Graph Theory

Definition.

1. *A **Graph** $G = (V, E)$, where V is a set of vertices, and $E \subseteq V \times V$ is a set of edges. Let $|V| = n$, then $0 \leq |E| \leq n^2$*

2. *A **Directed Graph** is a graph where each edge (u, v) has a direction going form u to v.*

3. *An **Undirected Graph** is a graph where edges has no direction. If (u, v) is an edge, then so is (v, u)*

4. *A **Weighted Graph** is a graph with weight function $W : E \to \mathbb{R}$*

5. *A graph is **Connected** when there is a path between every pair of vertices. $n - 1 \leq |E| \leq n^2$ and $|V| = n > 2$. If $|E|$ is close to n^2, then G is called a **dense** graph If E is much less than n^2, then G is called a **sparse** graph*

6. *A **Connected Undirected Graph** is a graph where for every pair of vertices $u, v \in E$, there is a path $p = \langle v_0, \cdots v_k \rangle$ ($(v_i, v_{i+1}) \in E$) with $v_0 = u$ and $v_k = v$*

7. *A **Weakly Connected Directed Graph** same definition as above*

8. *A **Strongly Connected Directed Graph** is a graph such that for every pair $u, v \in E$, there is a path $p : u \to v$, and a path $p' : v \to u$*

Graph Representation

1. **Adjacency List**
   (a) **size** $O(|V| + 2|E|) = O(|V| + |E|)$ for undirected graphs (each edge counted twice) or $O(|V| + |E|)$ for directed graphs
   (b) **search** $O(E)$
2. **Adjacency Matrix**
   (a) **size** $O(|V|^2)$
   (b) **search** $O(1)$

Definition. ***Breadth-First Search (BFS)***

1.
```
Function Breadth-First-Search (G, s)
    for u ∈ V \ {s} do
        discovered[u] ← False
        dist[u] ← ∞
        parent[u] ← NIL
    discovered[s] ← True
    dist[s] ← 0
    parent[s] ← NIL
    Q ← Ø be a queue
    Enqueue(Q, s)
    while Q is not empty do
        u ← Dequeue(Q)
        for v ∈ Adj[u] do
            if discovered[v] is False then
                discovered[v] ← True
                dist[v] ← dist[u] + 1
                parent[v] ← u
                Enqueue(Q, v)
```

2. ***BFS-Tree*** *is tree generated with BFS on a tree, which is also the shortest path tree for that graph*

3. ***Complexity analysis*** *Space complexity is $\Theta(|V|)$, which is just space requried for Q. Time complexity is $\Theta(|V| + |E|)$. Queue operation runs at most twice on each $v \in V$. since discovered$[v]$ is set to true as soon as v is dequeued and discovered$[v]$ is never set back to false again, so each v can be in the queue only once. So queue operations takes $O(|V|)$ in total. For each $v \in V$, we traversed $Adj[v]$, which takes $O(|E|)$ in total for traversing the adjacency list.*

4. ***Proof of correctness***

   (a) *claim 1: If $(u, v) \in E$ then $\delta(s, v) \le \delta(s, u) + 1$*

   Proof. Let $p : s \to u$ be shortest path from s to u.

   i. If p along with (u, v) is shortest path from s to v, then

   $$\delta(s, v) = \delta(s, u) = 1$$

   ii. else, there exists some $p' : s \to v$ such that

   $$\delta(s, v) < \delta(s, u) = 1$$

   □

(b) *claim 2: Upon termination of BFS, $\delta(s,u) \le dist[u]$ for all $u \in V$.*

Proof. By induction on the number k vertices discovered by the algorithm. If $k=1$, $\delta(s,s)=0=dist[s]$, holds. If $k>1$, then assume results holds for $\le k-1$ Prove it holds for kth vertex. Say v is discovered from u, then by induction hypothesis $dist[u]=\delta(s,u)$, we have

$$dist[v] = dist[u] + 1 = \delta(s,u) + 1$$

By claim 1

$$dist[v] \ge \delta(s,v)$$

□

(c) *claim 3: In any step, if the queue Q consists of $v_1, \cdots, v_k$ then*

$$dist[v_1] \le dist[v_2] \le \cdots \le dist[v_k] \le dist[v_1] + 1$$

Proof. By induction on state of queue. If $Q=\{s\}$, $dist[s] \le dist[s]+1$, claim holds. Now assume claim true upto current configuration of Q, two cases. An element is dequeued, the claim holds trivially. Then vertex v_{k+1} is enqueued to back of Q. By algorithm $dist[v_{k+1}] = dist[v_1]+1$, the claim holds. □

(d) *the algorithm is correct. Define $\delta(s,v)$ is the shortest distance of u from s in G. We claim that $dist[v]=\delta(s,v)$ for all $v \in V$.*

Proof.

i. $\delta(s,v)=\infty$ then $dist[v]=\infty$ by claim 2, claim true.

ii. $\delta(s,v) \ne \infty$, do induction on $\delta(s,v)$ If $\delta(s,v)=0$, then $s=v$, so $dist[s] = 0 = dist[v]$. Now we assume result holds for $\delta(s,v) \le k-1$, then consider a vertex w with $\delta(s,w)=k$ By claim 3, $\delta(s,w)>k$, the algorithm discovers w after discovering every $v \in V$ such that $\delta(s,v)=k-1$. Now consider parent vertex u such that $(w,u) \in E$, hence by the algorithm $dist[u]=k-1$. By induciton hypothesis, we have

$$dist[w] = dist[u] + 1 = \delta(s,u) + 1 = k - 1 + 1 = k$$

□

Definition. ***Breadth-First Search (BFS)***

1.

```
Function Depth-First-Search (G, s)
    for u ∈ V \ {s} do
        discovered[u] ← False
        parent[u] ← NIL
    discovered[s] ← True
    parent[s] ← NIL
    S ← Ø be a stack
    Put(S, s)
    while S is not empty do
        u ← Pop(S)
        for v ∈ Adj[u] do
            if discovered[v] is False then
                discovered[v] ← True
                parent[v] ← u
                Put(S, v)
```

Shortest Path Algorithm

Given a weighted directed graph with weight function $W : E \to \mathbb{R}$ A source node $s \in G$. Find the shortest path weights from s to all reachable vertices. If

$$p = \langle v_0 \overset{w_0}{\to} v_1 \cdots \overset{w_{k-1}}{\to} v_k \rangle$$

then $w(P) = \sum_{i}^{k-1} w_i$. So the shortest-path weihgt $\delta(s, v)$ is defined as

$$\delta(s,v) = \begin{cases} min\{w(P) : s \overset{p}{\to} v\} & \text{if such path exists} \\ \infty & \text{otherwise} \end{cases}$$

Possible variations

1. single source shortest path
2. single destination shortest path (Find solution to this problem if solution to first problem is known by reversing the edge directions, i.e. transpose of $G = (V, E)$ is $G^T = (V, E^T)$)
3. All pairs shortest path
4. Weights are not non-negative

Proposition. *Optimal substructure of shortest path. In other words, if* $p = \langle v_0, \cdots, v_k \rangle$ *is a shortest path from* v_0 *to* v_k*, then* $\langle v_i \cdots v_j \rangle$ *is also a shortest path from* v_i *to* v_j *or* $0 \le i \le j \le k$

Solution. $\square$

Bellman-Ford algorithm

Function `Bellman-Ford-algorithm` (G, w, s)
    for $v \in V \setminus \{s\}$ **do**
        $dist[v] \leftarrow \infty$
        $parent[v] \leftarrow NIL$
    $dist[s] \leftarrow 0$
    $parent[s] \leftarrow NIL$
    for $i = 1$ **to** $|V| - 1$ **do**
        for $(u, v) \in E$ **do**
            if $dist[u] > dist[u] + w(u, v)$ **then**
                $dist[v] \leftarrow dist[u] + w(u, v)$
                $parent[v] = w$
    for $(u, v) \in E$ **do**
        if $dist[v] > dist[u] + w(u, v)$ **then**
            return *False*
    return *True*

Note the algorithm fails, i.e. return $false$, if there is a negative-weight cycle. A negative-weight cycle C is a cycle such that $w(C) < 0$. Because each iteration over the cycle will decrement the weight by $w(C)$