Chapter 7 : Shortest Path Problem - Dijkstra's Algorithm

Omar Saadi
omar.saadi@um6p.ma

UM6P - College of Computing

1st year of the engineering degree of UM6P-CS
Academic year 2024-2025

Outline

Outline

Shortest path problem

Some real-world problems :

- How can you find the shortest routes from your home to every place in town ?
- How to find the cheapest flight(s) from Casablanca to Hong Kong allowing stopovers ?
- How to find the fastest driving route (Google Maps) ?
- The internet : how to find the fastest path to send information through a network of routers ?

All these problems can be modeled using weighted (di)graphs, and solving the shortest path problem.

Framework

- We have a graph (or digraph) $G = (V, E)$.
- Each edge $(u, v) \in E$ has a **nonnegative** weight $w(u, v)$ (that represents a distance, a cost, a time, ...).
- We have a starting vertex s and a destination vertex d.

Shortest path problem

We want to solve the following problem :

$$\underset{P \text{ a } s,d\text{-path in } G}{\text{minimize}} \; w(P),$$

where $w(P)$ is the weight of the path P given as the sum of the weights of the edges forming it.

Outline

Dijkstra's Algorithm

Input : A graph (or digraph) with nonnegative edge weights and a starting vertex s.

Output : Distance $d(s,t)$ from s to each other vertex t and a shortest path tree given by identifying the parent $P(u)$ of each vertex u.

Key idea : $\forall t \in V$, the algorithm computes an estimate $d[t]$ of the distance of t from the source s such that :

1. At any point in time, $d[t] \geq d(s,t)$, and
2. when t is finished, $d[t] = d(s,t)$.

Let $w(x,y) = +\infty$ if (x,y) is not an edge.

Dijkstra's Algorithm

Idea : Maintain the set S of vertices to which a shortest path from s is known, enlarging S to include all vertices.
To do this, maintain a tentative distance $d[t]$ from s to each $t \notin S$, being the length of the shortest s,t-path yet found.

Initialization : Set $S = \{s\}; d[s] = 0; P(s) = \text{NONE}$. For $t \neq s$, set $d[t] = w(s,t)$ and if $w(s,t) \neq +\infty$ then take $P(t) = s$.

Iteration : Select a vertex $x \notin S$ such that $d[x] = \min_{t \notin S} d[t]$. Add the vertex x to S.
For each edge $(x,y) \in E$ such that $y \notin S$: update $d[y]$ to $\min\{d[y], d[x] + w(x,y)\}$.
If $d[y]$ changes, then update $P(y)$ to x.
The iteration continues until $S = V(G)$ or until $d[y] = +\infty$ for every $y \notin S$.
At the end, set $d(s,x) = d[x]$ for all $x \in V$.

Lemma 1

For every u, at any point of time $d[u] \geq d(s,u)$ and $d[u]$ represents the length of some path from s to u.

Proof

We proceed by induction over the iterations of the algorithm.

Base case : we know that $d[s] = 0 = d(s,s)$ and for each vertex $t \neq s$ we have $d[t] = w(s,t) \geq d(s,t)$, so we know that the claim holds initially.

Induction : when $d[u]$ is changed to $d[x] + w(x,u)$ then (by the induction hypothesis) there is a path from s to x of weight $d[x]$ (that does not contain u) and an edge (x,u) of weight $w(x,u)$. This means there is a path from s to u of weight $d[u] = d[x] + w(x,u)$. This implies that $d[u]$ is at least the weight of the shortest path $= d(s,u)$.

Lemma 2

The assertion $d[x] \leq d[y]$ for all $y \notin S$ stays true at all points after x is inserted into S.

Proof

We take $F = V \setminus S$. By absurd, assume that at some point for some $y \in F$ we get $d[y] < d[x]$ and let y be the first such y.
Before $d[y]$ was updated $d[y'] \geq d[x]$ for all $y' \in F$.
But then when $d[y]$ was changed, it was due to some neighbor y' of y in F, such that $d[y] = d[y'] + w(y', y) \geq d[y'] \geq d[x]$, so we get a contradiction.

Proposition

When vertex x is placed in S, $d[x] = d(s, x)$, and backtracking from x to s using the parent list P yields a shortest path from s to x.

Proof

We take $F = V \setminus S$. We do an induction on the order of placement of vertices into S. For the base case, s is placed into S where $d[s] = d(s, s) = 0$ and $P(s) = \text{NONE}$, so initially, the claim holds.
Inductive step : we assume that for all vertices y currently in S, $d[y] = d(s, y)$ and P yields shortest paths for them.
Let x be the vertex that will be added to S, i.e. satisfying $d[x] = \min_{t \notin S} d[t]$.
Let p be a shortest path from s to x. Suppose z is the vertex on p closest to x for which $d[z] = d(s, z)$. We know z exists since there is at least one such vertex, namely s, where $d[s] = d(s, s)$. By the choice of z, for every vertex y on p between z (not inclusive) to x (inclusive), $d[y] > d(s, y)$.

Proof (Cont.)

We have the following two options for z :

1. If $z = x$, then $d[x] = d(s, x)$ and we are done.
2. Suppose $z \neq x$. Then there is a vertex z' after z on p. We know that $d[z] = d(s, z) \leq d(s, x) \leq d[x]$.
   Finally, towards a contradiction, suppose $d[z] < d[x]$.
   By the choice of $x \in F$ we know $d[x] = \min_{t \in F} d[t]$. Thus, since $d[z] < d[x]$, we know $z \in S$.
   This means the edges out of z, and in particular (z, z'), were already considered by our algorithm.
   But this means that $d[z'] \leq d(s, z) + w(z, z') = d(s, z')$, because z is on the shortest path from s to z'. However, this contradicts z being the closest vertex on p to x meeting the criteria $d[z] = d(s, z)$. Thus, the assumption $d[z] < d[x]$ is false and $d[x]$ must equal $d(s, x)$.

Note that updating the parent $P(y)$ whenever $d[y]$ is improved insures that at the end P contains the parents in shortest paths with lengths $d[x] = d(s, x)$ for all $x \in V$.