Algorithms and Data Structure Project
Elementary Graph Algorithms

Himani Tokas

27th May 2023

1 Introduction

Graphs can be directed or un-directed networks. A graph is represented as G(V,E), where 'E' represent edges and 'V' represent vertices of the graph.Depth First Search and Breadth First Search are two elementary graph algorithms.

1.1 Idea of Depth First Search

Its idea is to search deeper in the tree/adjacency list.Depth-first search explores edges out of the most recently discovered vertex v that still has unexplored edges leaving it. Once all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered. This process continues until we have discovered all the vertices that are reachable from the original source vertex. If any undiscovered vertices remain, then depth-first search selects one of them as a new source, and it repeats the search from that source. The algorithm repeats this entire process until it has discovered every vertex.

1.2 Idea of Breadth First Search

Given a graph G(V,E) and a distinguished source vertex s, breadth-first search systematically explores the edges of G to "discover" every vertex that is reachable from s. It computes the distance (smallest number of edges) from s to each reachable vertex. It also produces a "breadth-first tree" with root s that contains all reachable vertices. For any vertex v reachable from s, the

simple path in the breadth-first tree from s to v corresponds to a "shortest path" from s to v in G, that is, a path containing the smallest number of edges. The algorithm works on both directed and undirected graphs.

1.3 Graph Representation

1.3.1 Undirected Graph Representation

An undirected graph does not have any arrows. It looks as follow:

```
Total number of nodes:  6
Total number of edges:  6
List of all nodes:  [5, 3, 7, 2, 4, 8]
List of all edges:  [(5, 3, {}), (5, 7, {}), (3, 2, {}), (3, 4, {}), (7, 8, {}), (4, 8, {})]
Degree for all nodes:  {5: 2, 3: 3, 7: 2, 2: 1, 4: 2, 8: 2}
List of all nodes we can go to in a single step from node 5:  [3, 7]
```

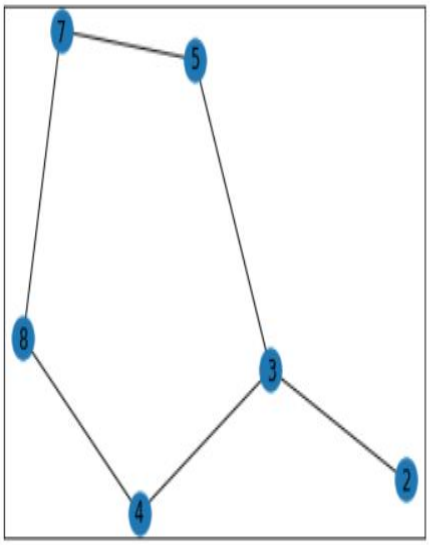

Figure 1: An undirected graph with 6 vertices and 6 edges

1.3.2 Directed Graph Representation

An directed graph have arrows. It looks as follow:

```
Total number of nodes:  6
Total number of edges:  6
List of all nodes:  ['A', 'B', 'C', 'D', 'E', 'F']
List of all edges:  [('A', 'B'), ('A', 'C'), ('B', 'D'), ('B', 'E'), ('C', 'F'), ('E', 'F')]
In-degree for all nodes:  {'A': 0, 'B': 1, 'C': 1, 'D': 1, 'E': 1, 'F': 2}
Out degree for all nodes:  {'A': 2, 'B': 2, 'C': 1, 'D': 0, 'E': 1, 'F': 0}
List of all nodes we can go to in a single step from node 2:  ['D', 'E']
List of all nodes from which we can go to node 2 in a single step:  ['A']
```

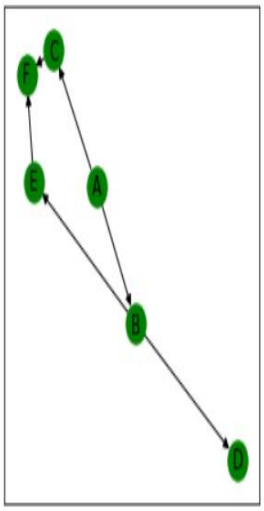

Figure 2: A directed graph with 6 vertices and 6 edges

2 Methodology

For analyzing the algorithm more than one graph is required. Since, it is not possible to create adjacency lists randomly, first adjacency matrices of size varying from 10×10 to 2000×2000 were generated and converted into adjacency lists. The function where adjacency list is being created call the

depth first search and breadth first search functions to solve these algorithms and returns back the output.

3 Data used to study Depth First Search Time Complexity

3.1 Data

Random adjacency matrices of size varying from 10×10 to 2000×2000 were generated and converted into adjacency lists. A total of 199 data points are used to find time complexity:

```
In [121]: dfsData
```

Out[121]:

	AdjMatrixSize	V+E	Duration
0	10	45	0.017370
1	20	205	0.005030
2	30	487	0.037808
3	40	837	0.049507
4	50	1267	0.022724
...	...	...	...
194	1950	1903077	14.599379
195	1960	1923459	14.502845
196	1970	1942133	14.632833
197	1980	1961535	16.596036
198	1990	1982549	14.883430

199 rows × 3 columns

Figure 3: Data points for depth first search analysis: V+E represent sum of vertices and edges and Duration is the time taken to evaluate the particular adjacency list.

4 Time Complexity of Depth First Search Algorithm

Time complexity is given as O(V+E) because we are visiting every edge, traversing through all vertices.

4.0.1 Time Complexity found through data regression

```
Depth First Search complexity for time vs v+e:

alpha=5.447488756406896e-06, beta=1.0186913499397199
```

Figure 4: Beta value for time complexity is approximately 1.01. If we get more data points it will be more closer to 1.Hence time complexity for DFS is O(V+E).

4.0.2 Regression Graph

Following curve is obtained for duration versus sum of vertices and edges for DFS.Data points were increased to 790 to get a better visual.

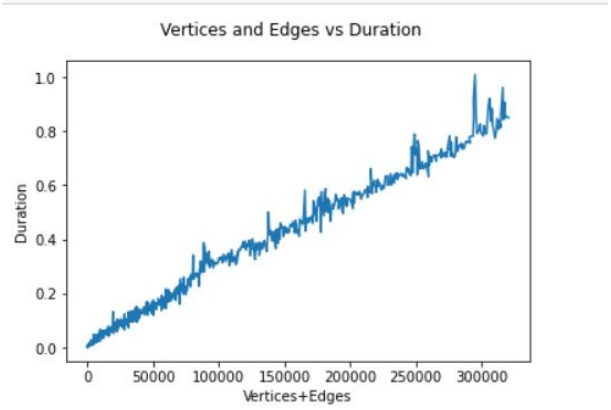

Figure 5: DFS plot for duration and number of edges and vertices. Total data points on graph is 790

5 Application of Depth First Search from CLRS Book

5.1 Topological Sort

A topological sort of dag(directed a-cyclic graph) G(V,E) is linear ordering of all its vertices such that if G contains an edge (u,v) then u must come before v. Following problem is given in the book:

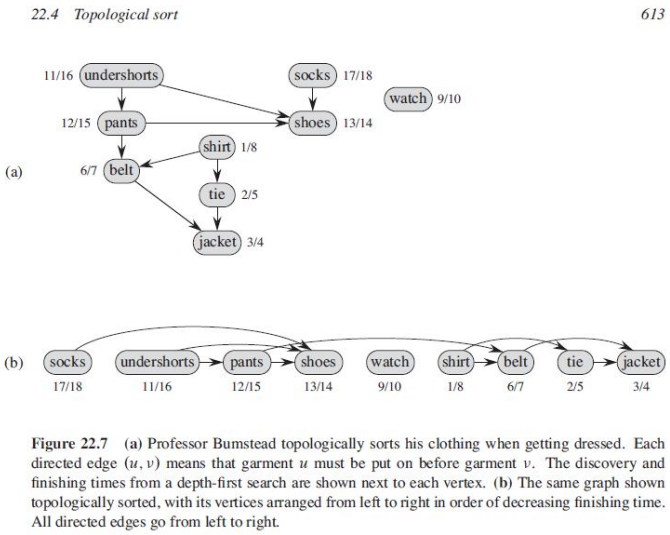

Figure 22.7 (a) Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge (u, v) means that garment u must be put on before garment v. The discovery and finishing times from a depth-first search are shown next to each vertex. (b) The same graph shown topologically sorted, with its vertices arranged from left to right in order of decreasing finishing time. All directed edges go from left to right.

Figure 6: From fig(a) an adjacency list is created and DFS is applied to find (b) as output.

5.1.1 Adjacency List

Following Adjacency List is formed from fig(a):

```
Adjacency List:
 {'undershorts': ['shoes', 'pants'], 'pants': ['belt', 'shoes'], 'shirt': ['belt', 'tie'], 'belt': ['jacket'], 'tie': ['jacke
t'], 'socks': ['shoes'], 'watch': [], 'shoes': [], 'jacket': []}
```

Figure 7: Adjacency List in form of a python dictionary

5.1.2 Execution: Starting from 'Undershorts'

Following sorted order is obtained if we start sorting from 'undershorts' using DFS:

```
dfs(visited, graph, 'undershorts')
```

```
Topological Sort Order:
undershorts
shoes
pants
belt
jacket
```

Figure 8: Topological Sort starting from 'undershorts'. Different sorted orders will be obtained if we change the starting point.

6 Data Used for Breadth First Search Analysis

Adjacency matrices order varying from 10×10 to 300×300 were converted into adjacency lists. A total of 290 data points are used:

```
bfsData
```

	AdjMatrixSize	V+E	Duration
0	10	53	0.001982
1	11	62	0.003016
2	12	85	0.004364
3	13	96	0.001775
4	14	89	0.008468
...	...	...	...
285	295	43572	0.221406
286	296	44023	0.225139
287	297	44210	0.225877
288	298	44813	0.219074
289	299	44598	0.223998

290 rows × 3 columns

Figure 9: Data for breadth first search analysis

7 Time Complexity for BFS Algorithm

It is also given as O(V+E), similar to DFS.

7.0.1 Time Complexity found through data regression

```
Breadth First Search complexity for time vs v+e:

alpha=3.90134110277477e-06, beta=1.0232266625674822
```

Figure 10: Beta value is closer to 1, hence time complexity is O(V+E)

7.0.2 Data Regression Graph

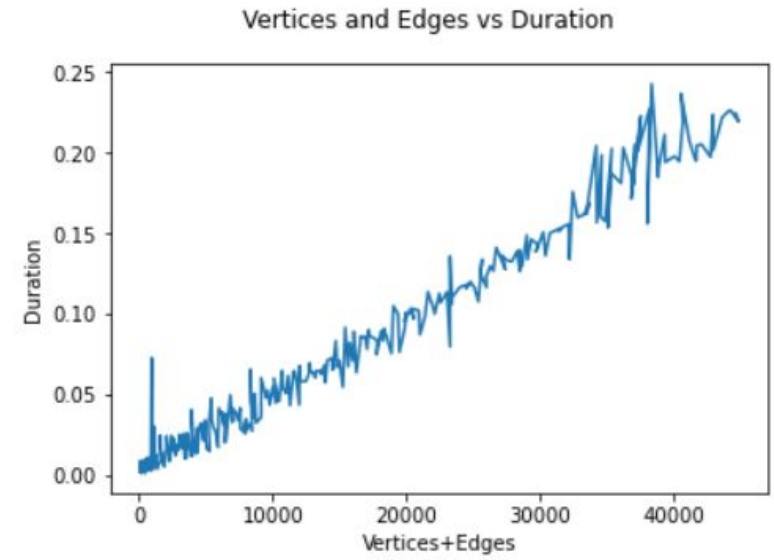

Figure 11: This graph is obtained if we plot duration and sum or vertices and edges.

8 References

1. Book: CLRS (Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein): Introduction to Algorithms, Third Edition

2. https://stackoverflow.com/