ASSIGNMENT REPORT

Lab related to Fibonacci heaps

CS201: Data Structures

ANEEKET MANGAL

2019CSB1071

PROBLEM STATEMENT

Write a program for implementing Johnson's algorithm for All Pair Shortest Path where the algorithm's time complexity is analysed (considering the execution times on large graphs), considering different data structures based heaps implementation.

IMPLEMENTATIONAL DETAILS

1. Code file has 4 classes:-
    a. Array Heap
    b. Binary Heap
    c. Binomial Heap
    d. Fibonacci Heap

    Where each class provides functionality evident by their names.
2. The time calculated for data sampling includes running time of dijkstra and bellman ford both.
3. Data has been plotted on charts using google sheets.
4. Random graphs were generated using python.

OBSERVATIONS

Time taken for Johnson Algorithm (in sec) [Bellman running time inclusive] (calculated on Ubuntu 19 g++ 9.3 compiler)

On X-axis:- Total number of nodes in the test case

On Y-axis:- Time taken by Johnson's algorithm

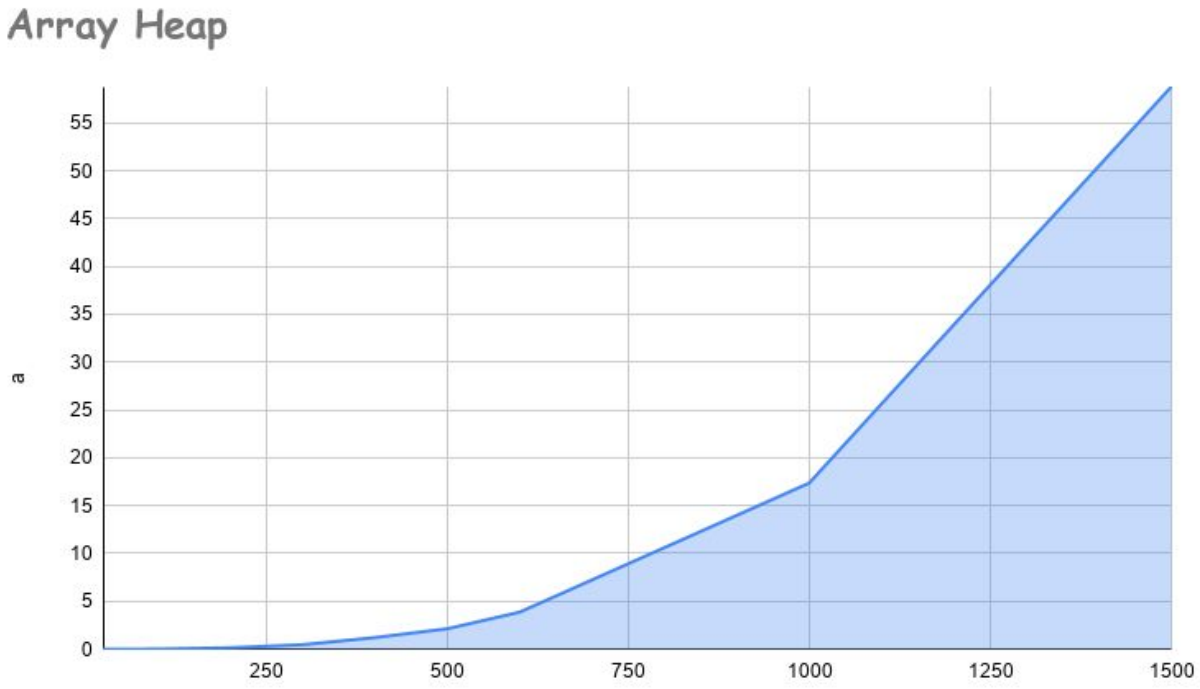
Total Nodes	Array Heap	Binary Heap	Binomial Heap	Fibonacci Heap
5	0.000054	0.000043	0.000066	0.000125
10	0.000127	0.000107	0.000211	0.00046
20	0.000482	0.000464	0.000765	0.001873
50	0.003873	0.003716	0.009857	0.013418
100	0.022637	0.02375	0.072865	0.060907
200	0.153195	0.154136	0.673877	0.310929
300	0.473683	0.454553	2.236451	0.846948
400	1.217817	1.143829	5.670469	1.75422
500	2.140866	1.900788	11.426701	2.918743
600	3.8775	3.54109	19.230969	4.719912
1000	17.975078	17.075042	97.380368	21.850084
1500	58.792882	54.517637	345.881615	66.33836

ARRAY HEAP

Time complexity of different operations(theoretically):-

- ExtractMin() : O(V)
- DecreaseKey() : O(1)
- Dijkstra's running time : O(V^2)

The graph is showing a similar trend as of quadratic graph

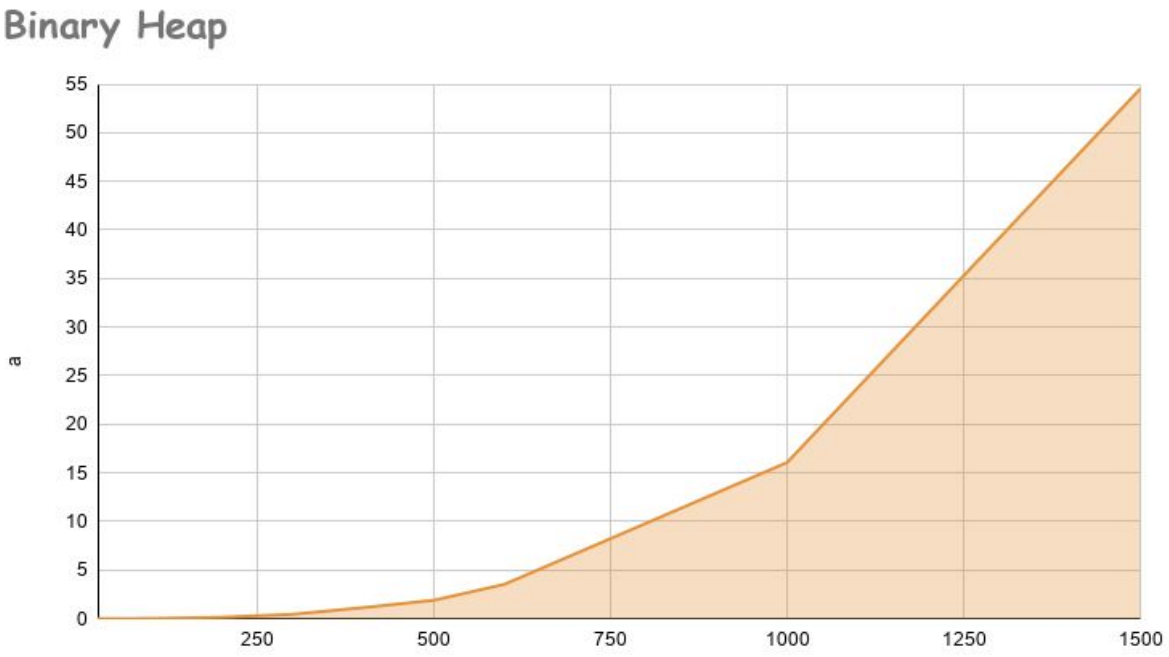


BINARY HEAP

Time complexity of different operations(theoretically):-

- ExtractMin() : O(logV)
- DecreaseKey() : O(logV)
- Dijkstra's running time : O(ElogV)

This graph shows that binary heap implementation is better than array heap which is evident by theoretical binary heap dijkstra running time as well.

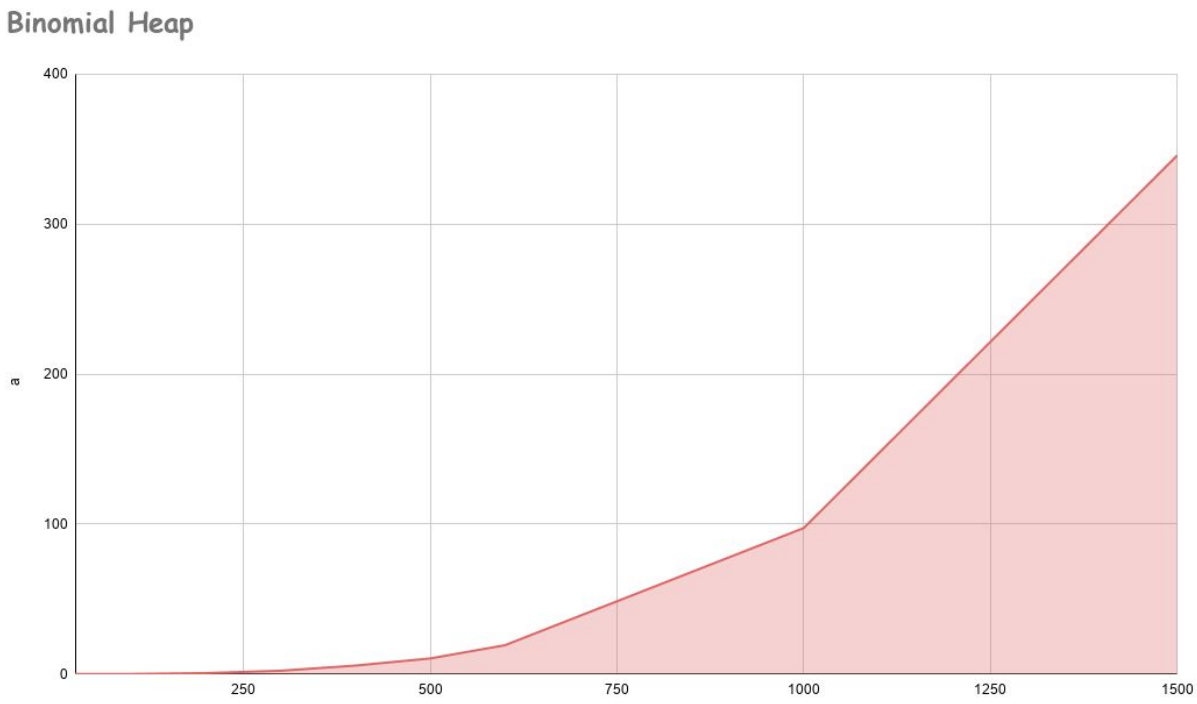


BINOMIAL HEAP

Time complexity of different operations(theoretically):-

- ExtractMin() : O(logV)
- DecreaseKey() : O(logV)
- Dijkstra's running time : O(ElogV)

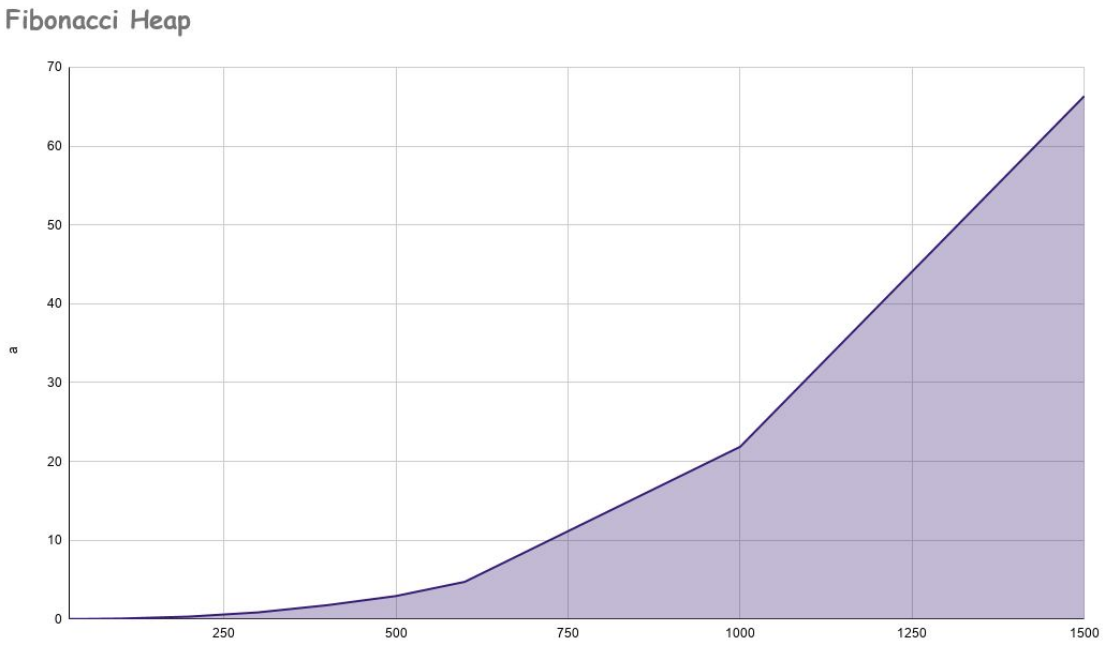
Due to implementational issues, binomial heap does not seem to be working because the decreasekey could not be implemented. So, the trend could not be studied well.



FIBONACCI HEAP

Time complexity of different operations(theoretically):-

- ExtractMin() : O(logV)
- DecreaseKey() : O(1) {amortized}
- Dijkstra's running time : O(VlogV + E)



CONCLUSION

1. Array heap and Binary heap were the fastest amongst all johnson algorithm implementations.
2. Fibonacci Heap was supposed to be the fastest theoretically, but it turned out that it worked slower as compared to the others(array and binary). This might be due to too many intricate operations in implementation or unoptimized implementation.
3. The nodes vs time graphs are increasing.
4. Problems with the code:-
    a. The decrease key functionality of binomial heap is broken, so to run the code, instead of decreasing key I am adding another key each time.
    b. Due to this anomaly, the binomial heap could not perform as expected.

REFERENCES

1. Introduction to algorithms by Cormen 3rd edition .
2. Geeks for geeks
3. Wikipedia
