Recitation Class 4

Q1 – Suppose there exists an $n \times m$ 2D binary array A where obstacles are denoted by 1. You start at an arbitrary location s on the grid and want to get to another location t. Give an efficient algorithm for finding the shortest path from s to t. The only moves which are allowed are UP, DOWN, LEFT, and RIGHT. Diagonal moves are not allowed.

Answer:

Construct graph G in the following way; For each entry of array take a vertex. The total number of vertices is nm. For each location on the grid, there are at most 4 other locations which are neighbors to that. Add edges between the associated vertices of neighboring locations. The total number of edges is at most $2nm$. Here is the algorithm

1. Construct graph G from the input array A.
2. Run BFS on graph G with starting from s.
3. d_t computed by BFS gives the shortest path distance between vertex t and s.
4. In order to obtain the path, follow t, $\pi(t)$, $\pi\big(\pi(t)\big)$, ... until you reach vertex s.

The running time is $O(mn)$.

Q2 – Design a linear-time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

Answer:

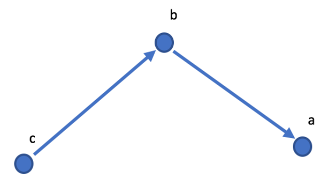
1. Construct graph G' by removing edge e from graph G.
2. Let u, v be the two endpoints of edge e.
3. Run BFS on graph G' starting from vertex u.
4. Check if $\pi(v)$ is NULL, then no cycle contains edge e.
5. Otherwise, there is some cycle that contains edge e.

The running time of this algorithm is $O(|V| + |E|)$, where V, E are respectively vertices and edges of graph G.

Q3 – Explain how a vertex b of a directed graph can end up in a depth-first tree containing only b, even though b has both incoming and outgoing edges in G.

Answer:

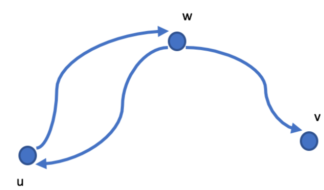
Take the following graph as an example, and run DFS starting on vertex a, then run on b, and finally run on c. In this way, there are 3 DFS-trees each contains a single vertex.



Q4 – Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, then any depth-first search must result in $\mathrm{v.d} \leq \mathrm{u.f}$, where $v.d$ is discover time of vertex v and $u.f$ is finish time of vertex u.

Answer:

In this example, run DFS starting from vertex w, then visit vertex u, and finally visit vertex v. In this way, finish time of vertex u is 3, and discover time of vertex v is 4.



Q5 – Explain how strongly connected components of a DAG may look like.

Answer:

In DAGs (directed acyclic graphs), each vertex of the graph forms a strongly connected component, in other words there is no strongly connected components in a DAG that contains more than one vertex. This can be proved by contradiction. Assume there is a DAG that has a strongly connected component with at least two vertices u, v. From the definition of strongly connected component, we know that there is a directed path P_1 from u to v, and a directed path P_2 from v to u. Thus, Union of $P_1$and P_2 forms a directed cycle. This is a contradiction that the graph is DAG.