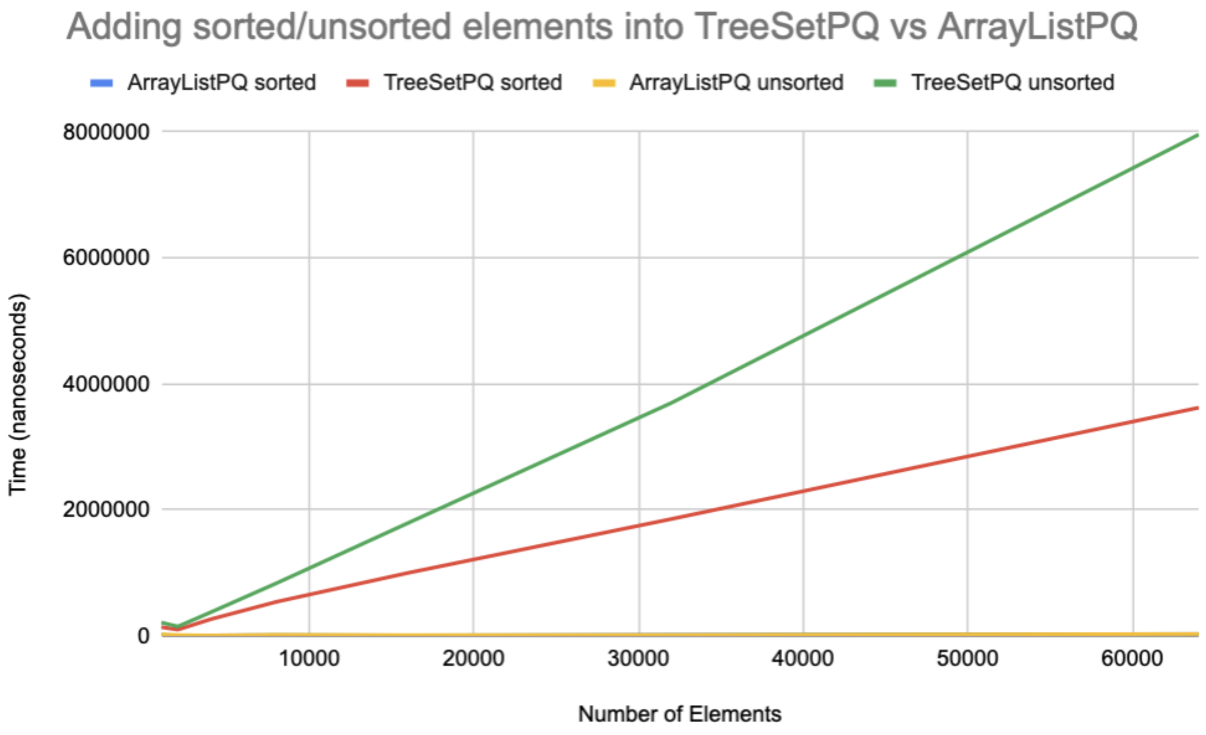
Aiden Pratt and Melanie Prettyman

1. First, time your data structure "creation". For varying sizes of N, how long does it take to build you data structure? For the tree based data structure, just call add on each element, for the heap version, use the constructor taking an arraylist

This data shows that the TreeSetPQ for inserting both sorted and unsorted elements take significantly longer than the ArrayListPQ. Adding the elements to the TreeSet is O(NlogN) time complexity. Adding the elements to the ArrayList is O(N) time complexity.

N	ArrayListPQ sorted	TreeSetPQ sorted	ArrayListPQ unsorted	TreeSetPQ unsorted
1000	14887	138524	22195	211693
2000	4677	98399	12344	148551
4000	7859	264546	7218	375143
8000	12756	541550	16879	837392
16000	7367	1000923	11256	1794280
32000	17024	1856855	14750	3702031
64000	29495	3623106	30135	7959152

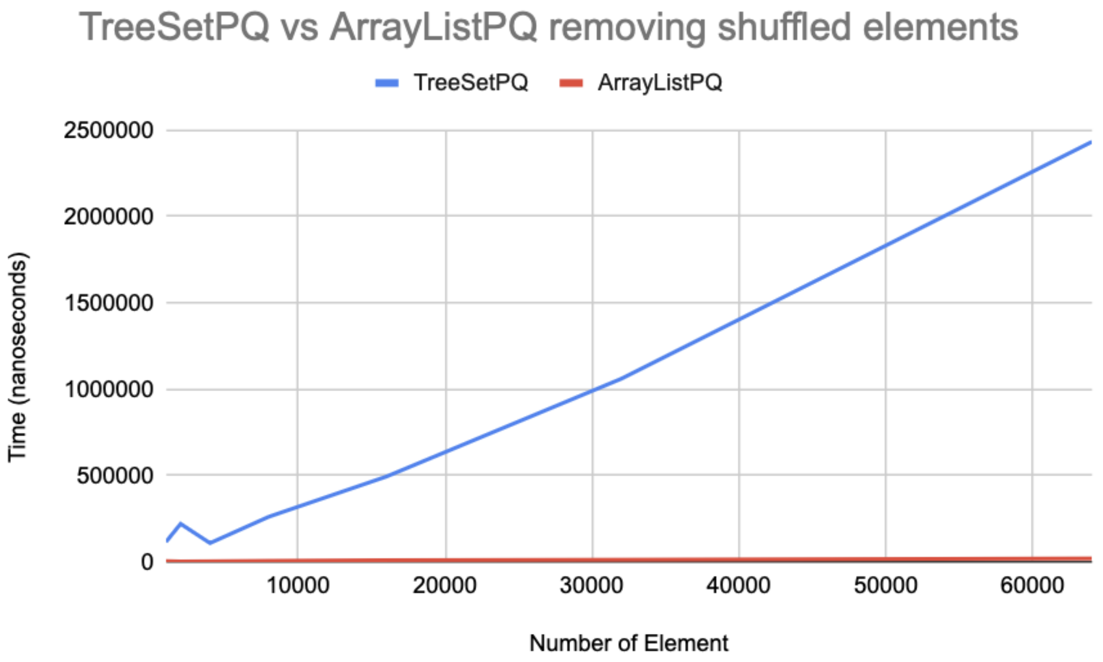


2. Second, time the following loop with each data structures:

```
while(!pqueue.isEmpty()){ pqueue.removeMin(); }
```

Our data for the removeMin method shows that the TreeSetPQ takes significantly more time than the ArrayListPQ to remove the minimum method. This is because the TreeSetPQ time complexity is O(NlogN) and the ArrayListPQ is O(N).

N	TreeSetPQ	ArrayListPQ
1000	114427	3398
2000	219472	1350
4000	107751	2392
8000	260424	4855
16000	493057	8805
32000	1060484	11353
64000	2432733	18630


TreeSetPQ vs ArrayListPQ removing shuffled elements
TreeSetPQ
ArrayListPQ
Time (nanoseconds)
Number of Element