adjusting the heap, the time complexity will be . Each time when deleting the minimum value, it will also adjust the heap.

**Result**

For testing, 100,000 numbers are inserted in the both priority queues. Then delete 100,000 times. To compare the time complexity more obviously, the running time is printed. The result as following:

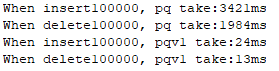


Fig. 1. The test result of comparing test (pq: old version, pqv1: new version)

**Summary**

From the Big-O table, we can find that the new version implemented by heap has lower time complexity. In old version, all the elements stored in array are sorted. Each time performing insertion and deletion, it will need to shift the elements in the array. In worst case, it will shift all the elements, which will take O(N) time. In new version, elements are stored in heap. When inserting and deleting elements from queue, it will adjust the heap. Adjusting heap do not need to take too much time. The time complexity is . Therefore, the new version can have better performance than the old version.

**References**

**[1]** Johnson, D. B. (1981). A priority queue in which initialization and queue operations take O(loglogD) time. Mathematical Systems Theory, 15(1), 295–309. <https://doi.org/10.1007/BF01786986>

This article illustrated the importance of priority queue and the way to implement this data structure.