# Report

**1-)**

***decreaseKey(index, key)***

| | |
|---|---|
| heap[index]-key <- key | O(1) |
| while heap[parent]-key > heap[child] – key do | O(lgn) |
|     swap(heap[child], heap[parent]) | O(5) |
|     index <- (index -1)/2 //assigning parents index to the “index” | O(1) |

While loop will operate as many as depth of the heap so asymptotic upper bound is O(lgn)

Total bound is O(lgn) + O( 7)

Since O(lgn) is dominant term, running time of this function is O(lgn)

***extract(index)***

| | |
|---|---|
| if heap-size < 1 | O(1) |
|     then error “heap underflow” | O(1) |
| if index != 0 | O(1) |
|     decreaseKey(index, -∞) | O(lgn) |
| temporary <- heap[0] | O(1) |
| swap (heap[0], heap[heap-size - 1] ) | O(5) |
| heap.pop-back(0) | O(5) |
| return temporary | O(1) |

Since there is no loop, sum is O(lgn) + O(15)

Since O(lgn) is dominant term, running time of this function is O(lgn)

***insertVehicle(input)***

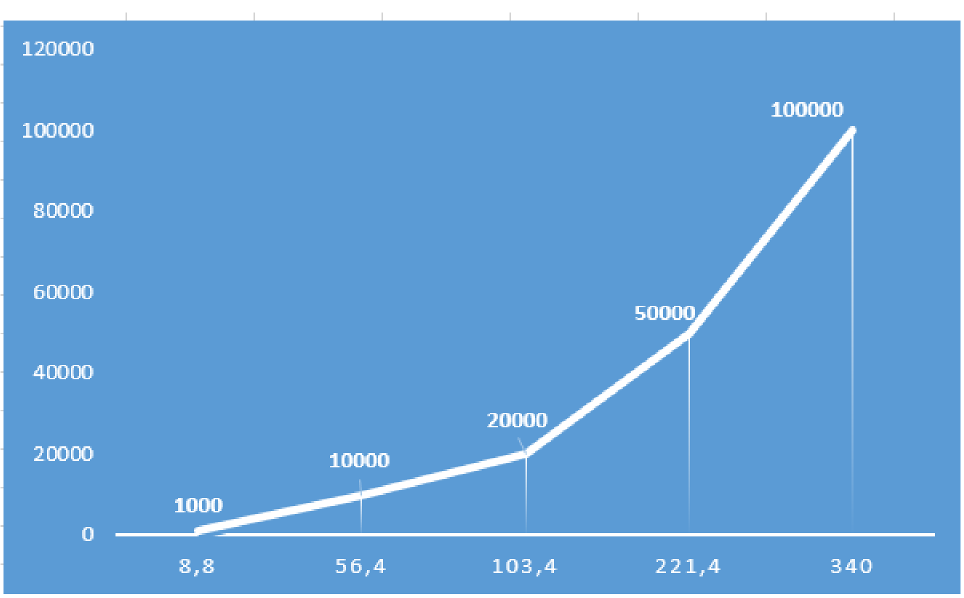
| | |
|---|---|
| temporary_key <- input – key | O(1) |
| input-key <- +∞ | O(1) |
| heap-push_back(input) | O(5) |
| decreaseKey(heap-size - 1, temporary_key) | O(lgn) |

Since there is no loop, sum is O(lgn) + O(7)

Since O(lgn) is dominant term, running time of this function is O(lgn)

**2-) Table is in milliseconds**

| | 1 | 2 | 3 | 4 | 5 | Avg. |
|---|---|---|---|---|---|---|
| **1000** | **10** | **8** | **6** | **10** | **10** | **8.8** |
| **10K** | **44** | **58** | **64** | **56** | **60** | **56.4** |
| **20K** | **67** | **110** | **119** | **106** | **115** | **103.4** |
| **50K** | **254** | **263** | **271** | **163** | **156** | **221.4** |
| **100K** | **487** | **301** | **311** | **301** | **300** | **340** |




decreaseKey function has O(lgn) asymptotic upper bond as well as extract and insert functions.

In total O(lgn) + O(lgn) + O(lgn) = O(3lgn). While lgn is dominant term, it will be O(lgn) anyway. When we observe the plot table above, we can see nlgn function. The reason for that both decrease, insert and extract functions are repeated n times due to n number of data in the heap. So we must multiply it for lgn to get the final runtime for overall algorithm.

Ramal Seyidli

150180901