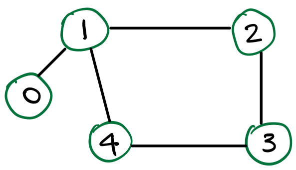
Seonhye Yang (sy3420) **Homework 7** CS-GY 1170

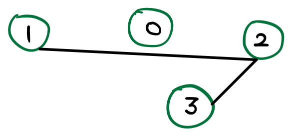
Problem 1

- a cycle in undirected graph is sequence of distinct vertices $(v_1, v_2, \cdots, v_k)$ with $k \geq 3$ such that edges $\{v_1, v_2\}, \{v_2, v_3\} \cdots \{v_{k-1}, v_k\}$ and $\{v_k, v_1\}$ all belong to the graph.
- Give undirected connected graph $G = (V, E)$, develop algo that determines whether G has cycle.

Example of cycle:



Example of no cycle



Algorithm: DFS

- make a graph using the given number of vertices and edges. Then make a recursive function that contains the current vertex, visited array, and parent node.
- Keep track of current node as we run through.
- Look for <u>all</u> the vertices which weren't visted and are adjacent to the current node. Then recursively call the algorithm for those vertices.
- If adjacent node has already been visited and is not a parent node, then return true or pass.
- Make decorators to call recursive functions for all the vertices. We do this since nodes might not be connected from starting vertex. In order to ensure that every vertex of G is connected, we use the recursive function for unvisited.
- Return false for others.

This algorithm is DFS traversal.

$TC = O(V+E)$

Problem 3

- Given directd graph G and 2 vertices u, v in G. Function P finds shortebt path from u to v in G.

```
Path (G, u, v):
    make Red graph from G
    make blue graph from G

    For i=1 to V in Red G:
        R[v] = P(s, v)        ← shortest  th from s to v
    For i=1 to v in Blue G:
        B[v] = P(t, v)        ← shortest path from v to t.
    For i=1 to V in G:
        All[v] = min ( R[v] + B[v])
    return All[v].
```

Approach:

- Create 2 graphs based on colors: Red and Blue. Red graph has red edges and Blue Graph has blue edges.
- We let the shortest path be between u and v. We can traverse from u to v through each vertex and find the path from $u_1, u_2, .. u_n \rightarrow$ to V in Blue Graph.
- Take out blue edges from G and find shortest path from s to all other vertices and store them in R[V]. Do the same for red edges and store them in B[V].
- R[V] + B[V] for all vertices of V. R[V] + B[V] will have the shortest path from s to t from v and edges from s to V are Red. And edges from v to t is blue.
- Then return the minimum of R[V] + B[V] to get shortest path.

Time Complexity

- Finding min(R[V] + B[V]) takes $O(|V|)$ time. Function P, T_p, take T_p time (assuming its constant). And calculating Red and Blue (R[V] + B[V]) takes $O(|V| + |E|)$ time. Overall, it takes $O(|V| + |E| + T_p)$ time.

Problem 2

a.

- graph G contains a path from u to v. If $u.d < v.d$ in a depth first search of G, then u is a descendant of a in the depth-first forest produced.

Lets consider 3 vertices u, v, and w.

w	1	6
u	2	3
v	4	5

There's a path from u to v. We can also see that $u.d < v.d$, but v is not a descedant of u.

b. We know that there is a path from u to v in G. Let's consider a simple example:

- Directed graph G with vertices $\{1, 2, 3\}$.
- edges $(1,2), (1,3), (2,1)$.

There exists a path from 2 to 3. But if we start DFS at 1 and process 2 before 3, it will result in $2.f = 3 < 4 = 3.d$.