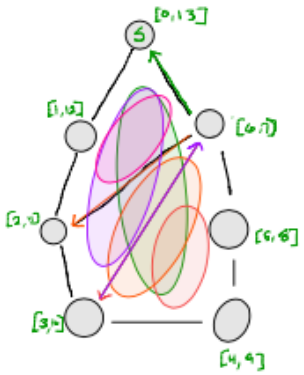
Intro to Algorithms: Homework #7

Due on March 25, 2021

Prof. Zaki

Jared Gridley

4.4) • Level of a vertex: distance in DFS Tree from root



Big Cycle:

6 - 0 + 1 = 7

Middle Cycle

6 - 2 + 1 = 5

Bottom Cycle

6 - 5 + 1 = 4

Big Return Cycle:

6 - 3 + 1 = 4

Small Return Cycle

6 - 2 + 1 = 5

So a graph with two cross edges will Break this method. this is because it does not check for inner loops, it only connects back to loops that have the source node in its path.

So this graph will break it:

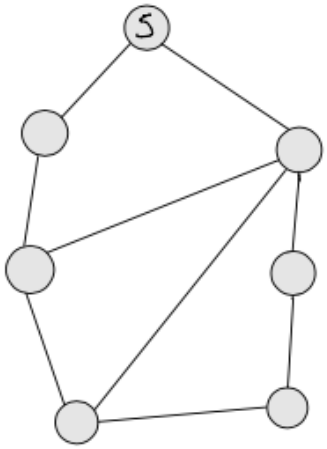


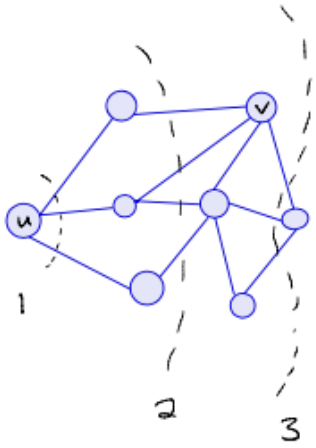
Figure 1: Page 1

4.5) Undirected Graph $G = (V, E)$, unit length edges, $u, v \in V$

↳ Number of distinct Shortest paths from $u \to v$

Tinkering:

- Breadth-First Search
  - Stop when we get to v
  - ↳ record in list of lists, the path to get to v



Best way would be to modify Dijkstra's Algorithm:

```
Shortest_Paths(G, u, v):
    paths = []
    For all x ∈ N(u):
        d(x) = 0
    dist(u) = 0
    Q = [u]
    found = False
    while Q not empty:
        if (found):
            break
        u = eject Q
        for all x ∈ N(u)
            if x == v:
                paths.append(this path)
                found = True
            if not visited:
                add v to Q, set visited flag
                d(v) = d(x) + 1
    return paths
```

```
BFS(G, u):   queue (FIFO)
    For v ∈ V:
        d(v) = ∞
    d(u) = 0          // Distance
    Q = [u]
    while Q not empty:
        u = eject Q
        for all v ∈ N(u):
            if not visited:
                add v to Q, set visited flag
                d(v) = d(u) + 1
```

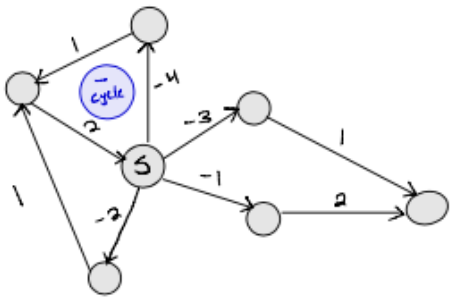
BFS

Figure 2: Page 2

4.9) Directed Graph with negative edges leaving s (only). Can Dijkstra's Algo solve this, starting at s?

Algo Dijkstra (G, s): G is a graph, s is a vertex ∈ G

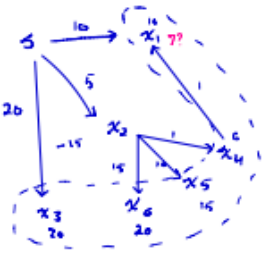
```
1   For all u ∈ V
2       dist(u) = ∞ ;
3   dist(s) = 0
4   Q = {s}
5   while Q is not empty:
6       u = Find and delete the first vertex
7       for all edges (u, v):
8           IF dist(v) > dist(u) + w(u,v):
9               dist(v) = dist(u) + w(u,v)
```



Dijkstra will fail with negative edges of any kind (Bellman-Ford solved part of this problem).

Proof by counter example:

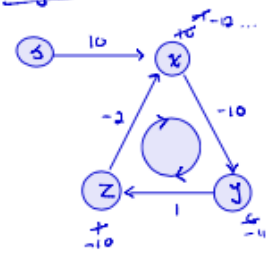
No Negative Cycles



$s \to x_1^{10}$

$s \to x_2^5 \to x_4^6 \to x_1^7$ ✓ $\Rightarrow$ Dijkstra's Fails!

Negative Cycle



Dijkstra's will be caught in an infinite loop, and will thus fail.

Dijstra's algorithm will always pick the node with the shortest path for further expansion. Thus, with a negative node, the path is possibly longer than the shortest path. Since we know that Dijkstra's doesn't work with ne

Since we have shown both examples of where it fails, we have proved that it can fail on graphs with negative edges

Figure 3: Page 3

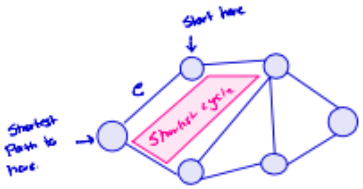
4.12) Give $O(|V|^2)$ algorithm:

- Undirected Graph $G = (V, E)$; edge len. $l_e > 0$, edge $e \in E$
  - ↳ Length of shortest cycle with e.

Thinking:

Start at e. Find cycles starting at e, including e

↳ use BFS algorithm



Algorithm (G, e): $e \Rightarrow (u, x)$

```
H = remove e from G
cycles = [ ]
For v ∈ H(V):
    d(v) = ∞
d(u) = 0
found_x = False
Q = [u]
While Q is not empty:
    u = eject Q
    if found_x:
        return cycles list
    for all v in N(u):
        if not visited:
            at v to Q, set visited to True
            d(v) = d(u)+1
        if v = x:
            add v-path to cycles list, with e back in it!
            found_x = True

return cycles.
```

BFS (G, u): queue (FIFO)

```
For v ∈ V:
    d(v) = ∞
d(u) = 0          // Distance
Q = [u]
while Q not empty:
    u = eject Q
    for all v ∈ N(u):
        if not visited:
            add v to Q, set visited flag
            d(v) = d(u) + 1
```

Figure 4: Page 3