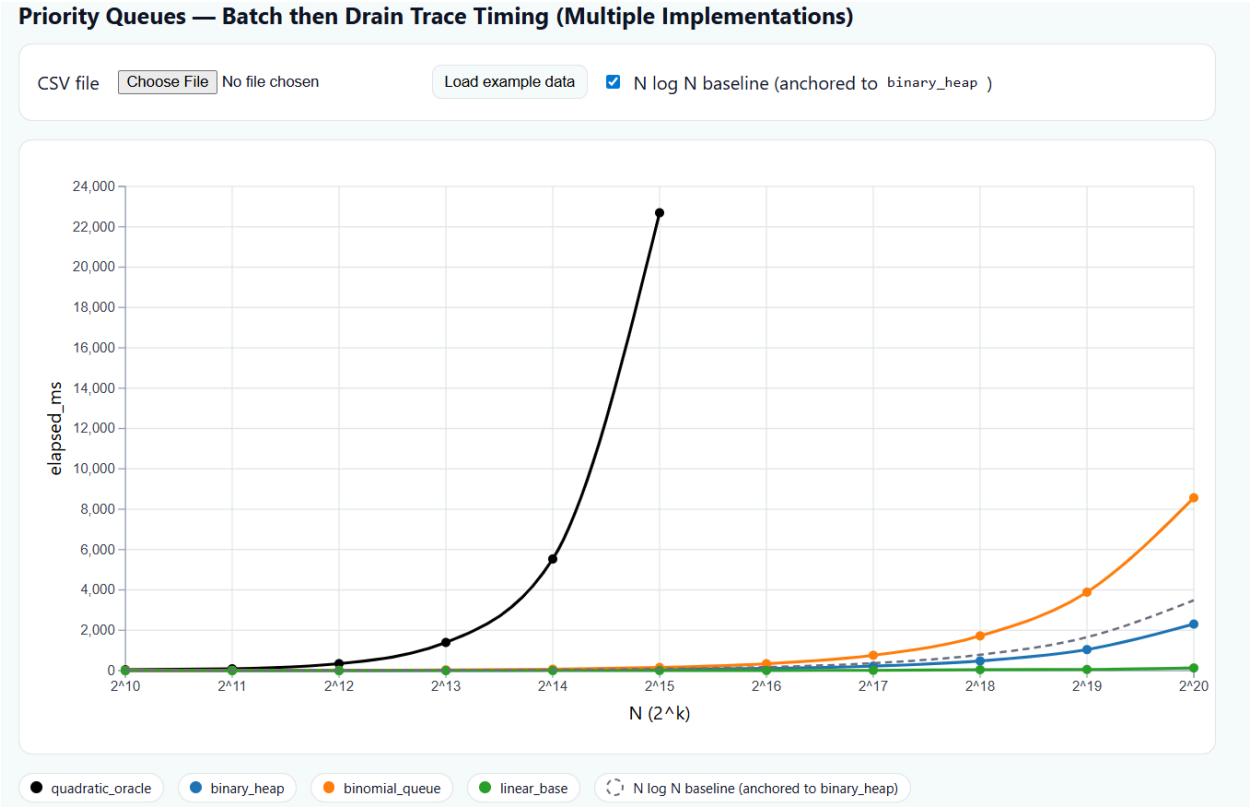
Jaime Yah-Perez

11-16-2025

CS 315

Regarding the implementation of the batch then drain profile, I had thought about how each implementation would perform and how it would compare to the data gathered from the huffman profile. I expected for this scenario for there to be a significant change in the rankings of the data structures, as would an implementation like the binary heap rank best amongst the implementations? Leading me to form the hypothesis that structures with a lower cost for deletion and finding would perform the best, such as with linear base and binary heap implementations.

To test out this hypothesis, a new trace profile was created modeled after the batch then drain method, which is an offline sort where assumption that all values to be inserted will be known prior to sorting, and therefore can all be inserted at once then continuously drained after for a profile of N inserts and N * (find, delete). The range of keys for distribution of keys for this profile was set to be between 1 and 2^20, with the id increasing incrementally, and the random seed utilized was kept the same across runs. Once the trace generator set up these distributions in trace files, the harness would then read from these files and deliver them to preprepared implementations of the binary heap, binomial queue, linear base, and quadratic oracle data structures. In the case of the quadratic oracle, N is only ran up to 2^16 due to a faster O() increase. And results of the trials are then written into a csv, with recording of time elapsed in ms, compared to a range of N = (2 ^ k).



The results of trials have displayed some immediate details, such as the quadratic oracle having a rapid increase in time performance after N = 2^12, with the binomial queue beginning to lag behind after 2 ^ 16. After 2 ^ 17 the gap between binary heap and linear base also begins to widen, which leaves linear base as the best performing from that point and onwards.

Based off the data, I am able to interpret the O() of each of the implementations. That being O(N^2) for quadratic oracle, approximately O(NlogN) for both binomial queue and binary heap, and what looks like O(N) for the linear base. The difference between binomial queue and binary heap may be due to the way they perform their delete min and restructuring afterwards, as it may be assumed that restructuring a binomial queue takes longer to perform than restructuring a binary heap. The linear base performing best for this profile makes sense as this profile is deletion heavy, and a sorted vector would allow for a O(1) for find, and O(N) for deletion. This combined with the binary heap keeping up for large N trials leads me to believe that my hypothesis is correct for this profile.