CS201 EndTerm Assignment Report

Johnson's Algorithm for All Pair Shortest Path using different types of heaps

Tanuj Maheshwari

2019CSB1125

OBJECTIVE

The assignment's objective was to implement Johnson's Algorithm for All Pair Shortest Path using four different types of heaps :- Array, Binary Heap, Binomial Heap & Fibonacci Heap.

This report covers the analysis of the algorithm using these four heaps and comments on their efficiency, both theoretically and practically.

THEORETICAL UPPER BOUNDS FOR ELEMENTARY OPERATIONS

The following table shows the upper bounds (in big-O notation) for elementary operations on all 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)

PRACTICAL ANALYSIS

For practical analysis, the code was run on different types of randomly generated graphs for all the four heaps. Then, the execution times were plotted against the parameters.

Note that only the execution times are considered. The times for reading input & printing output is not taken into account.

The parameters for generation of graphs are :-

- **Probability of edge** between a vertex u and another vertex v (for all vertices)
- **Number of nodes** in the graph

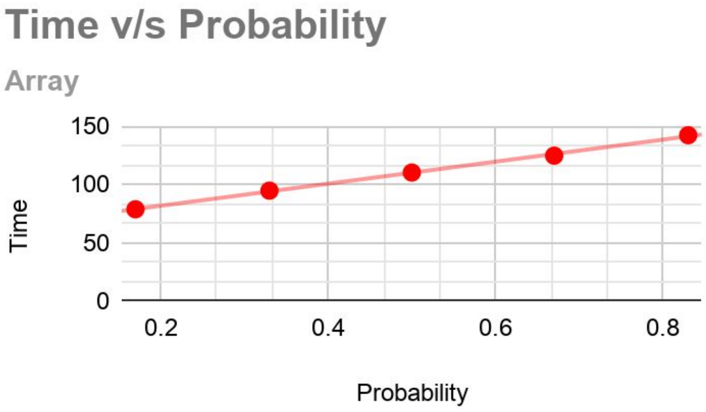
(Number of edges is not a parameter, as the number of edges can be easily found out by multiplying the probability of edge existence with the square of the number of nodes. In other words, the *number of edges is a function of these two parameters*).

RELATION BETWEEN PROBABILITY (EDGES) AND EXECUTION TIME[1]

1. ARRAY

The following table and plot shows the variation of execution time v/s probability of edge between two vertices for the construction of graphs. (Nodes = 1000).

Probability	Time(s)
0.17	78.7
0.33	94.7
0.50	110.2
0.67	124.7
0.83	142.3

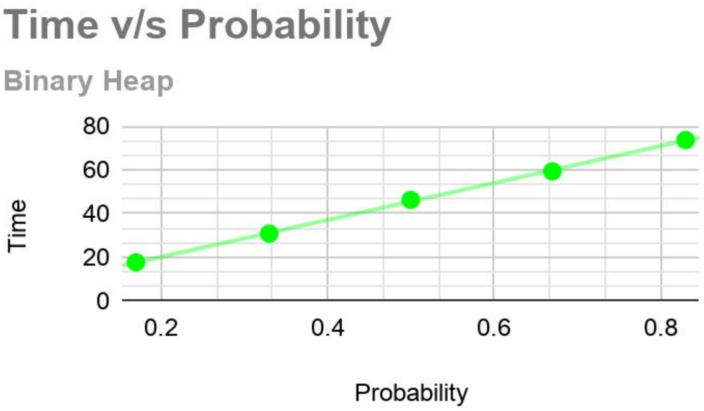


[1] The relation between number of nodes and execution time will be plotted later with other heaps.

2. BINARY HEAP

The following table and plot shows the variation of execution time v/s probability of edge between two vertices for the construction of graphs. (Nodes = 1000).

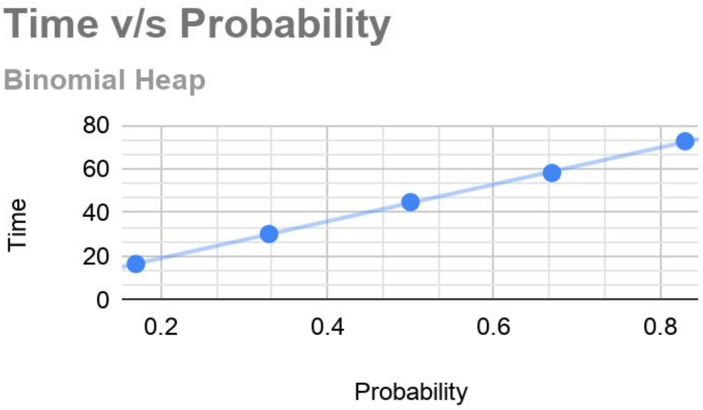
Probability	Time(s)
0.17	17.7
0.33	30.8
0.50	46.3
0.67	59.3
0.83	73.7



3. BINOMIAL HEAP

The following table and plot shows the variation of execution time v/s probability of edge between two vertices for the construction of graphs. (Nodes = 1000).

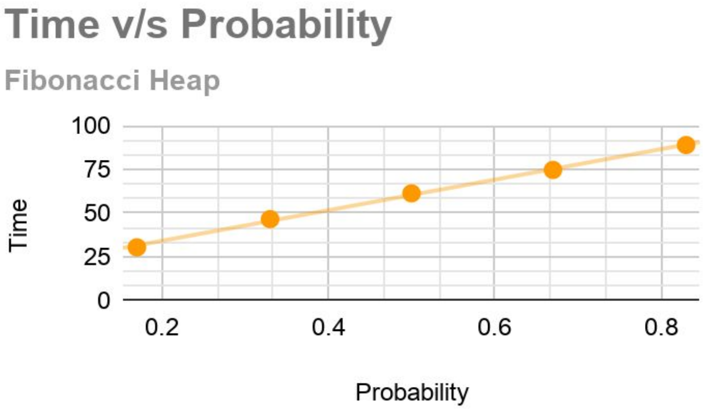
Probability	Time(s)
0.17	16.5
0.33	30.2
0.50	44.8
0.67	58.1
0.83	72.6



4. FIBONACCI HEAP

The following table and plot shows the variation of execution time v/s probability of edge between two vertices for the construction of graphs. (Nodes = 1000).

Probability	Time(s)
0.17	30.4
0.33	46.6
0.50	61.3
0.67	74.7
0.83	88.9



FINAL VERDICT ON PROBABILITY V/S TIME OF EXECUTION

On plotting the probability v/s execution time for all the heaps, it is clear that the execution time depends linearly on the probability of edge between two vertices. That is the same as saying **execution time depends linearly on the number of edges in a graph, given that the number of vertices remains constant**.

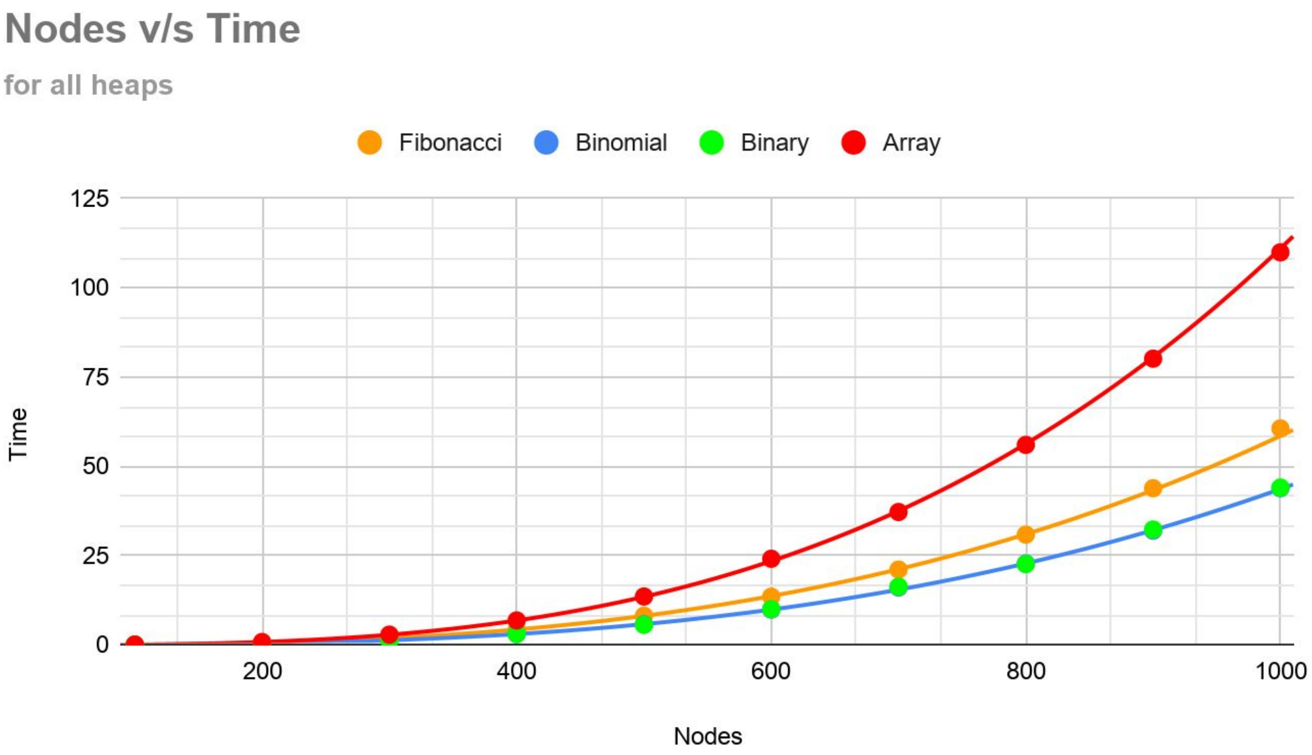
As it is evident from the graphs, *the above statement is true for all the four types of heaps discussed*.

RELATION BETWEEN NUMBER OF NODES AND EXECUTION TIME

The following table and graph shows the relation between number of nodes and average execution times for all the heaps (and hence is also a measure of efficiency). The probability of an edge between two nodes (and hence the number of edges) is kept constant to 50% (edges = $V^2/2$).

	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

Following is the plot for the above data. Note that because the difference between execution times of Binary Heap and Binomial Heap is very less, only the points are plotted for binary heap and only the trendline is plotted for binomial heap.



FINAL VERDICT ON NUMBER OF NODES V/S EXECUTION TIME

On plotting the number of nodes v/s execution time for all the heaps, it is clear that **theoretical trends for execution times are followed, however, due to the lack of constants in the big-O notation, efficiency can't be accurately determined.**

Although big-O notation predicts that Fibonacci Heap should be the fastest, this is not observed practically.

DISCUSSION

On analysis of the results and after plotting graphs, it is clear that :-

- **Array performs the worst**, as expected, because all operations take O(N) time.
- **Fibonacci Heap, contrary to theoretical results, performs even worse than Binary Heap** (but better than array). This implies that although it has the smallest amortized costs in big-O notation, *the constants involved are too big for any practical purposes*, maybe because of the complexity of operations involved.
- **Binary & Binomial Heaps perform the best**, with Binomial Heap just very slightly better for larger graphs, that too not very significant.

CONCLUSION

In conclusion, it can be said that the efficiency of heaps can’t be determined by big-O notation, and the **actual trend of efficiency** is :-

Binomial Heap ≥ Binary Heap > Fibonacci Heap > Array