

Graphs: Single Source Shortest Path

- Single Source Shortest Path
- Dijkstra's algorithm

Single Source Shortest Path Problem

■ **Problem statement** Given $G = (V, E)$, graph, and a source vertex $s \in V$, find a shortest paths from s to all $v \in V - \{s\}$

■ **Ideas**

1. What if G is unweighted?

A shortest path is a path of minimal length (min number of edges).

2. What if the graph G is weighted?

The weight of p is

$$w(p) = \sum_{(u,v) \in p} w(u, v).$$

A shortest path, p , from s to d , $s, d \in V$, is a path of minimal weight, i.e.

$$w(p) \leq \text{weight of any path connecting } s \text{ to } v$$

We denote the weight of the shortest path from u to v by $\delta(u, v)$

■ Single Source Shortest Path on a Weighted Graph

- If the weights may be negative, this may create problems

What's the problem: negative-weight cycle.

- *If negative-weight cycle reachable from s , the shortest path does not exist.*

■ **Dijkstra's Algorithm:** Solves the single source shortest path assuming there are no negative-weight edges.

■ **Bellman-Ford Algorithm** : Can deal with negative edges. When negative-weight cycle is reachable it reports that no solution exists. We will not consider this algorithm.

Dijkstra's Algorithm

■ **Introduction** Greedy, similar to Prim's.

While the shortest paths are being built, a partitioning of the vertices is maintained, S and $V - S$. S is the set of vertices for which the shortest paths (from s) have been obtained already.

Priority queue, Q , is used to maintain the partition. In particular, $Q = V - S$.

For $v \in S$, keys $d[v]$ are the weights of the shortest paths from s to v in G , $\pi(v)$ is the predecessor of v on the path. For $v \in V - S$, $d[v]$ is an estimate of the shortest path.

Initially $S = \emptyset$. First s is moved from V to S . At each step, $v \in V - S$ that is *closest* to S is moved to S .

Next shortest paths, from s , to all vertices in the subgraph induced by S are updated.

At termination, $d[v]$ and $\pi[v]$ define the shortest paths weights and the edges composing the paths. If $d[v] = \infty$ — unreachable.

■ Dijkstra's algorithm

Note that

Prim's keys, $key[v] = w(u, v)$, for safe (u, v) .

Dijkstra's keys, $d[v] = d[u] + w(u, v)$, where $d[u] = \delta(s, u)$.

Dijkstra(G, w, s)

 for each $v \in V$

$d[v] = \infty$

$\pi[v] = \text{nil}$

$d[s] = 0$

MakePriorityQueue(Q, V)

 \\construct the shortest paths

 while *not Empty*(Q)

$u = \text{ExtractMin}(Q)$

 \\update Q keys

 for each $v \in \text{Adj}[u]$

 if $d[v] > d[u] + w(u, v)$

 \\ set $d[v] = d[u] + w(u, v)$, $\pi[v] = u$

DecreaseKey($Q, w(u, v), v, u$)

 return π, d

■ Example

Dijkstra's Algorithm (*continued*)**■ Time complexity** Just like Prim's.

For binary heap implementation $O(m \log n)$.

For implementation using *Fibonacci heaps* it can be shown that the time is $O(m + n \log n)$ which is better (linear in m).

If we run Dijkstra's algorithm n times, once from each vertex, we get $O(nm + n^2 \log n)$ for *all-pairs-shortest-paths* problem in a graph with nonnegative-weight edges.

■ Correctness We claim that in Dijkstra's alg, $d[v] = \delta(s, v)$ for $v \in S$.

Proof by induction on $|S|$.

Base: $S = \{s\}$. Since $d[s] = 0$, true.

Ind. Hyp: Assume that at each step, up to i , we have maintained S s.t. $d[u] = \delta(s, u)$, $u \in S$.

Dijkstra's Algorithm (*continued*)

Ind. Step: Consider the next $i + 1$ step.

Let v be the vertex extracted from Q . We know that v is *closest* to S , i.e. if $u = \pi(v) \in S$,

$$d[v] + w(u, v) \leq d[a] + w(a, b),$$

for any $a \in S$ and $b \in V - S$, where $(a, b) \in E$.

Since $u \in S$, by ind hyp, let p be the shortest path from s to u , $d[u] = \delta(s, u)$.

We want to show that $p \cup (u, v)$ is a shortest path from s to v .

Assume the contrary, i.e., there exist p^* connecting s to v , $w(p^*) = \delta(s, v) < w(p)$.

Since p^* connects S to $V - S$, there must be an edge $(u_1, v_1) \in P^*$, $u_1 \in S, v_1 \in V - S$.

Then, we can break the path $p^* = p_1 \cup (u_1, v_1) \cup p_2$.

Dijkstra's Algorithm (*continued*)

By ind hyp $d[u_1] = \delta(s, u_1)$, and since the weights are nonnegative,

$$\begin{aligned} d[u_1] + w(u_1, v_1) &\leq w(p_1) + w(u_1, v_1) \\ &\leq w(p_1) + w(u_1, v_1) + w(p_2) \\ &= w(p^*), \text{ and by assumption} \\ &< d[u] + w(u, v) \end{aligned}$$

If this was true u could not have been on the top of the priority queue, since u and u_1 are both in S , and the priority queue is organized in such a way that the vertex on the top is one *closest* to S . That is, $d[v] + w(u, v)$ is minimal, compared to all $d[a] + w(b, a)$, where $a \in V - S$ and $b \in S$. This contradiction completes the proof.