Ankit Kumar **2021MMB1342**

Johnson's Algorithm with Various Heap Implementations

Objective:

The primary goal of this assignment was to implement Johnson's Algorithm for the All Pair Shortest Path problem using four different types of heaps: Array, Binary Heap, Binomial Heap, and Fibonacci Heap. This report delves into the analysis of the algorithm with these various heap structures and provides insights into their efficiency, both theoretically and empirically.

Theoretical Upper Bounds for Elementary Operations:

The table below outlines the upper bounds (in big-O notation) for elementary operations on the four types of heaps.

Operation	Array	Binary Heap	Binomial Heap	Fibonacci Heap
insert	O(1)	O(log N)	O(log N)	O(1)
find-min	O(N)	O(1)	O(log N)	O(1)
delete-min	O(N)	O(log N)	O(log N)	O(log N)
decrease-key	O(N)	O(log N)	O(log N)	O(1)
union	O(N)	O(N)	O(log N)	O(1)

Space Complexity:

In addition to time complexity, the space complexity of each heap type should be considered. Array heaps generally have a fixed size, leading to a straightforward space analysis. However, more dynamic structures like Fibonacci Heaps might have varying space complexities due to their nature of deferred consolidation.

Implementation Challenges:

Discuss challenges faced during the implementation of Johnson's Algorithm with each heap type. For instance, handling edge cases, optimizing data structures for specific operations, or managing memory efficiently.

Practical Performance:

Evaluate the practical performance of each heap implementation in the context of Johnson's Algorithm.
Trade-offs: Highlight trade-offs associated with each heap type. For example, while Fibonacci Heaps offer constant-time insertion and decrease key operations, they might have higher constant factors, making them less efficient for small datasets.

Adaptability: Discuss the adaptability of each heap type to different problem sizes and characteristics. Some heaps may excel in certain scenarios but perform suboptimally in others.

Comparison with Other Algorithms: Compare the efficiency of Johnson's Algorithm with different heap implementations against other algorithms for the All Pair Shortest Path problem. This could include algorithms like Floyd-Warshall or Bellman-Ford.

Real-world Applications: Explore real-world applications where the choice of heap data structure could impact the overall performance of algorithms for solving similar problems.

Execution Time

Number of Nodes	Array	Binary Heap	Binomial Heap	Fibonacci Heap
100	0.1	0.0	0.0	0.1
200	0.8	0.4	0.4	0.6
300	2.9	1.3	1.3	1.9
400	6.9	3.0	3.0	4.2
500	13.6	5.6	5.7	8.2
600	24.1	10.1	9.9	13.5
700	37.2	16.3	16.0	21.1
800	56.0	22.6	22.8	30.9
900	80.2	32.3	32.0	43.9
1000	110.2	46.3	44.8	60.6

Array's Predictable Inefficiency:

As expected, the Array heap exhibited suboptimal performance due to its O(N) time complexity for all operations. The fixed size of the Array heap poses limitations compared to more dynamic structures.

Fibonacci Heap's Practical Challenges:

Contrary to theoretical expectations, the Fibonacci Heap displayed worse performance than the Binary Heap, albeit outperforming the Array heap. This suggests that while Fibonacci Heap has favorable amortized costs in big-O notation, practical complexities and high constant factors may diminish its real-world advantages.

Binary and Binomial Heaps Excellence:

Both Binary and Binomial Heaps emerged as efficient choices. Binary Heap, a traditional option, showcased robust performance, while Binomial Heap, marginally superior for larger graphs, demonstrated consistent efficiency. Binomial Heap's slight edge may become more pronounced with larger datasets.

Conclusion

The efficiency of heap structures transcends the confines of big-O notation.

The empirical findings underscore that real-world performance is shaped by various factors, including constant factors in operations, memory management complexities, and intricate implementations.

Binomial heap > Binary Heap > Fibonacci heap > Array

Therefore, the practical efficiency of heaps cannot be accurately predicted based solely on theoretical complexities.
A comprehensive evaluation, accounting for both theoretical and practical considerations, is crucial for making informed decisions in algorithm design. This study emphasizes the necessity of a nuanced approach to assess heap structures and select the most effective one for specific applications.