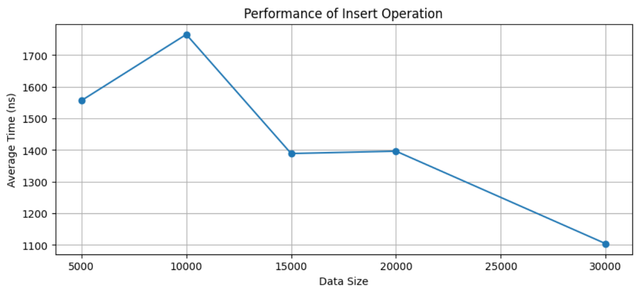
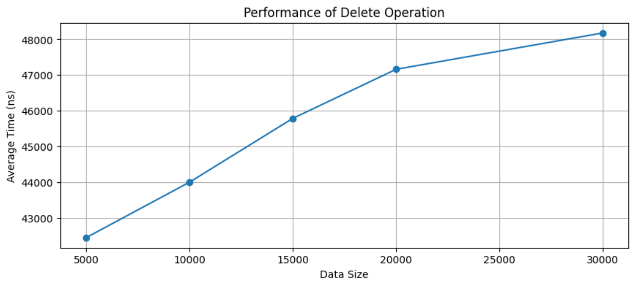
Performance analysis of Treaps

Pratyush Sethi 31619970

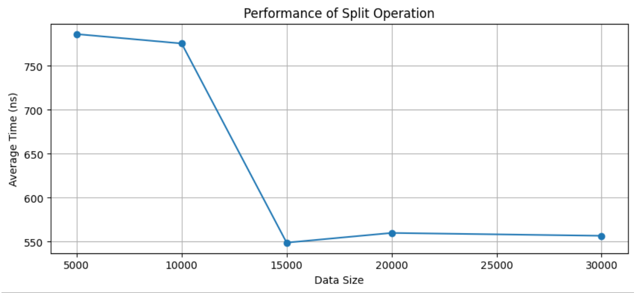
The implementation of treaps is done on 5000, 10000, 15000, 20000, 30000 keys over 1000 trials. The test was performed 10 times and aggregate of the results (i.e measurement of time in nano seconds) was done for each operation that is insert, delete, split, join and find.

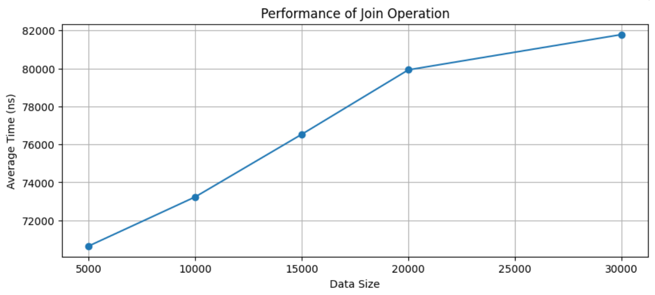
Observation: For ***insert*** opertion shows variability, initially increasing from 1556.48 ns to 1765.58 ns, then decreasing to around 1104.14 as the data grows. As for ***delete*** time approximately ranged from 42452.22 to 48177.75 ns which shows that increasing trend in tme as size of data increases.
Analysis: For ***insert*** the fluctuation could be attributed to the randomness inherent in treaps which affects the balance of the tree. Overall, the times suggest a reasonable efficiency in insertion operation. For ***delete*** the gradual increase increase suggest a near linear relationship between data size and time, which is consistent with the logarithmic avg time complexity of treap operations. Insert time fluctuate but do not show a clear exponential growth with the size data and same for delete. Expected complexity for treaps insert and delete is **O(log n)**, it doesnt align with insert due to randomness in the data but delete works exponentially.



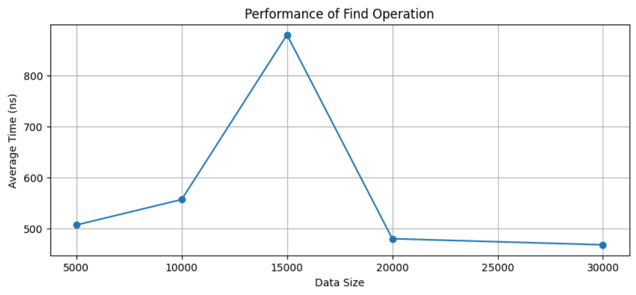


Observation: ***split*** operation times are relatively stable, ranging from 547 to 786 ns, whereas ***join*** operation which performed simultaneously after split takes the highest time from 70647 ns to increasing 81782 ns.
Analysis: The consistency and low duration of ***split*** operations indicate effective performance, but for ***join*** operation being more complex as it involves merging two treaps it takes longer time, which corresponds to the increasing size of tree and longer time take to join them. Split & Join operation in teraps have **O(log n)** time complexity, which can be seen in join operation but due to randomness and stability in data for split it shows somewhat linear complexity.





Observation: The ***find*** operation time is significantly lower compared to rest of the operations, ranging from 467 to 879 ns. Interestingly, the time decreases as the data size increases afte 15,000 keys.
Analysis: The relatively stable and low time indicates efficient search capability. The decrease in time with larger data sets could be due to more balanced tree structure as the number of elements increases. The expected time complexity remains **O(log n)** for find but observed performance indicate data might be well balanced and distributed in the tree.


Performance of Find Operation
Average Time (ns)
Data Size