Guillermo Rojas

Professor Ali Kooshesh

CS - 315

November 14, 2025

Batch-and-drain analysis

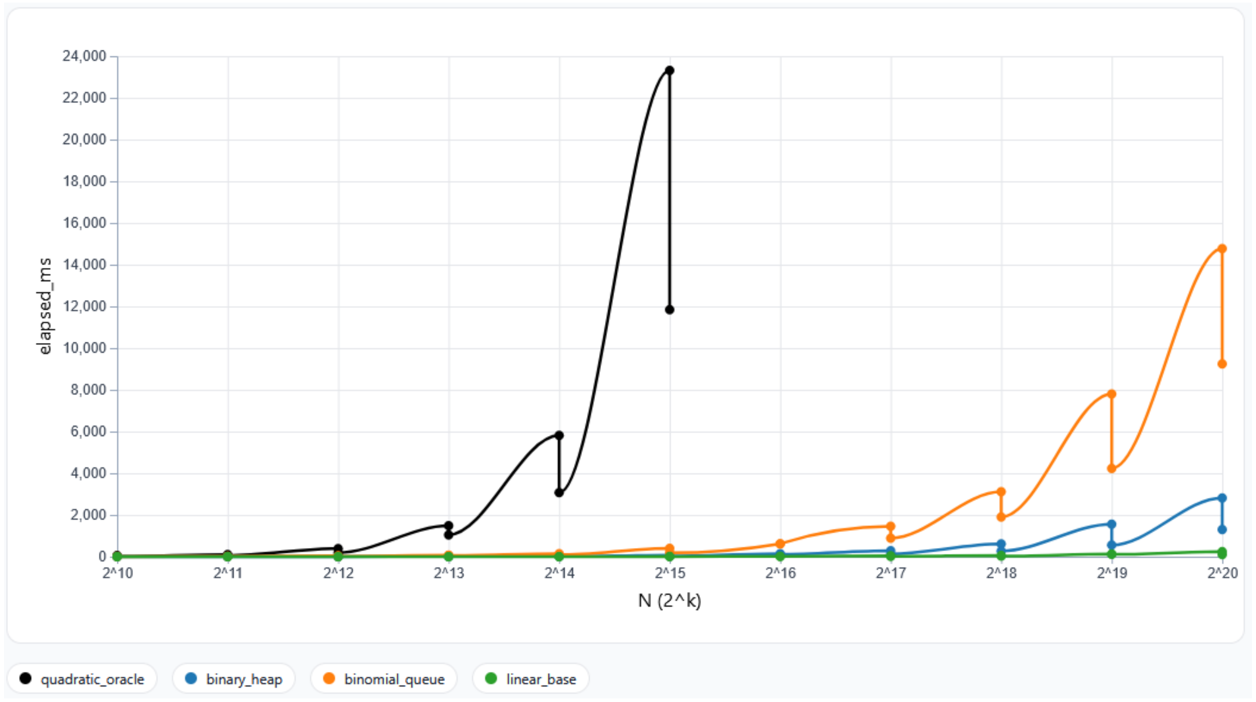
My question when going into this project was how a batch-then-drain pattern work and does this type of all insertions before any extractions affect the performance of the priority queue? I know this was different than Huffman encoding where you keep switching from inserting to extracting which does not seem efficient. My hypothesis was that binary heaps should be better off in this setup because once you build the heap, you're just pulling the smallest element until you are left with nothing. For binomial queues I can see it be a little more efficient, but it would be slow because they must keep checking different trees to find the min which doesn't seem efficient. As for the sorted list, it should in theory be way faster because it only must sort once and then just pull the smallest which is the first item.

The way I tested was that I used the same setup as our Huffman coding project but changed the pattern. Instead of having to extract, extract, and insert I did all (N) inserts then all (N) extractions so only a total of 2N operations were used instead. I tested N from 13 all the way up to 20^2 and each test was ran 7 times and a median was taken. The same seed of 23 was used to have a fair comparison between the algorithms. Binary heap, binomial queue, a sorted array, and a quadratic oracle was used to test them.

I found that on average all structures got faster and some got even faster. The binary heap sped up by about 2.3x more and N = 1million The binomial queue was about

1.6x faster but still slower than the heap because it is not optimized for taking bulk inserts.

The linear baseline looks like it is extremely fast and pulling the first element in the list is fast with the linear baseline but to keep that list sorted it had to do a lot of work during the insertion phase because it constantly had to sort it but since we are only looking at extraction then it appears faster. Whereas the heap is just better in every way. It matters because it's not just about big O notation its how useful it would be in real world applications. Binomial queues are better when you're mixing inserts and extracts constantly


elapsed_ms
2^10
2^11
2^12
2^13
2^14
2^15
2^16
2^17
2^18
2^19
2^20
N (2^k)
quadratic_oracle
binary_heap
binomial_queue
linear_base