

# CSC373 Worksheet 4 Solution

August 4, 2020

1. • Calculating out-degree

Let  $G = (V, E)$  be a directed graph. Let  $[v_1, \dots, v_n]$  be a list of vertices in graph  $G$ .

I need to calculate the outdegree of every vertex using adjacency list.

We know that in addition to counting each  $v_i$  in adjacency list where  $i = 1, \dots, n$ , we are also counting  $|Adj[v_i]|$  edges.

Since there are  $|V| = n$  many vertices, we can write that the total count is  $|V| + \sum_{i=1}^n |Adj[v_i]| = |V| + |E|$ , which is  $\mathcal{O}(|V| + |E|)$ .

• Calculating In-degree

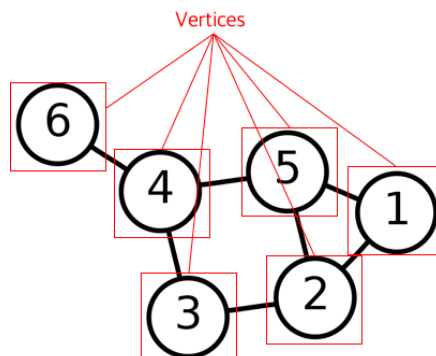
The outdegree of a vertex is indegree of another vertex.

Using this fact, we can conclude the running time of computing indegree of every vertex is  $\mathcal{O}(|V| + |E|)$ .

Notes:

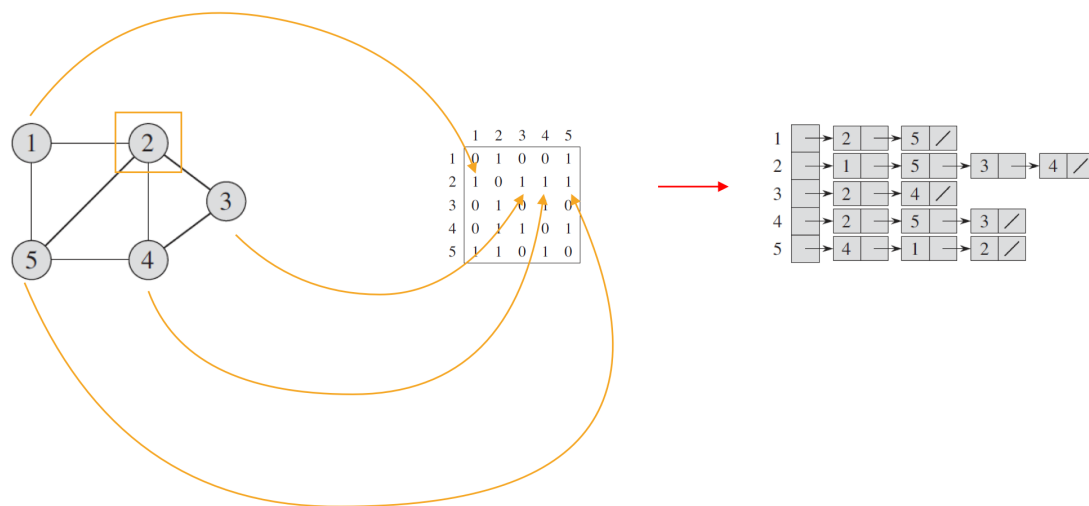
• **Vertex**

- Is a fundamental unit of which graphs are formed
- Also means node



### • Adjacency-list Representation

- Associates each vertex in a graph with the collection of its neighbouring vertices or edges
- Is represented by  $Adj[v]$ 
  - \* Means all vertices that are neighbour to vertex  $v$
  - \* In a directed graph,  $Adj[v]$  are all out-degree vertices of vertex  $v$
  - \*  $|Adj[v]|$  means the total number of outdegree of vertex  $v$







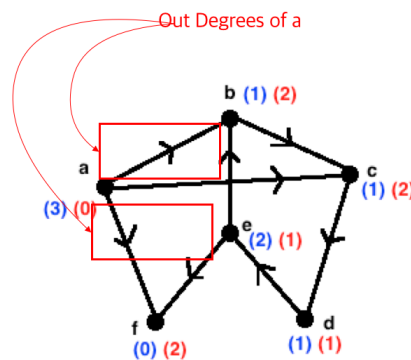
### • Directed graph

- Is a graph that is made up of a set of vertices connected by edges, where the edges have a direction associated with them



### • Out-degrees

- For a directed graph  $G = (V(G), E(G))$  and a vertex  $x_1 \in V(G)$ , the Out-Degree of  $x_1$  refers to the number of arcs incident from  $x_1$ . That is, the number of arcs directed away from the vertex  $x_1$ .



### • In-degrees

- For a directed graph  $G = (V(G), E(G))$  and a vertex  $x_1 \in V(G)$ , the In-Degree of  $x_1$  refers to the number of arcs incident to  $x_1$ . That is, the number of arcs directed towards the vertex  $x_1$ .



- Computing the outdegree of every vertex using adjacency list



3)



$$(v_1 + v_2 + v_3) + (e_1 + e_2 + e_3 + e_4 + e_5)$$

3)



$$(v_1 + v_2 + v_3) + (e_1 + e_2 + e_3 + e_4 + e_5)$$

4)



$$(v_1 + v_2 + v_3 + v_4) + (e_1 + e_2 + e_3 + e_4 + e_5)$$

4)



$$(v_1 + v_2 + v_3 + v_4) + (e_1 + e_2 + e_3 + e_4 + e_5)$$

6)



$$(v_1 + v_2 + v_3 + v_4 + v_5 + v_6) + (e_1 + e_2 + e_3 + e_4 + e_5 + e_6 + e_7 + e_8)$$

So it has  $\mathcal{O}(V + E)$

- Computing the outdegree of every vertex using adjacency list

The outdegree of a vertex is indegree of another vertex.

Using this fact, we can conclude the running time of computing indegree of every vertex is  $\mathcal{O}(V + E)$ .

- Computing  $G^T$  from  $G$  in Adjacency List



```

1  COMPUTE-G-TRANSPOSE-ADJ-MATRIX(Adj, V)
2      Let Adj' be a new adjacency list containing keys  $v_1 \dots v_n$ 
3
4      for i = 1 to |V|
5          for every vertex w in Adj[ $v_i$ ]
6              Insert(Adj'[w],  $v_i$ )
7
8      return Adj'
9

```

- Computing  $G^T$  from  $G$  in Adjacency-Matrix





```

1  COMPUTE-G-TRANSPOSE-ADJ-MATRIX(A,V)
2      Let A'[1..|V|, 1..|V|] be a new adjacency matrix
3
4      for i = 1 to |V|
5          for j = 1 to |V|
6              A'[j,i] = A[i,j]
7
8      return A'
9

```

### Correct Solution:

- Computing  $G^T$  from  $G$  in Adjacency List



```

1  COMPUTE-G-TRANSPOSE-ADJ-MATRIX(Adj,V)
2      Let Adj' be a new adjacency list containing keys
   v1...vn
3
4      for i = 1 to |V|
5          for every vertex w in Adj[vi]
6              Insert(Adj'[w], vi)
7
8      return Adj'
9

```

The running time is  $\mathcal{O}(|V| + |E|)$

- Computing  $G^T$  from  $G$  in Adjacency-Matrix



```

1  COMPUTE-G-TRANSPOSE-ADJ-MATRIX(A,V)
2      Let A'[1..|V|, 1..|V|] be a new adjacency matrix
3
4      for i = 1 to |V|
5          for j = 1 to |V|
6              A'[j,i] = A[i,j]
7
8      return A'
9

```

The running time is  $\mathcal{O}(|V|^2)$

```

31 Breadth-First-Search(V, v_i)
32     d = 0
33     for each v_i ∈ V
34         while performing BFS(V, v_i)
35             let w be the current node in BFS
36             if δ(v_i, w) > d
37                 d = δ(v_i, w)
38
39     return d
10

```

### Finding Runtime of Algorithm

Since the graph iterates  $\sum_{i=1}^n Adj[v_i] = |E|$  times for each  $v_i \in V$ , the algorithm iterates total of  $|V| \cdot |E|$  times, which is  $\mathcal{O}(|V||E|)$ .

### Notes:

- **Breadth First Search**

- Is an algorithm for searching or traversing a graph
- Is one of the simplest algorithm

- **Largest of All Shortest Path Distance**

- Means the shortest distance between two furthest apart nodes



OR



## References

1) McGill University, 308-360 Tutorial, [link](#)



4.



(e)



(f)



(g)



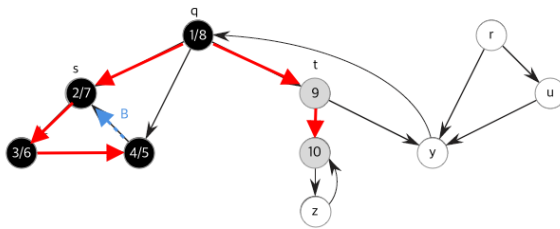
(g)



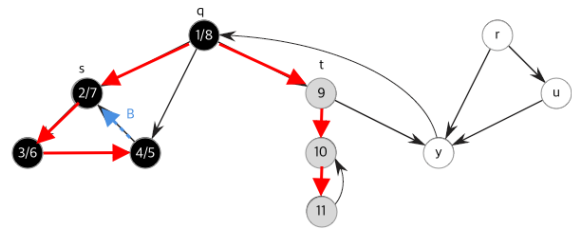
(h)



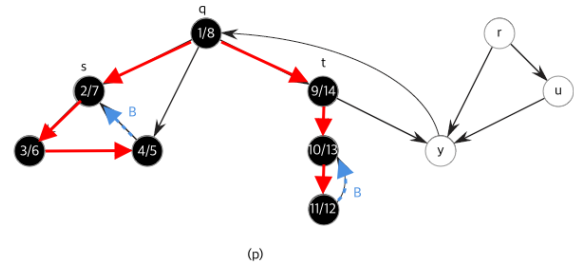
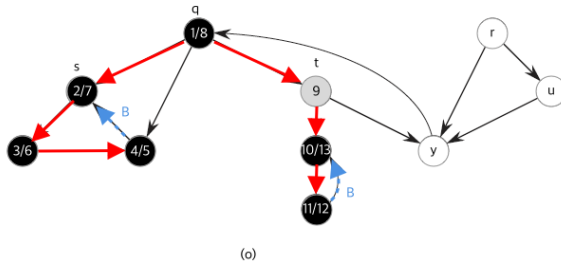
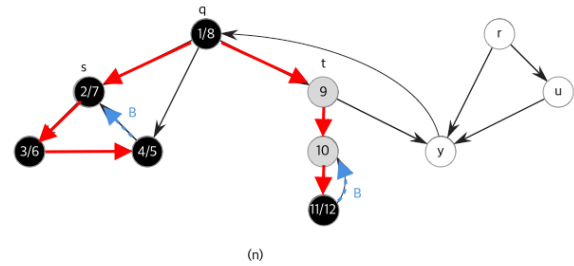
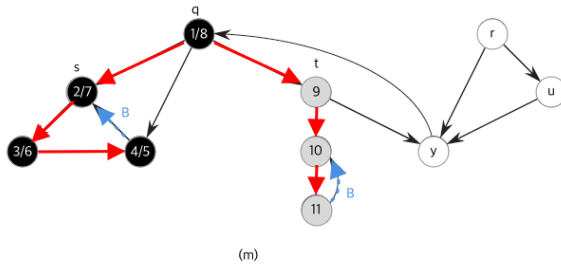
(i)



(k)



(l)



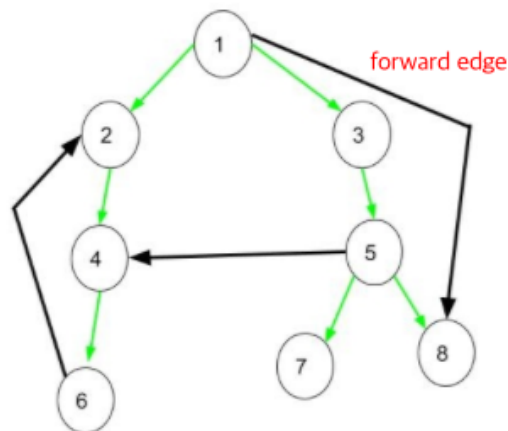
### Notes:

- **Depth First Search**

- Searches deeper in the graph whenever possible

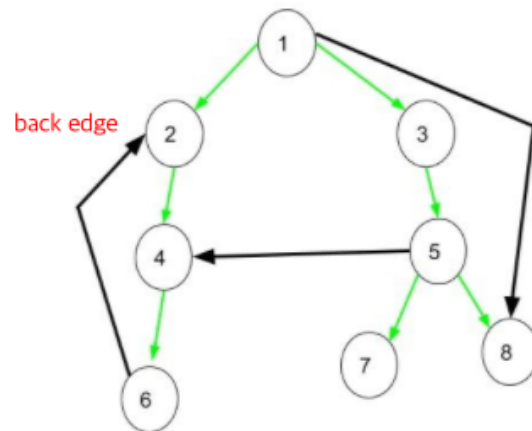
- **Forward Edge**

- Is an edge  $(u, v)$  such that  $v$  is descendant but not part of the DFS tree. Edge  $1 \rightarrow 8$  is a forward edge



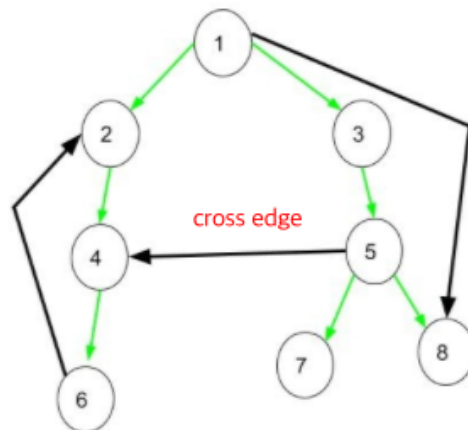
- **Back Edge**

- It is an edge  $(u, v)$  such that  $v$  is ancestor of edge  $u$  but not part of DFS tree. Edge from  $6 \rightarrow 2$  is a back edge.
- Indicates a cycle in a graph



- **Cross Edge**

- It is a edge which connects two node such that they do not have any ancestor and a descendant relationship between them. Edge from node 5  $\rightarrow$  4 is cross edge.



### References

- 1) Geeks For Geeks, Tree, Back, Edge and Cross Edges in DFS of Graph, link