Jay Urbain

Shortest Paths Problem

Naive Method - enumerate all paths, add up distances, and select shortest

→ enormous number of possibilities, even if you disallow cycles.

Shortest Paths Problem -

- Given a weighted, directed graph $G = (V, E)$ w/ weight function $w: E \to R$

- The weight $w(p)$ of path $p = \langle v_0, v_1, \dots, v_k \rangle$ is the sum of its constituent edges

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- Shortest path weight: $\delta(u, v)$ from u to v:

$$\delta(u, v) = \begin{cases} \min\{ w(p) : u \overset{p}{\leadsto} v \} & \text{if there is a path} \\ \infty & \text{otherwise} \end{cases}$$

Dijkstra's Algo.

- Solves single-source shortest-paths problem on a weighted directed graph $G = (E, V)$ for non-negative weights.

- Maintains a set S of vertices whose final shortest-path weights from the source $\underline{s}$ have already been determined.

- Algo repeatedly selects vertex $u \in V - S$ w/ the minimum shortest path estimate, adds u to S, and relaxes all edges leaving u.

- Use min. priority queue Q of vertices keyed by their d values.

$s.d = \emptyset$
$s.\pi = NIL$

$\emptyset$ NIL

```
Dijkstra(G, w, s)
    initializeSingleSource(G, s)   // init d & π values.
    S = ∅               //{empty set}
    Q = G.V             { init min-pri-queue w/ all vertices
    while Q ≠ ∅         // maintain loop invariant Q = V - S
        u = ExtractMin(Q)      { s is first vertex
        S = S ∪ {u}
        for each vertex v ∈ G.Adj[u]
            Relax(u, v, w);  {
```

- first time through loop $u = s$

"relax each edge leaving u, thus updating estimate $v.d$ & $v.\pi$ if we can improve the shortest path to v found so far through u."

- Each vertex is extracted from Q and added to S exactly once.

- Because Dijkstra's algo. always chooses the "lightest" or "closest" ~~edge~~ vertex in $V - S$ to add to set S, it uses a Greedy strategy.

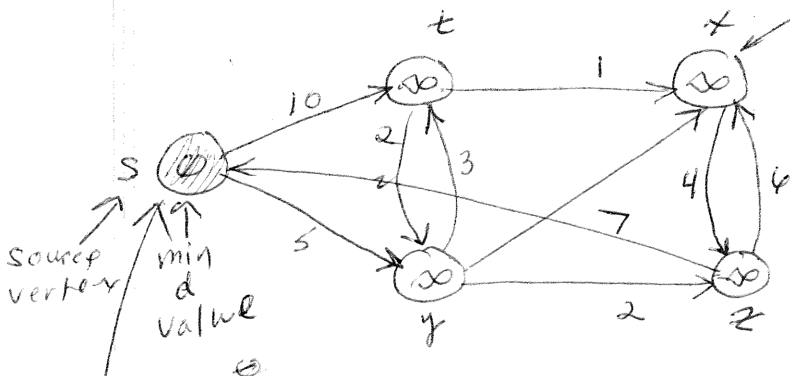
- Key is to show that each time it adds a vertex u to set S, we have $\underline{u.d = \delta(s, u)}$.

Relaxation

- Algo uses "relaxation"
- For each vertex $v \in V$, we maintain attribute $v.d$, wh/ is an upper bound on the weight of the shortest path from source s to v
- The process of relaxation, consists of testing whether we can improve the shortest path to v found so far by going through u.

```
Relax (u, v, w)
    if  v.d > u.d + w(u,v)
        v.d = u.d + w(u,v)
        v.π = u.
```

shortest path estimates appear in vertices



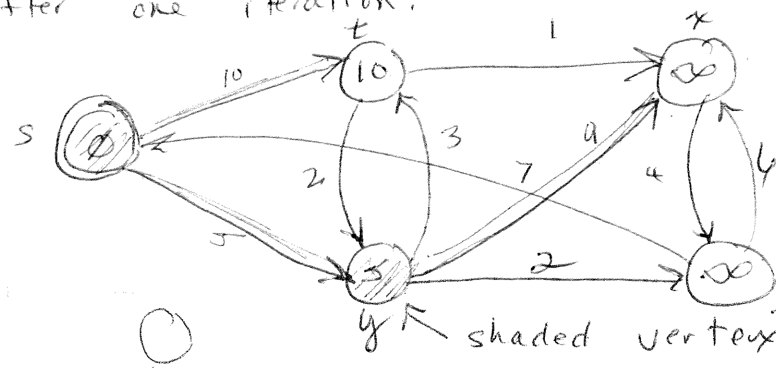
- shaded edges indicate predecessor values

- black vertices are in set S

- white vertices in Q = V − S

- s chose as min. d value from Q (line 5)

- After one iteration:

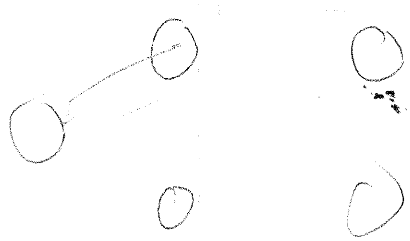


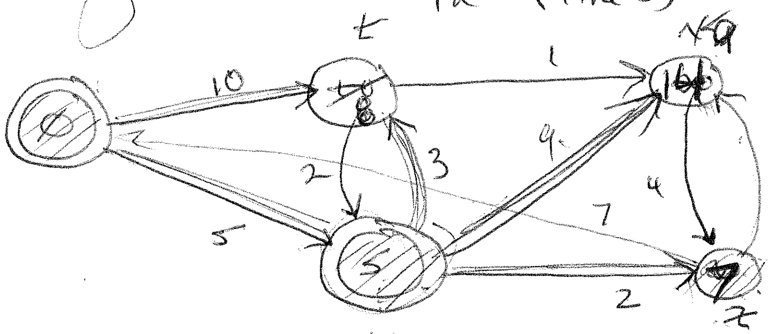
Q 10 t

Q:

5	10	∞	∞
y	t	x	z

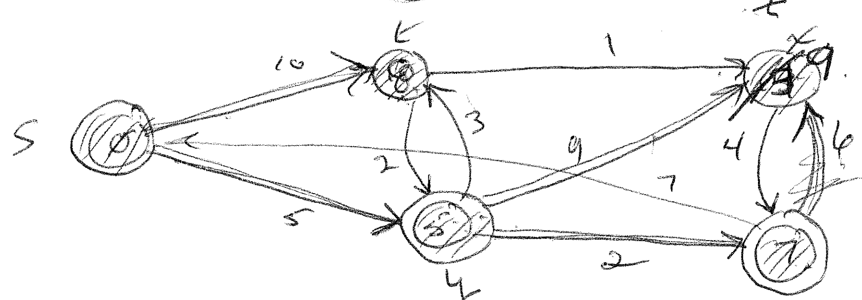
shaded vertex has minimum d value & is chosen as vertex u in (line 5)





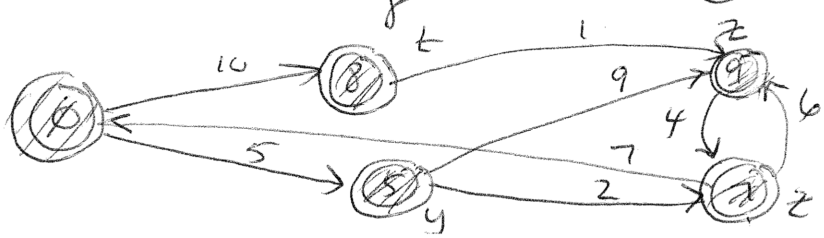
Q

7	8	14
z	t	x



Q

8	13
t	x



Q

9
x

Dijkstra – Proof of Correctness

- key is to show that each time it adds a vertex u to set S, we have $u.d = \delta(s,u)$.

Proof We use loop invariant:

At the start of each iteration of the while loop of lines 4-8, $v.d = \delta(s,v)$ for each vertex $v \in S$.

It suffices to show that for each vertex $u \in V$, we have ~~u ∈ V~~ $u.d = \delta(s,u)$ at the time when u is added to set S.

Analysis

$O(E \lg V$.

Proof

- Loop invariant: At the start of each iteration of the while loop (4-8) $v.d = \delta(s, v)$ for each $v \in S$

- Suffices to show for each vertex $u \in V$, $u.d = \delta(s,u)$ at the time u is added to S.
- once we show $u.d = \delta(s,u)$, we rely on upper-bound property to show that the equality holds at all times thereafter

Init: $S = \emptyset$, invariant is trivially true

Maintenance: For each iteration, $u.d = \delta(s,u)$ for the vertex added to set S

- For purposes of contradiction, let u be the first vertex added to S where $u.d \neq \delta(s,u)$
- Must have $u \neq s$, because s is first vertex added to set S and $s.d = \delta(s,s) = \emptyset$
- Because $u \neq s$ we also have $S \neq \emptyset$ before u is added to S.
- There must be at least one path from s to u otherwise $u.d = \delta(s,u) = \infty$ by no path property whl would violate our assumption that $u.d \neq \delta(s,u)$

Analysis

Maintain min-pri-queue Q

$\lg V$ $O(V^2 + E)$ $- O(V^2$ $(E \lg V)$

$O(1)$	insert	- once per vertex
$O(V)$	extract_min	- once per vertex
$O(1)$	decrease_key (implicit in	

- because each vertex $u \in V$ is added to S exactly once, each edge in adjacency list Adj[u] is examined in for loop exactly once.

- Since total # edges is |E| for loop iterates |E| times and thus decrease key is called |E| times overall

- Running time depends on implementation of mpq.