**Assignment**

1.

(b)

*I followed different path (Pseudo code) than it was shown in the document but still the algorithm works correctly.*

Before sorting

[0,074112, 0,501269, 0,862014, 0,023933, 0,052857, 0,213698, 0,624462, 0,937675, 0,412764, 0,256883]

After sorting

[0,023933, 0,052857, 0,074112, 0,213698, 0,256883, 0,412764, 0,501269, 0,624462, 0,862014, 0,937675]

(c)

|  |  |  |  |
| --- | --- | --- | --- |
| Input size | Bucket sort running time | Merge sort running time | Heap sort running time |
| 10 | 15700 | 231400 | 9900 |
| 100 | 100500 | 70000 | 21400 |
| 1000 | 755400 | 727900 | 158300 |
| 10000 | 4165800 | 1506700 | 582900 |
| 100000 | 10483500 | 7873400 | 3931900 |
| 1000000 | 53882800 | 166318800 | 3298700 |
| 100000000 | 851021900 | 727390700 | 22726900 |

Which algorithm performs best at which input size?

*I highlighted the ones that work better. As we can see that heap sort works faster compared to other sorting algorithms.*

(d)

|  |  |  |  |
| --- | --- | --- | --- |
| Input size | Bucket sort RAM | Merge sort RAM | Heap sort RAM |
| 10000000 | 548 | 345 | 162 |

*Bucket sort uses more RAM compared to other sorting algorithms since the space complexity of bucket sort is O(n+k) (n is input size & k is the number of buckets.).So, the used memory gets bigger when the input size increases. In other words, we need to have more buckets when the input size is higher.*