Single-source shortest paths problem

Overview:

- Weighted Graphs
- General Approach ✔️
- Negative Edges
- Optimal Substructure : technique that most shortest path algorithms use to get efficient complexity

Motivation:

Shortest way to drive from A to B

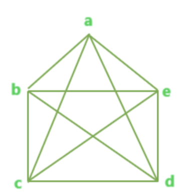
Weighted graph **G(V, E, W)**,   W: E -> R (set of real number)

G: Graph, V: Vertices, E: Edge, W: Weight

Two Algorithms:

1) Dijkstra : + weight edges, **O(V lg V + E)** practically linear time, dominated in many cases by E,

   E = O(V^2) ex. complete graph V=n, E = n(n-1)/2



2) Bellman Ford : -/+ edges, **O(VE)** ** have to find cycle

Definitions:

$$\text{path } p = < v_0, v_1, \ldots v_k >$$
$$(v_i, v_{i+1}) \in E \quad \text{for} \quad 0 \leq i < k$$
$$w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$$

** path의 weight w(p)는 path에 존재하는 모든 edge의 weight의 합이다.

** shortest path problem: find p with minimum weight

complexity는 weight와 무관하다.

Weighted Graphs:

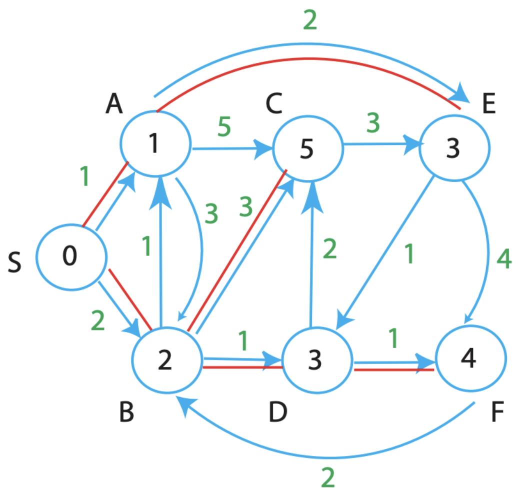
$$v_0 \xrightarrow{p} v_k$$

v(0) is a path from v0 to v0 of weight 0

Shortest path weight from u to v as

$$\delta(u,v) = \begin{cases} \min\left\{w(p) : u \xrightarrow{p} v\right\} & \text{if } \exists \text{ any such path} \\ \infty & \text{otherwise} \quad (v \text{ unreachable from } u) \end{cases}$$

example:



d(u) : current weight
(for initial) d(S) = 0, d(A), ... = Infinite

(1) d(A) = 1, d(B) = 2

(2) d(C) = 6 : S > A > C , d(D) = 3

(3) d(C) = 5: S > B > C

Predecessor relationship:

$$d[v] \quad = \quad \text{value inside circle}$$ current weight, at end $\delta(s,v)$

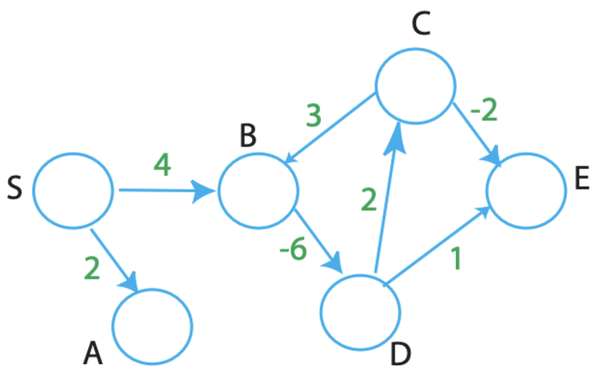
$$\Pi[v] = \text{predecessor on best path to } v,\ \Pi[s] = \text{NIL}$$

$$\begin{aligned} d[v] &= \text{value inside circle} \\ &= \left\{ \begin{array}{ll} 0 & \text{if } v = s \\ \infty & \text{otherwise} \end{array} \right\} \Longleftarrow \quad \text{initially} \\ &= \delta(s,v) \Longleftarrow \quad \text{at end} \\ d[v] &\geq \delta(s,v) \quad \text{at all times} \end{aligned}$$

Negative-Weight Edges:

Motivation? Reverse tall , Social networks (like, dislike)

Negative cycle is a problem!



delta(s,s): 0, delta(s,a) : 2, delta(s, b) => negative infinite

Bellman Ford: detect negative weight cycle, termination condition

General structure of S.P Algorithm(no negative cycles)

Initialize: for $v \in V$: $d[v] \leftarrow \infty$
$\Pi[v] \leftarrow$ NIL

$d[S] \leftarrow 0$

Main: repeat

select edge (u, v) [somehow]

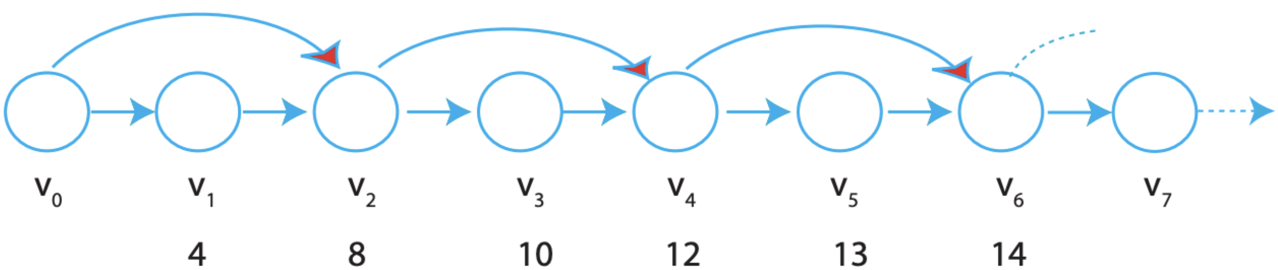
"Relax" edge (u, v)

- if $d[v] > d[u] + w(u, v)$:
  - $d[v] \leftarrow d[u] + w(u, v)$
  - $\pi[v] \leftarrow u$

until all edges have $d[v] \leq d[u] + w(u, v)$

Termination?

Could be **exponential** time with poor choice of edges



take $O(2^{\frac{n}{2}})$

Optimal Substructure:

Theorem: Subpaths of shortest paths are shortest paths

Let $p = < v_0, v_1, \ldots v_k >$ be a shortest path
Let $p_{ij} = < v_i, v_{i+1}, \ldots v_j > \quad 0 \leq i \leq j \leq k$

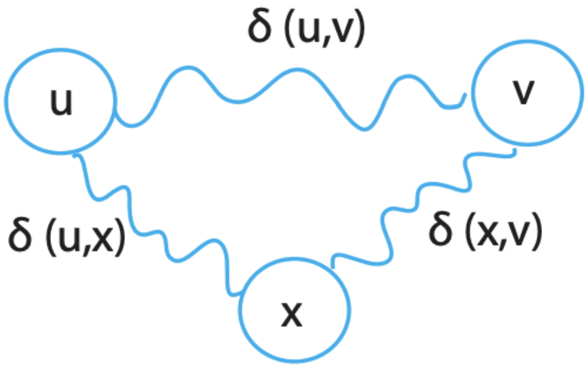
Then p_{ij} is a shortest path.

Proof:

$$p = v_0 \xrightarrow{p_{0,i}} v_i \underset{p'_{ij}}{\overset{p_{ij}}{\rightrightarrows}} v_j \xrightarrow{p_{jk}} v_k$$

만약 **Pij > P'ij** 라면, P안에 있는 Pij를 P'ij로 대체하게 되면 새로운 값은 P보다 작아지는 모순이 생김

Theorem: Triangle Inequality, For all u, v, x ∈ X, we have δ (u, v) ≤ δ (u, x) + δ (x, v)


δ (u,v)
u
v
δ (u,x)
δ (x,v)
x