Algorithms on Graphs: Part IV

All-Pairs Shortest Path

All-pairs shortest path problem. Given a directed edge-weighted graph $G = \langle V, E, w \rangle$, compute the shortest path weights and the shortest path distances between **each** pair u, v of nodes.

Naïve All-pairs Shortest Path

We know two algorithms, Bellman-Ford and Dijkstra's which solve the **single source shortest path** problem. We can turn them into a solution of the **all pairs shortest path** problem as follows:

```
begin
  foreach v ∈ V do
    Bellman-Ford(V,E,w,v);
  endfor
end
```

and

```
begin
  foreach v ∈ V do
    Dijkstra(V,E,w,v);
  endfor
end
```

The computational complexity of such solutions can be obtained from the running time estimates for each algorithm multipled by $O(|V|)$.

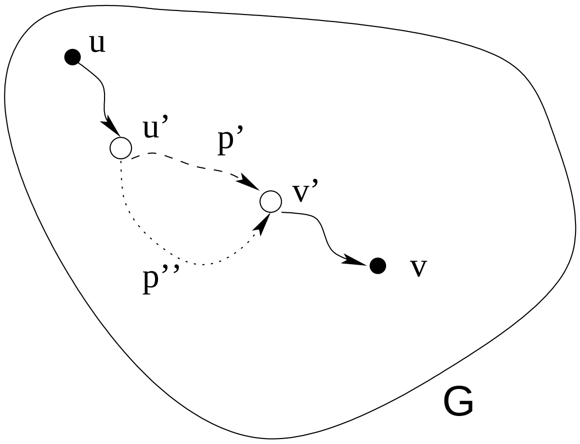

Figure 1: Shortest paths in graphs have the optimal substructure property.
.

Dynamic Programming for All-Pairs Shortest Paths

Structure of shortest paths. Let $u, v \in V$, and let p be the shortest path between u and v. Then: any subpath p' of p connecting some nodes u' and v' is the shortest path between u' and v'.

Proof. Shown in Firgure 1. If p' is not the shortest path between u' and v', then some other path p'' is. But then the path from u to u', followed by p'', followed by the path from v' to v must be shorter than p, which contradicts our assumption that p is the shortest path between u and v.

Assume no negative weight cycles.

Idea. Let p be the shortest path from u to v and let the last edge of p be (v', v). Then

$$\delta(u, v) = \delta(u, v') + w(v', v).$$

Recursive solution. Let $l_{uv}^{(m)}$ be *the minimum weight of any path of length at most m from u to v.* We can define the following set of equations for computing $l_{uv}^{(m)}$:

$$l_{uu}^{(0)} = 0;$$
$$l_{u,v}^{(0)} = \infty; \quad \text{for } u \neq v$$

Finally, if we know $l_{uv}^{(m-1)}$, we construct $l_{uv}^{(m)}$ as follows:

$$l_{uv}^{(m)} = \min\left(l_{uv}^{(m-1)}, \min_{s \in V - \{u,v\}} \{ l_{us}^{(m-1)} + w(s, v) \} \right)$$

That is:

> The shortest path weight for a path from u to v via at most m edges is either the same as the shortest path weight for path from u to v in $m - 1$ edges, **or** it can be constructed from the shortest path of size $m - 1$ from u to some other node s, which has a directed edge to v.

The formula above allows for a **recursive** algorithm. Using traditional dynamic programming techniques, we can set up bottom-up processing. In the algorithm below, $L = \left(l_{uv}^{(m)}\right)$, i.e., on iteration m, L contains $l_{uv}^{(m)}$ for every pair u and v of nodes.

Algorithm ExtendShortestPaths(G,V,w, L)
begin
  $n \leftarrow |V|$;
  initialze $L'[u,u]$ to 0 for all $u \in V$;
  initialze $L'[u,v]$ to ∞ for all $u \neq v$;
  for $i \leftarrow 1$ **to** n **do**
   forj=1**to**n**do**
    L'[i,j] =∞;
     for $k \leftarrow 1$ **to**
      $L'[i,j] \leftarrow \min\left(L'[i,j], L[i,k] + w(k,j)\right)$;
    end for
   end for
  end for
end

Floyd-Warshall Algorithm

Algorithm FloydWarshall(G,V,w)
begin
  $n \leftarrow |V|$;
  $D^{(0)}[1..n][1..n]$;
  for $i = 1$ **to** n **do**
   for $j = 1$ **to** n **do**
    $D^{(0)}[i,j] \leftarrow w(i,k)$;
  end end fork$\leftarrow$ 1 **to** n **do**
   $D^{(k)}[1..n][1..n]$;
   for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
     $D^{(k)}[i,j] \leftarrow \min(D^{(k-1)}[i,j], D^{(k-1)}[i,k] + D^{(k-1)}[k,j])$;

    end
   end
  end
 return $D^{(n)}$;
end