1 Shortest Paths

We began our study of the following problems:

Single Pair Shortest Path
Input: A graph G. A weight $w(u,v)$ for each edge uv. Two vertices s and t
Output: The length of a shortest $s-t$ path.

Single Source Shortest Path
Input: A graph G. A weight $w(u,v)$ for each edge uv. A vertex s.
Output: The length of a shortest $s-t$ path for every vertex t of G.

All Pairs Shortest Path
Input: A graph G. A rational weight $w(u,v)$ for each edge uv.
Output: The length of a shortest $s-t$ path, for every pair s,t of vertices of G.

(The length of a path is the sum of the weights of the edges on it, if there is no $s-t$ path then the value returned by a shortest path algorithm is ∞.)

All these problems can also be considered on directed graphs where each arc uv is directed from u to v and the paths must use the arcs in the given directions.

We saw that the Longest Path Problem is the special case of the Single Pair Shortest Path Problem in which all edges have -1. Thus the general problem should be hard.

We decided to restrict our attention to directed graphs without negative cycles in the hopes that this made the problem easier.

We proved that for such directed graphs

1. Shortest paths are shortest walks
2. Every subwalk of a shortest walk is a shortest walk.
3. Every subpath of a shortest path is a shortest path.
4. The length of a shortest $s-t$ path is the length of a shortest $s-t$ walk with $|V|-1$ or fewer edges.

This last fact allowed us to solve the Single Source Shortest Path problem by computing the shortest $s-t$ walk with i edges for each i between 0 and $|V|-1$. To do so we took advantage of the following fact:

> In any graph (directed or not) if W is a walk with i edges from s to v and is a shortest such walk then if uv is the last arc of W, $W-v$ is an $s-u$ walk with $i-1$ edges and is a shortest such walk.

This fact allowed us to compute the lengths of the dxesird walks using the following dynamic programming algorithm:

We construct a table $T[0..n-1, V]$ where $T[i, v]$ is the length of a shortest $s-t$ path with exactly i edges (or ∞ if no such path exists). We also construct a table $P[0..n-1, V]$ where $P[i, v]$ is the predecessor of v on a shortest $s-v$ walk with exactly i edges.

Algorithm I (Directed)

```
For each $v \in V$ set $T[0, v] = \infty$.
Set $T[0, s] = 0$.
For $i = 1$ to $n - 1$ do
    Set $T[i, v] = \infty$.
    For each arc $uv$ do
        If $T[i-1, u] + w(u, v) < T[i, v]$ then
            set $P[i, v] = u$ and $T[i, v] = T[i-1, u] + w(u, v)$.
```

Algorithm I (Undirected)

```
For each $v \in V$ set $T[0, v] = \infty$.
Set $T[0, s] = 0$.
For $i = 1$ to $n - 1$ do
    Set $T[i, v] = \infty$.
    For each edge $uv$ do
        If $T[i-1, u] + w(u, v) < T[i, v]$ then
            set $P[i, v] = u$ and $T[i, v] = T[i-1, u] + w(u, v)$.
        If $T[i-1, v] + w(u, v) < T[i, u]$ then
            set $P[i, u] = v$ and $T[i, u] = T[i-1, v] + w(u, v)$.
```

We note that the running time of each of these algorithms is $O(|V||E|)$.

To compute, for a directed graph, a table T where $T[i, v]$ contains the shortest walks with at most i arcs and $P(i, v)$ is the predecessor of v in such a walk we only need to modify the algorithm slightly.

Algorithm II (Directed)

```
For each $v \in V$ set $T[0, v] = \infty$.
Set $T[0, s] = 0$.
For $i = 1$ to $n - 1$ do
    Set $T[i, v] = T[i-1, v]$ and $P[i, v] = P[i-1, v]$.
    For each edge $uv$ do
        If $T[i-1, u] + w(u, v) < T[i, v]$ then
            set $P[i, v] = u$ and $T[i, v] = T[i-1, u] + w(u, v)$.
```

Again the running time of this algorithm is $O(|V||E|)$.

We proved that for both algorithms $T[i, v]$ and $P[i, v]$ are indeed as claimed.

Given a directed version of Single Source Shortest Paths, we set $d(v)$ to be $T[n-1, v]$ in the second algorithm, as we proved above this is the length of the shortest $s-v$ path or ∞ if no such path exists.

If G is an undirected graph then instead of forbidding just negative cycles we must also forbid negative arcs. We obtain that for graphs without negative weight edges

1. Shortest paths are shortest walks
2. Every subwalk of a shortest walk is a shortest walk.
3. Every subpath of a shortest path is a shortest path.
4. The length of a shortest $s-t$ path is the length of a shortest $s-t$ walk with $|V|-1$ or fewer edges.

Thus, the following variant of the Algorithm II allows us to solve the Single Source Shortest Path Problem on graphs with no negative weight edges.

Algorithm II (Undirected)

```
For each $v \in V$ set $T[0, v] = \infty$.
Set $T[0, s] = 0$.
For $i = 1$ to $n-1$ do
   Set $T[i, v] = T[i-1, v]$ and $P[i, v] = P[i-1, v]$.
   For each edge $uv$ do
      If $T[i-1, u] + w(u, v) < T[i, v]$ then
         set $P[i, v] = u$ and $T[i, v] = T[i-1, u] + w(u, v)$.
      If $T[i-1, v] + w(u, v) < T[i, u]$ then
         set $P[i, u] = v$ and $T[i, u] = T[i-1, v] + w(u, v)$.
```

We set $d(v)$ to be $T[n-1, v]$ in the second algorithm, as we proved above this is the length of the shortest $s-v$ path or ∞ if no such path exists.

This algorithm runs in $O(|V||E|)$ time. We will see that we can solve this problem more quickly.

We also proved that at the end of the directed version of algorithm II, for every edge uv we had: $d(u) + c(u, v) > d(v)$. Furthermore, letting V' be the set $\{v | d(v), \infty\}$, and D be the directed graph with arc set $\{P[n-1, v]v | v \in V' - s\}$ we have that D is an arborescence containing a (unique) shortest $s-v$ path for all $v \in V'$. Finally, we had that for all $v \in V'$, $d(p(v)) + c(p(v), v) = d(v)$.

In the same vein, at the end of the undirected version of algorithm II, for every edge uv we had: $d(u) + c(u, v) > d(v)$. Furthermore, letting V' be the set $\{v | d(v), \infty\}$, and D be the directed graph with arc set $\{P[n-1, v]v | v \in V' - s\}$ we have that D is an arborescence containing a (unique) shortest $s-v$ path for all $v \in V'$. Finally, we had that for all $v \in V'$, $d(p(v)) + c(p(v), v) = d(v)$.