भारतीय प्रौद्योगिकी
संस्थान, रोपड़
Indian Institute of
Technology Ropar

CS - 201

Lab - 6 Report

Submitted by :- AYUSH VERMA
Entry No.:- 2019CSB1147

Submitted to:- *Dr. Puneet Goyal*

Johnson's Algorithm

This algorithm is used to find all pairs shortest path between every pair of vertices in a weighted undirected/directed graph(where weights may be negative).
The idea of Johnson's algorithm is to re-weight all edges(as Dijkstra algo can't be applied for negative weight graphs) and make them all positive, then apply Dijkstra's algorithm for every vertex.
Bellman-Ford algorithm is used to reweight all the edges.

- We add a new source s to the graph G and then the graph becomes G'.
- Run bellman algorithm for this G' with s as source.
- Re-weight the edges of graph G with
  new_weight=original_weight+h[u]-h[v]
- Remove the added source vertex
- Run the Dijkstra Algorithm with every vertex as a source to find all pairs shortest path.

Bellman Ford Algorithm

Given a graph and a source vertex s in the graph, find shortest paths from s to all vertices in the given graph. The graph may contain negative weight edges.
This algorithm also reports if any negative weight cycle is present in the graph.

Time-Complexity of Bellman-Ford Algorithm
This algorithm takes O(VE) time complexity using adjacency-list representation of graphs.

where, V->Number of vertices
E->Number of Edges

Dijkstra's Algorithm

Given a graph and a source vertex in the graph, this algorithm finds shortest paths from source to all vertices in the given graph.
Note-This algorithm fails if there is any negative weights or negative edge cycle in the graph.Therefore Bellman-Ford algorithm is used at first place to re-weight the graphs.

Dijkstra algorithm can be used by the following 4 different heap data structures:-

1. Arrays
2. Binary heap
3. Binomial heap
4. Fibonacci heap

Note- Fibonacci Heap data structure has not been implemented by me.

Function/Data str.	**Array**	**Binary Heap**	**Binomial Heap**	**Fibonacci Heap**
Insert	O(1)	O(log n)	O(log n)	O(1)
Extract min	O(n)	O(log n)	O(log n)	O(log n)
Decrease Key	O(1)	O(log n)	O(log n)	O(1)

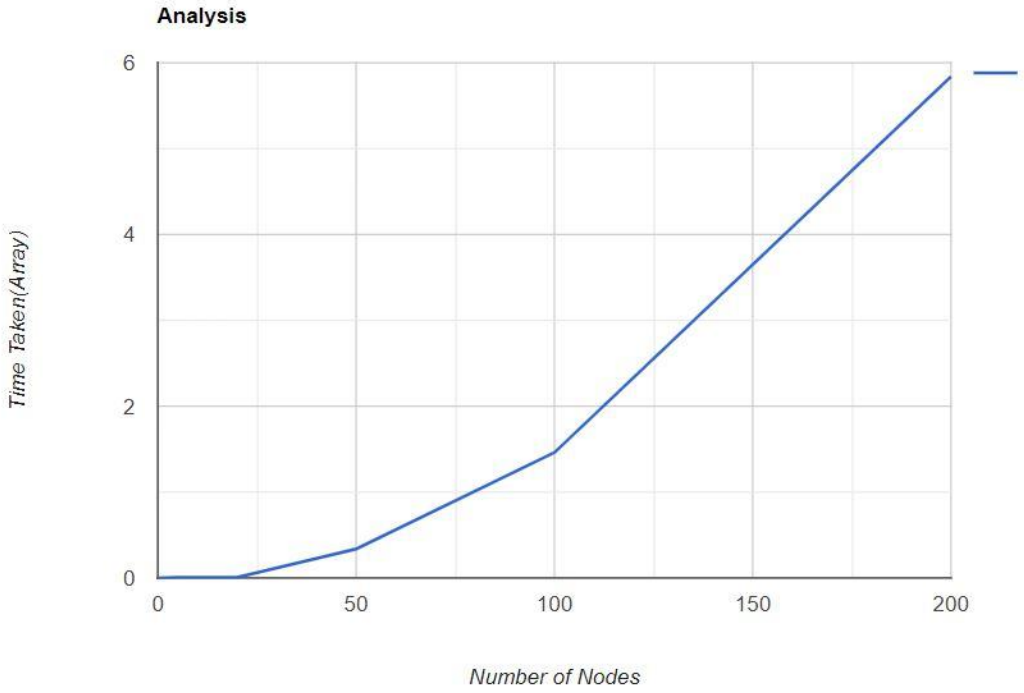
Table of Theoretical time complexity for various functions of Dijkstra algorithms using various heap data structures.

Theoretically,Array data structure has the worst time complexity while fibonacci heap has the best time complexity among all 4 data structures.

Observations on testcases

1. Array based implementation

Time complexity of array based Dijkstra algorithm is $O(V^2)$ for a single node.
So, overall time complexity of this implementation becomes $O(V^3)$.

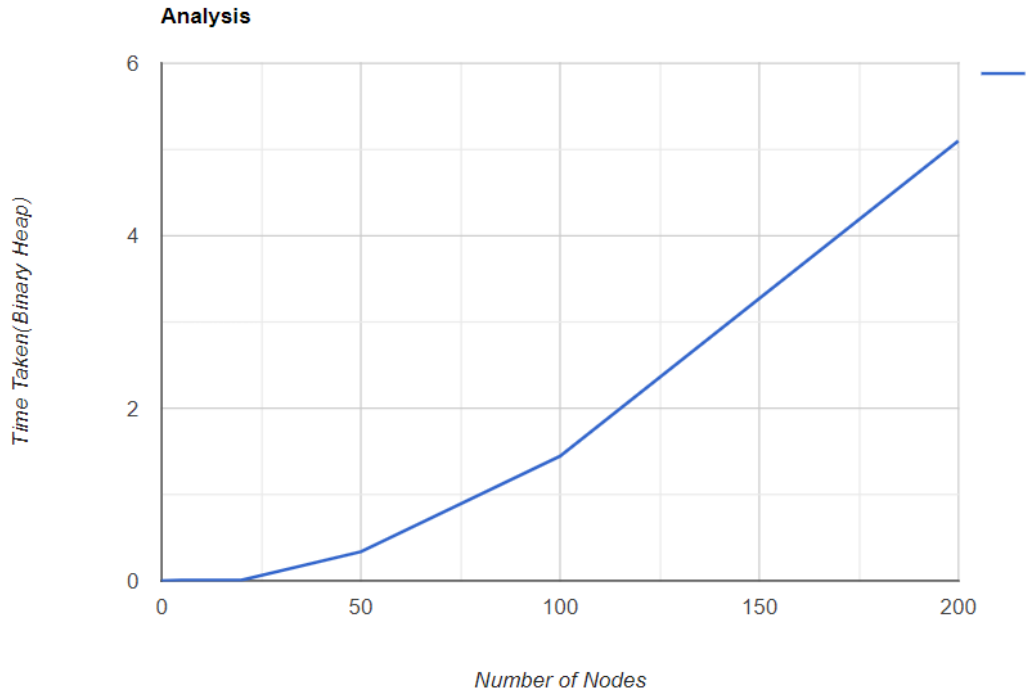


Number of Nodes	Time taken(Array)
0	0
5	0.006
20	0.056
50	0.337
100	1.462
200	5.84
1000	117.312

2.Binary heap based implementation

Time complexity of array based Dijkstra algorithm is O((V+E)logV) for a single node.

So, overall time complexity of this implementation becomes O(EVlogV).

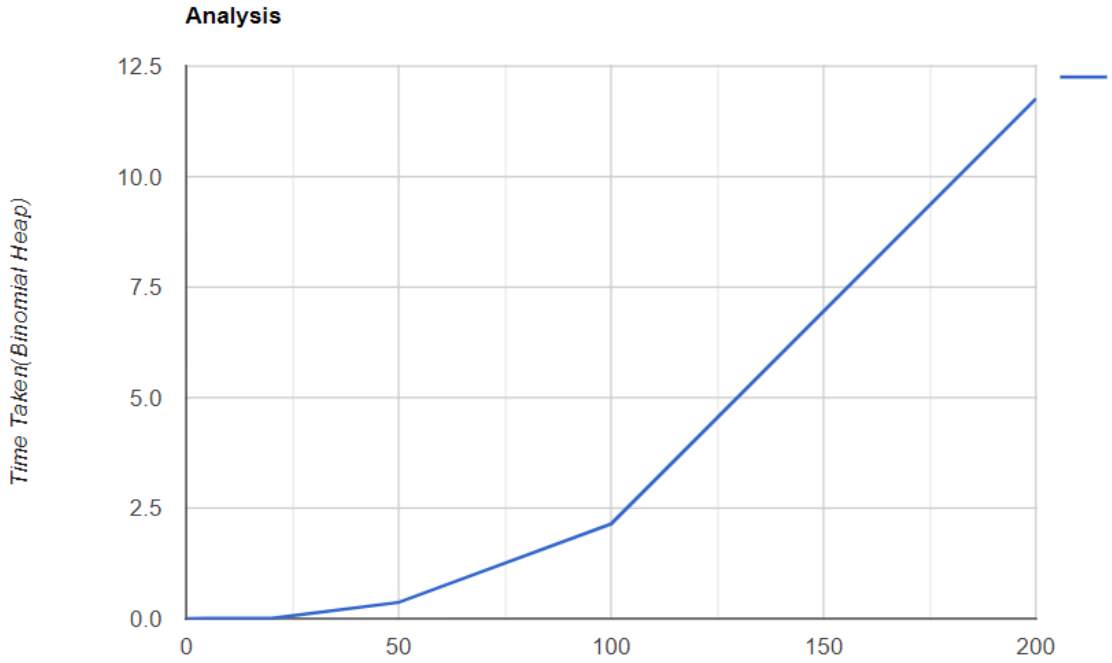


Number of Nodes	Time taken(Binary Heap)
0	0
5	0.005
20	0.0054
50	0.336
100	1.445
200	5.1
1000	100

3.Binomial Heap based implementation

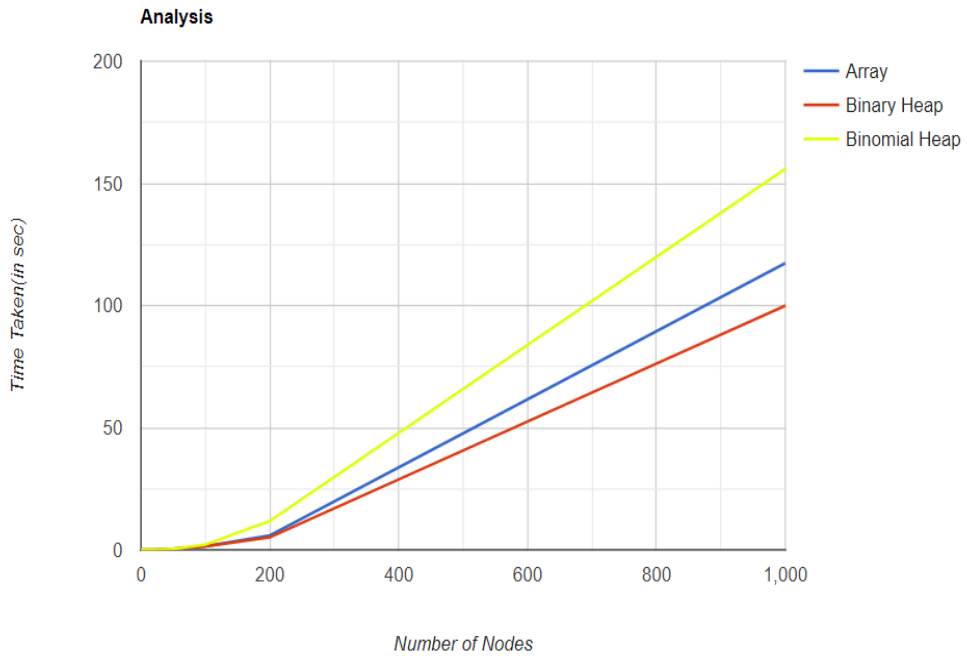
Time complexity of array based Dijkstra algorithm is O((V+E)logV) for a single node.

So, overall time complexity of this implementation becomes O(EVlogV).



Number of Nodes	Time taken(Binomial Heap)
0	0
5	0.006
20	0.0055
50	0.365
100	2.139
200	11.764
1000	156

Overall observation on test cases



Above is the plot of the time taken for the algorithm for different data structures on my code for randomised graphs(with no negative edge cycle)

Number of Nodes	Time taken(Array)	Time taken(Binary Heap	Time taken(Binomial Heap)
0	0	0	0
5	0.006	0.005	0.006
20	0.056	0.0054	0.0055
50	0.337	0.336	0.365
100	1.462	1.445	2.139
200	5.84	5.1	11.764
1000	117.312	100	156

Discussion

We can observe from the plot that Binary based implementation takes less execution time than the rest of two.Binomial heap based implementation takes most execution time,although it has better theoretical time complexity.

Conclusion

The time taken for the code to run on graphs with smaller number of vertices is almost the same.When the program is run on testcases with higher number of vertices, the time taken goes on increasing.
The time taken for higher N(~1000) is as follows:
Binomial Heap >Array>Binary Heap.
Although binomial heap has better theoretical time complexity than array method,it takes more execution time.
Binary heap based implementation runs fastest among the implemented data structures.