(b)

[87, 54, 93, 33, 38, 19, 48, 74, 87, 93]

[19, 33, 38, 48, 54, 74, 87, 87, 93, 93]

(c) *I added a row for “10000000” since it was not included in the table.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input size | Radix sort running time | Counting sort running time | Merge sort running time | Heap sort running time |
| 10 | 5500 | 2000 | 200800 | 10000 |
| 100 | 9100 | 7700 | 63400 | 77500 |
| 1000 | 84500 | 112500 | 572500 | 184400 |
| 10000 | 836000 | 929300 | 2043800 | 1274700 |
| 100000 | 3003200 | 2014500 | 13382600 | 14203600 |
| 1000000 | 32303000 | 23015300 | 143632800 | 143786000 |
| 10000000 | 477638300 | 422365500 | 1347815400 | 2150487600 |
| 100000000 | 8391754100 | 5031481200 | 15948088700 | 39604883600 |

Which algorithm performs best at which input size?

*I highlighted the algorithms that works better.*

(d)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input size | Input range | Radix sort RAM | Counting sort RAM | Merge sort RAM | Heap sort RAM |
| 100000000 | 100-999 | 109 | 190 | 220 | 170 |

Merge sort uses more ram since it needs to copy the given array into 2 subarrays.