Problem 1: Priority Queues

If we are to define maximum as being the highest value letter (so ones closest to Z on the alphabet), then:

M	// M
Y M	// Y
Y P M	// P
Y R P M	// R
Y R P M I	// I
Y R P O M I	// O
R P O M I	// * - REMOVES Y
R R P O M I	// R
R P O M I	// * - REMOVES R
P O M I	// * - REMOVES R
P O M I I	// I
O M I I	// * - REMOVES P
T O M I I	// T
O M I I	// * - REMOVES T
Y O M I I	// Y
O M I I	// * - REMOVES Y
M I I	// * - REMOVES O
I I	// * - REMOVES M
Q I I	// Q
U Q I I	// U

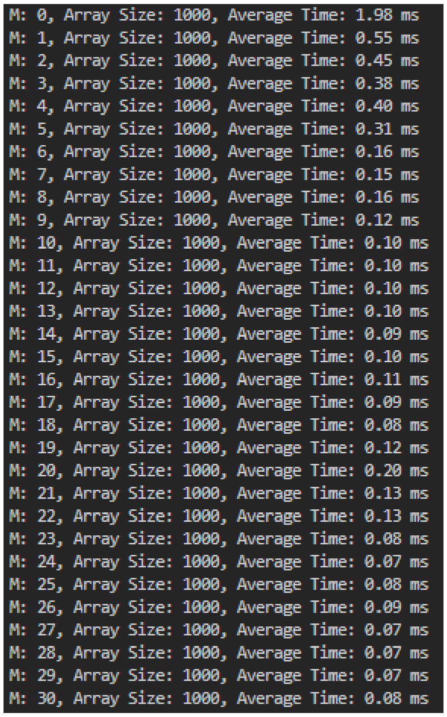
```
U Q I I E                      // E

Q I I E                        // * - REMOVES U

I I E                          // * - REMOVES Q

I E                            // * - REMOVES I

U I E                          // U

I E                            // * - REMOVES U

I E E                          // E

E E                            // * - REMOVES I
```

Sequence of letters by removal:

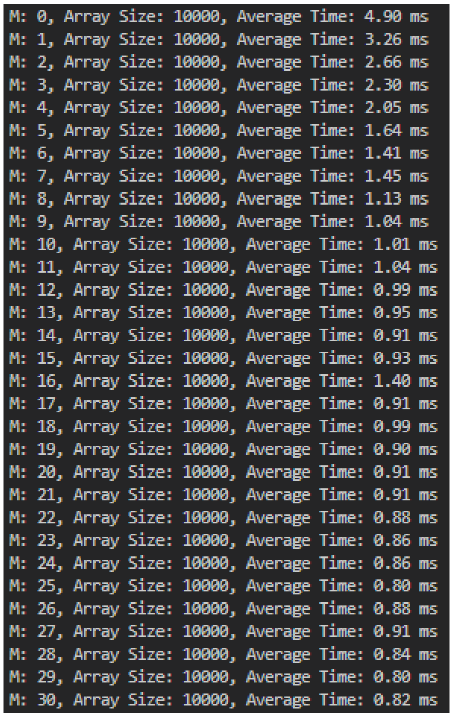
Y R R P T Y O M U Q I U I

Problem 2: 3-Way Quicksort Optimization

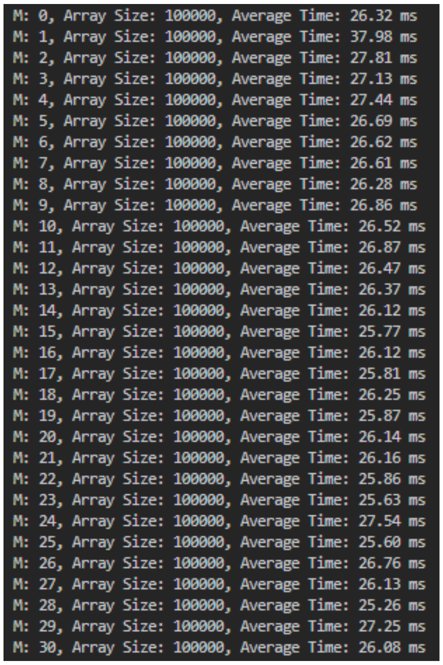
One of my example runs:



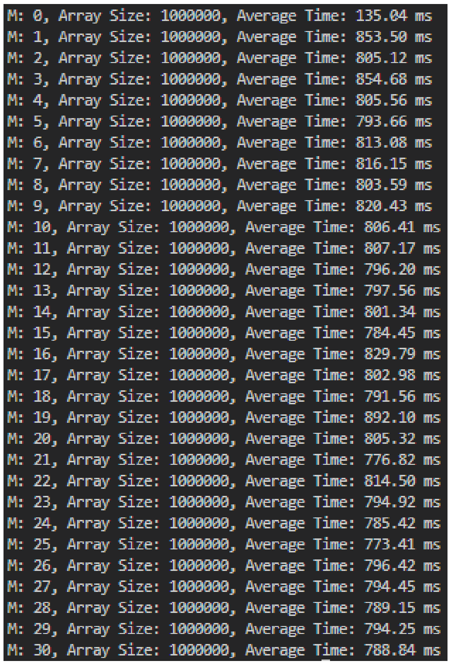
```
M: 0, Array Size: 1000, Average Time: 1.98 ms
M: 1, Array Size: 1000, Average Time: 0.55 ms
M: 2, Array Size: 1000, Average Time: 0.45 ms
M: 3, Array Size: 1000, Average Time: 0.38 ms
M: 4, Array Size: 1000, Average Time: 0.40 ms
M: 5, Array Size: 1000, Average Time: 0.31 ms
M: 6, Array Size: 1000, Average Time: 0.16 ms
M: 7, Array Size: 1000, Average Time: 0.15 ms
M: 8, Array Size: 1000, Average Time: 0.16 ms
M: 9, Array Size: 1000, Average Time: 0.12 ms
M: 10, Array Size: 1000, Average Time: 0.10 ms
M: 11, Array Size: 1000, Average Time: 0.10 ms
M: 12, Array Size: 1000, Average Time: 0.10 ms
M: 13, Array Size: 1000, Average Time: 0.10 ms
M: 14, Array Size: 1000, Average Time: 0.09 ms
M: 15, Array Size: 1000, Average Time: 0.10 ms
M: 16, Array Size: 1000, Average Time: 0.11 ms
M: 17, Array Size: 1000, Average Time: 0.09 ms
M: 18, Array Size: 1000, Average Time: 0.08 ms
M: 19, Array Size: 1000, Average Time: 0.12 ms
M: 20, Array Size: 1000, Average Time: 0.20 ms
M: 21, Array Size: 1000, Average Time: 0.13 ms
M: 22, Array Size: 1000, Average Time: 0.13 ms
M: 23, Array Size: 1000, Average Time: 0.08 ms
M: 24, Array Size: 1000, Average Time: 0.07 ms
M: 25, Array Size: 1000, Average Time: 0.08 ms
M: 26, Array Size: 1000, Average Time: 0.09 ms
M: 27, Array Size: 1000, Average Time: 0.07 ms
M: 28, Array Size: 1000, Average Time: 0.07 ms
M: 29, Array Size: 1000, Average Time: 0.07 ms
M: 30, Array Size: 1000, Average Time: 0.08 ms
```



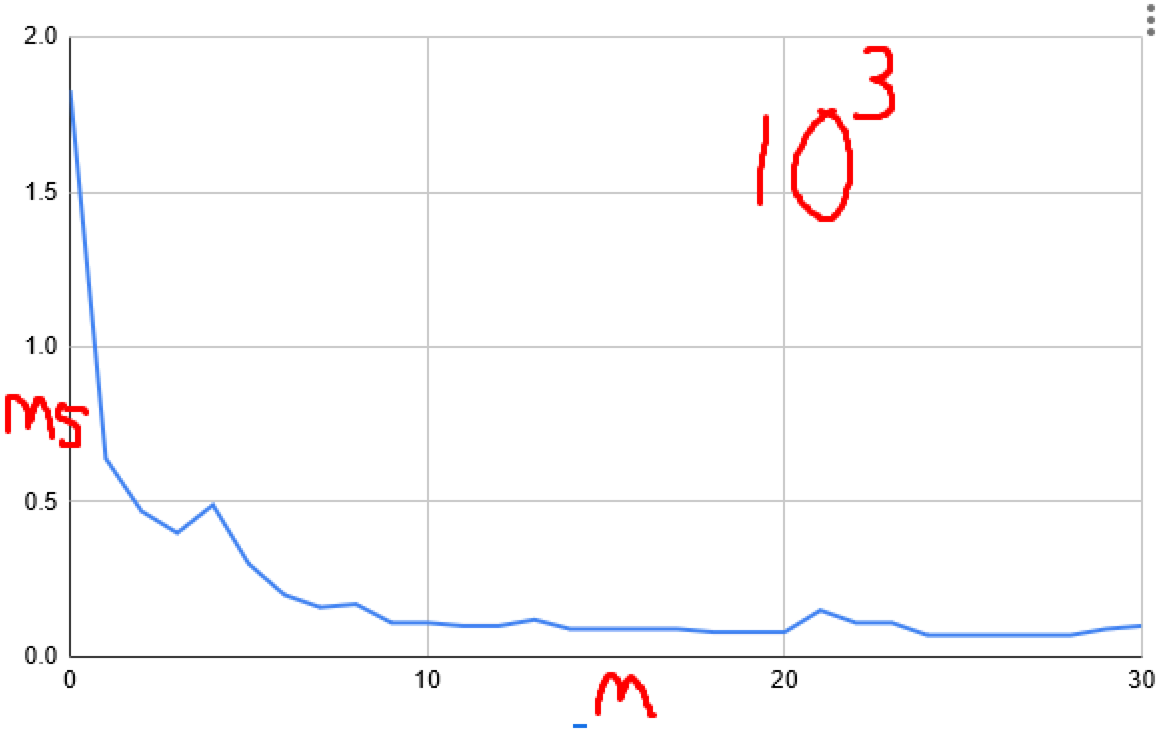
```
M: 0, Array Size: 10000, Average Time: 4.90 ms
M: 1, Array Size: 10000, Average Time: 3.26 ms
M: 2, Array Size: 10000, Average Time: 2.66 ms
M: 3, Array Size: 10000, Average Time: 2.30 ms
M: 4, Array Size: 10000, Average Time: 2.05 ms
M: 5, Array Size: 10000, Average Time: 1.64 ms
M: 6, Array Size: 10000, Average Time: 1.41 ms
M: 7, Array Size: 10000, Average Time: 1.45 ms
M: 8, Array Size: 10000, Average Time: 1.13 ms
M: 9, Array Size: 10000, Average Time: 1.04 ms
M: 10, Array Size: 10000, Average Time: 1.01 ms
M: 11, Array Size: 10000, Average Time: 1.04 ms
M: 12, Array Size: 10000, Average Time: 0.99 ms
M: 13, Array Size: 10000, Average Time: 0.95 ms
M: 14, Array Size: 10000, Average Time: 0.91 ms
M: 15, Array Size: 10000, Average Time: 0.93 ms
M: 16, Array Size: 10000, Average Time: 1.40 ms
M: 17, Array Size: 10000, Average Time: 0.91 ms
M: 18, Array Size: 10000, Average Time: 0.99 ms
M: 19, Array Size: 10000, Average Time: 0.90 ms
M: 20, Array Size: 10000, Average Time: 0.91 ms
M: 21, Array Size: 10000, Average Time: 0.91 ms
M: 22, Array Size: 10000, Average Time: 0.88 ms
M: 23, Array Size: 10000, Average Time: 0.86 ms
M: 24, Array Size: 10000, Average Time: 0.86 ms
M: 25, Array Size: 10000, Average Time: 0.80 ms
M: 26, Array Size: 10000, Average Time: 0.88 ms
M: 27, Array Size: 10000, Average Time: 0.91 ms
M: 28, Array Size: 10000, Average Time: 0.84 ms
M: 29, Array Size: 10000, Average Time: 0.80 ms
M: 30, Array Size: 10000, Average Time: 0.82 ms
```



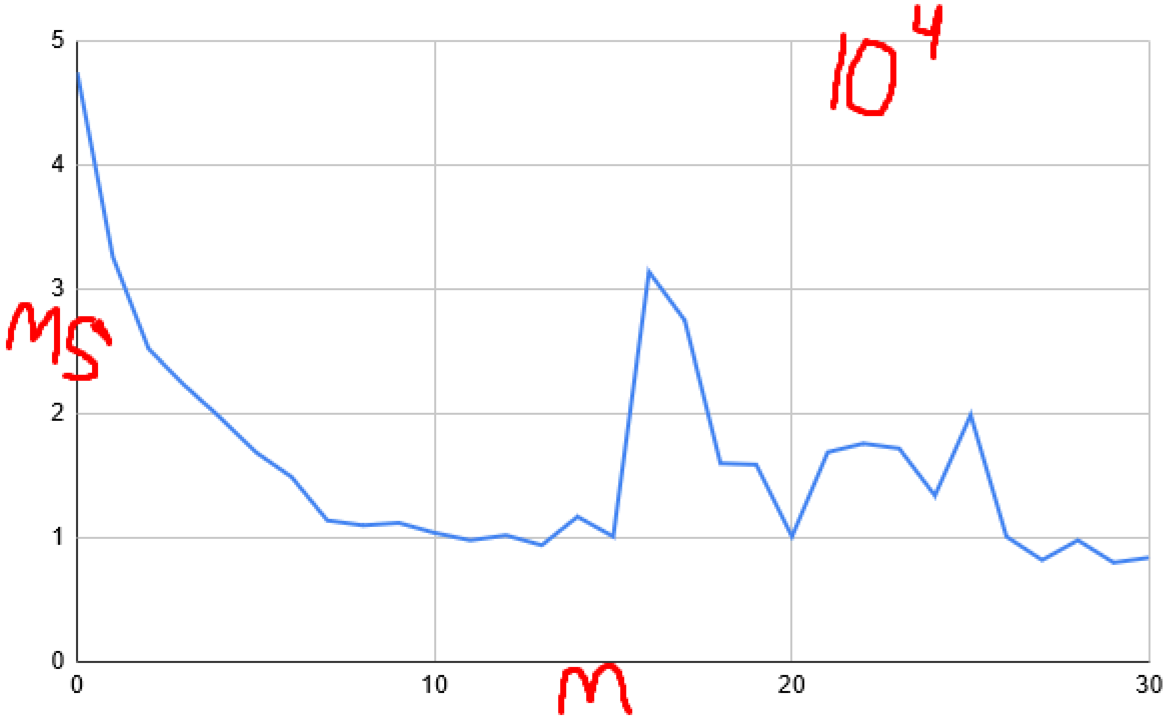
```
M: 0, Array Size: 100000, Average Time: 26.32 ms
M: 1, Array Size: 100000, Average Time: 37.98 ms
M: 2, Array Size: 100000, Average Time: 27.81 ms
M: 3, Array Size: 100000, Average Time: 27.13 ms
M: 4, Array Size: 100000, Average Time: 27.44 ms
M: 5, Array Size: 100000, Average Time: 26.69 ms
M: 6, Array Size: 100000, Average Time: 26.62 ms
M: 7, Array Size: 100000, Average Time: 26.61 ms
M: 8, Array Size: 100000, Average Time: 26.28 ms
M: 9, Array Size: 100000, Average Time: 26.86 ms
M: 10, Array Size: 100000, Average Time: 26.52 ms
M: 11, Array Size: 100000, Average Time: 26.87 ms
M: 12, Array Size: 100000, Average Time: 26.47 ms
M: 13, Array Size: 100000, Average Time: 26.37 ms
M: 14, Array Size: 100000, Average Time: 26.12 ms
M: 15, Array Size: 100000, Average Time: 25.77 ms
M: 16, Array Size: 100000, Average Time: 26.12 ms
M: 17, Array Size: 100000, Average Time: 25.81 ms
M: 18, Array Size: 100000, Average Time: 26.25 ms
M: 19, Array Size: 100000, Average Time: 25.87 ms
M: 20, Array Size: 100000, Average Time: 26.14 ms
M: 21, Array Size: 100000, Average Time: 26.16 ms
M: 22, Array Size: 100000, Average Time: 25.86 ms
M: 23, Array Size: 100000, Average Time: 25.63 ms
M: 24, Array Size: 100000, Average Time: 27.54 ms
M: 25, Array Size: 100000, Average Time: 25.60 ms
M: 26, Array Size: 100000, Average Time: 26.76 ms
M: 27, Array Size: 100000, Average Time: 26.13 ms
M: 28, Array Size: 100000, Average Time: 25.26 ms
M: 29, Array Size: 100000, Average Time: 27.25 ms
M: 30, Array Size: 100000, Average Time: 26.08 ms
```



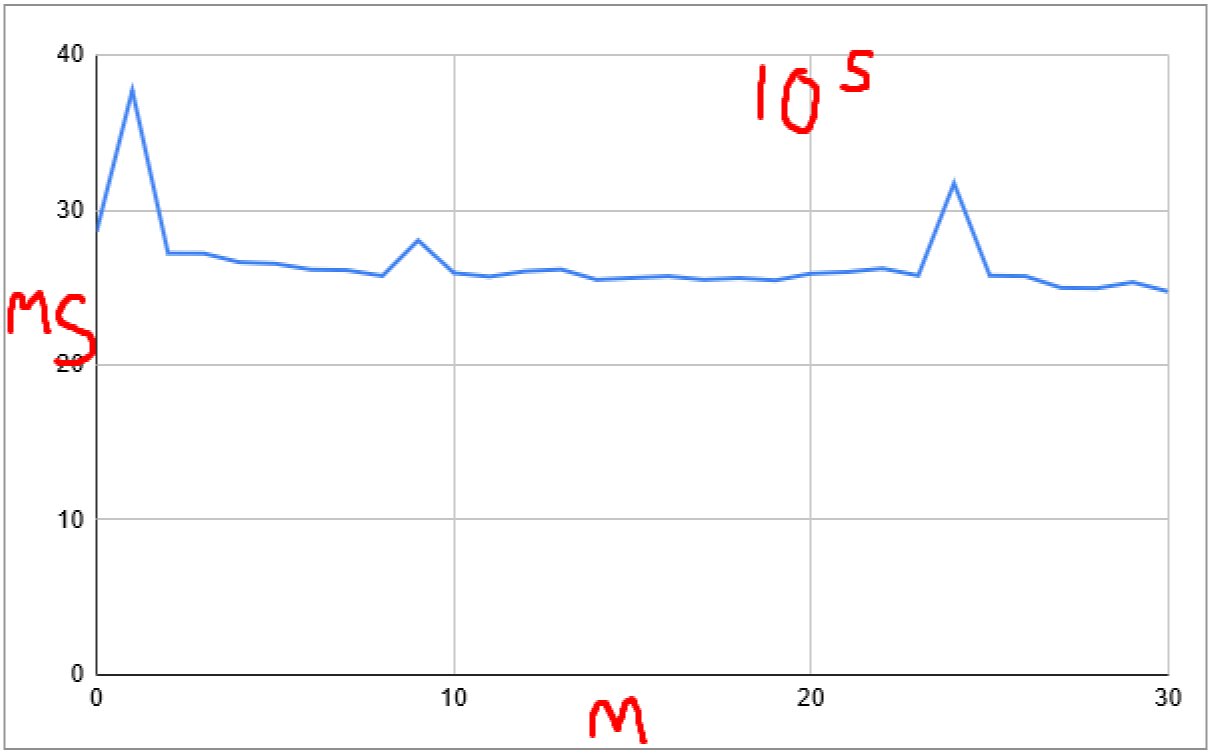
```
M: 0, Array Size: 1000000, Average Time: 135.04 ms
M: 1, Array Size: 1000000, Average Time: 853.50 ms
M: 2, Array Size: 1000000, Average Time: 805.12 ms
M: 3, Array Size: 1000000, Average Time: 854.68 ms
M: 4, Array Size: 1000000, Average Time: 805.56 ms
M: 5, Array Size: 1000000, Average Time: 793.66 ms
M: 6, Array Size: 1000000, Average Time: 813.08 ms
M: 7, Array Size: 1000000, Average Time: 816.15 ms
M: 8, Array Size: 1000000, Average Time: 803.59 ms
M: 9, Array Size: 1000000, Average Time: 820.43 ms
M: 10, Array Size: 1000000, Average Time: 806.41 ms
M: 11, Array Size: 1000000, Average Time: 807.17 ms
M: 12, Array Size: 1000000, Average Time: 796.20 ms
M: 13, Array Size: 1000000, Average Time: 797.56 ms
M: 14, Array Size: 1000000, Average Time: 801.34 ms
M: 15, Array Size: 1000000, Average Time: 784.45 ms
M: 16, Array Size: 1000000, Average Time: 829.79 ms
M: 17, Array Size: 1000000, Average Time: 802.98 ms
M: 18, Array Size: 1000000, Average Time: 791.56 ms
M: 19, Array Size: 1000000, Average Time: 892.10 ms
M: 20, Array Size: 1000000, Average Time: 805.32 ms
M: 21, Array Size: 1000000, Average Time: 776.82 ms
M: 22, Array Size: 1000000, Average Time: 814.50 ms
M: 23, Array Size: 1000000, Average Time: 794.92 ms
M: 24, Array Size: 1000000, Average Time: 785.42 ms
M: 25, Array Size: 1000000, Average Time: 773.41 ms
M: 26, Array Size: 1000000, Average Time: 796.42 ms
M: 27, Array Size: 1000000, Average Time: 794.45 ms
M: 28, Array Size: 1000000, Average Time: 789.15 ms
M: 29, Array Size: 1000000, Average Time: 794.25 ms
M: 30, Array Size: 1000000, Average Time: 788.84 ms
```

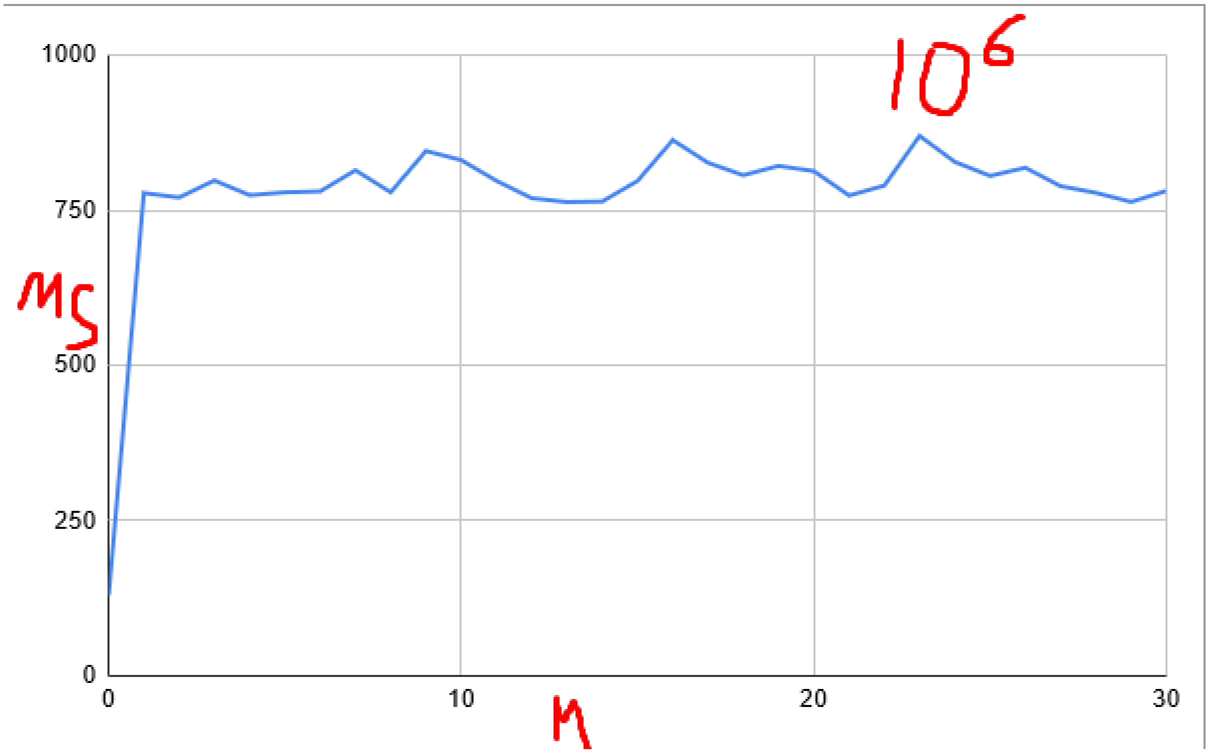


10^3
2.0
1.5
1.0
ms
0.5
0.0
0
10
20
30
m



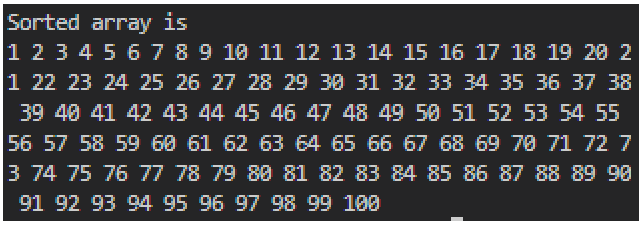
10^4
5
4
3
ms
2
1
0
0
10
20
30
m





Evidently, M=0 is very good for arrays fo size 10^6 as it selects insertion sort. Overall, M=30 was better for other array sizes.

Problem 3: Ternary Heapsort



```
Sorted array is
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 2
1 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 7
3 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
 91 92 93 94 95 96 97 98 99 100
```