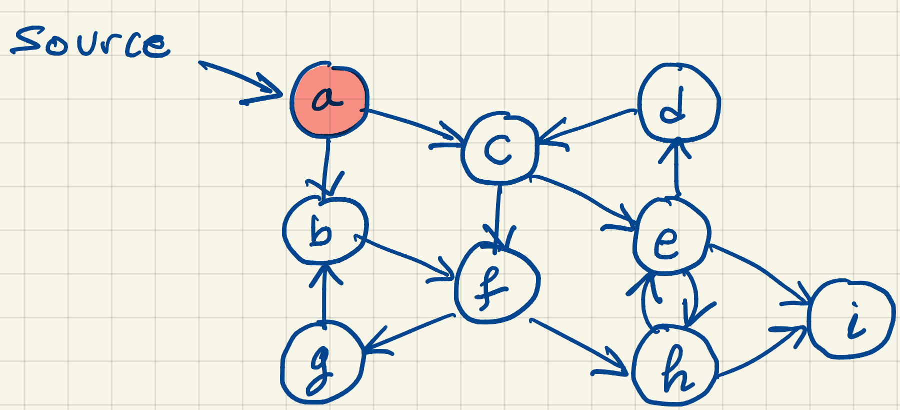
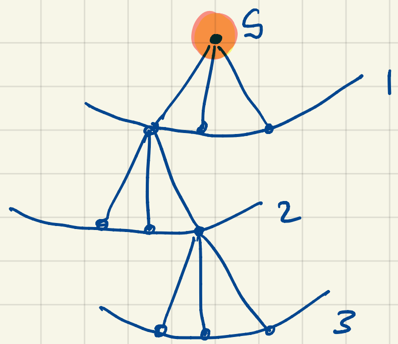
BFS and Shortest path

Breadth First Search produces a tree rooted at a source node s such that path from s to v in tree is shortest path from s to v in G.

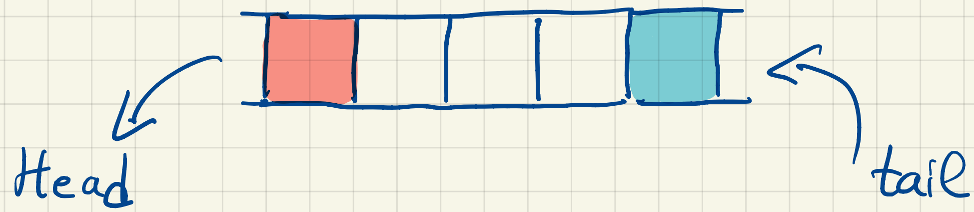


Note: Graph could be directed or undirected

Idea: Go by breadth, propagate a wave of distance 1



Use FIFO queue to process vertices



BFS(s)
- for each $u \in V - \{s\}$
  - do $d[u] \leftarrow \infty$
- $d[s] \leftarrow 0$
- $Q \leftarrow \{s\}$
- while $Q \neq \phi$
  - do remove u from Q
    - for each $v \in adj[u]$
      - do if $d[v] = \infty$
        - then $d[v] \leftarrow d[u] + 1$
        - put v in Q

adding/removing
can be done in $O(1)$ time

(Keep Head & Tail pointers)

code not updating
parent pointers
as in book

Running time: $O(V+E)$ (linear)

Example:

~~a~~

~~b~~ c

~~c~~ f

~~f~~ e

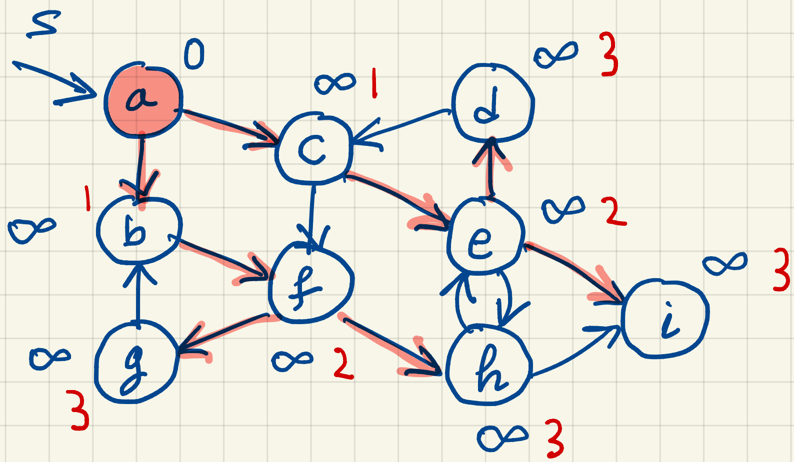
~~e~~ g h

~~g~~ h d i

~~h~~ d i

~~d~~ i

~~i~~



BFS may not reach all vertices

But that's ok since distance would be ∞

Proof of correctness ?

Similar to alg. with weighted edges, Dijkstra's Alg. (Later)

Generalize to weighted Graphs

- Given $G = (V, E)$ and a weight function $w: E \to \mathbb{R}$

  (BFS: $w(e) = 1$ for all $e \in E$)

- The weight of a path $p = v_0 \to v_1 \to \cdots \to v_k$ is

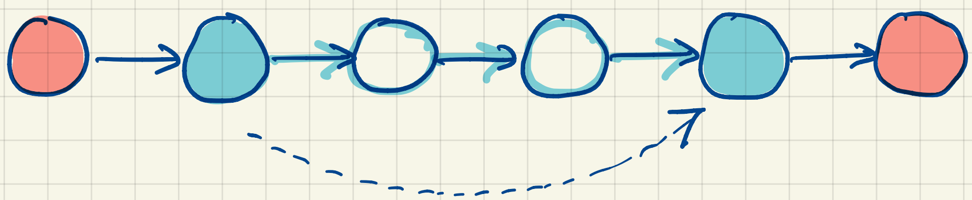
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- Shortest path means path with min. weight.

Properties of shortest path:

- Optimal substructure (greedy and DP later)

  subpath of shortest path are shortest path



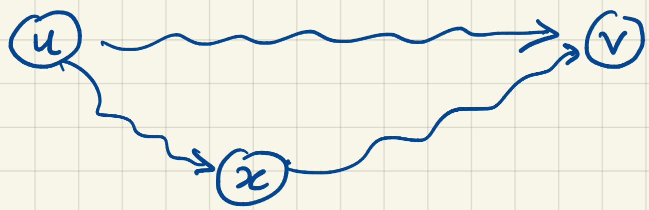
Proof: Cut-and-paste: If some subpath were not a shortest path, we could substitute the shorter subpath and create a shorter total path

- **Triangular Inequality:**

Define $\delta(u,v)$ = weight of shortest path from u to v.

$$\delta(u,v) \le \delta(u,x) + \delta(x,v)$$

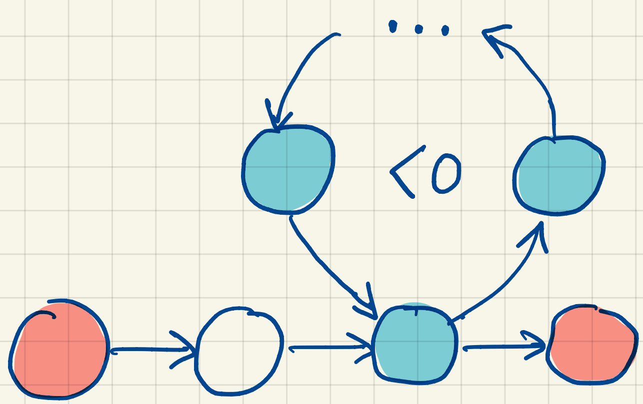
<u>Proof:</u>



Shortest path $u \rightsquigarrow v$ is not longer than any other path.

- **Well_definedness:**

If we have negative weight cycle in graph $\Rightarrow$ some shortest path may not exist. (go around the cycle again)



Most basic Algorithm: Bellman_Ford

for each $v \in V$ Initialization

    do $d[v] \leftarrow \infty$

$d[s] \leftarrow 0$

Relaxations

for $i \leftarrow 1$ to $|V|-1$

    do for each edge $(u,v) \in E$

$$\left.\begin{array}{l}\text{do if } d[v] > d[u] + w(u,v) \\ \quad \text{then } d[v] \leftarrow d[u] + w(u,v)\end{array}\right\} \text{Relax}(u,v)$$

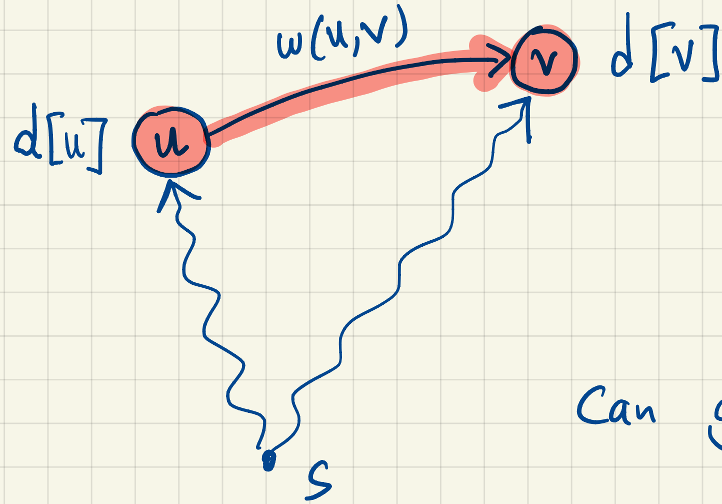
for each edge $(u,v) \in E$ Check

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

        then No solution (negative weight cycle)

Time $O(VE)$

Relaxation

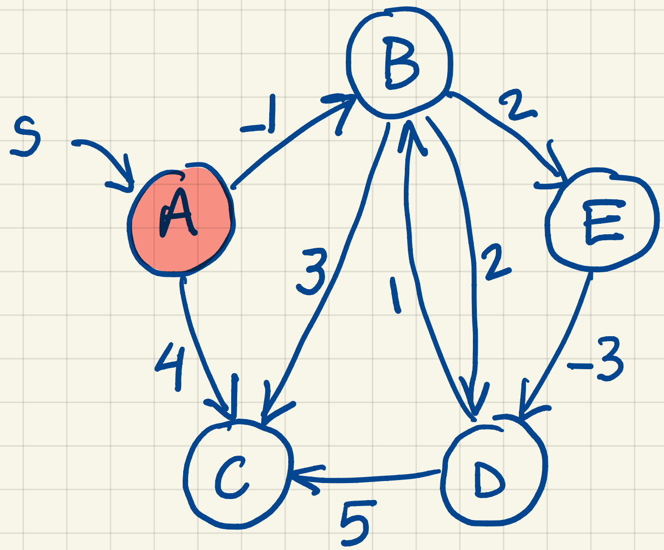


Can get better path from s to v by going through u.

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

Relax all edges $|V| - 1$ times.

Example:



	A	B	C	D	E
Init	0	∞	∞	∞	∞
Pass 1	0	−1	2	1	1
Pass 2	0	−1	2	−2	1

⋮

(no more changes)

Relax edges in this order:

(A,B) (A,C) (B,C) (B,D) (D,B) (D,C) (E,D) (B,E)

How fast do we converge? Depends on order of relaxations.

But after $|V|-1$ passes, we will (no negative weight cycle)

Why does Bellman Ford Converge ?

First, Lemma: $d[v] \geqslant \delta(s,v)$ at all times.

Assume v is first to violate above property,

so $d[v] = d[u] + w(u,v)$ (u caused d[v] to change)

$$d[v] < \delta(s,v)$$

$$\leqslant \delta(s,u) + w(u,v)$$ [triangular inequality]

$$\leqslant d[u] + w(u,v)$$ [v first to violate property]

Consider the shortest path from s to v

$$s \to v_1 \to v_2 \to \cdots \to v$$

- Initially $d[s] = 0$ is correct, and will never change

$*$ previous lemma $d[v] \geq \delta(s,v)$ $*$ code never increases d

- After 1 pass through edges, $d[v_1]$ will be set to $d[s] + w(s, v_1)$ and it will be the correct $\delta(s, v_1)$ (optimal substructure), and will never change.

- After 2 passes through edges, $d[v_2]$ will be set to $d[v_1] + w(v_1, v_2)$ and it will be the correct $\delta(s, v_2)$ (optimal substructure), and will never change.

$\ddots$

No negative weight cycle $\Longrightarrow$ every shortest path has at most $|V| - 1$ edge.

Dijkstra's algorithm

- No negative edge weights
- Perform 1 Pass by figuring out a good relaxation order
- Use a priority queue (min queue) like Prim's alg.

Main Idea:

vertex with smallest dist. so far is correct. Remove it from Q and relax its edges

Dijkstra(G)
  for each $v \in V$
    do $d[v] \leftarrow \infty$
  $d[s] \leftarrow 0$
  $S \leftarrow \phi$
  $Q \leftarrow V$
  while $Q \neq \phi$
    do $u \leftarrow$ Extract_Min(Q)
      $S \leftarrow S \cup \{u\}$
      for each $v \in adj[u]$
        do if $d[v] > d[u] + w(u,v)$
          then $\underbrace{d[v] \leftarrow d[u] + w(u,v)}_{\text{Decrease-Key}}$

Queue Operations

$O(V)$ inserts

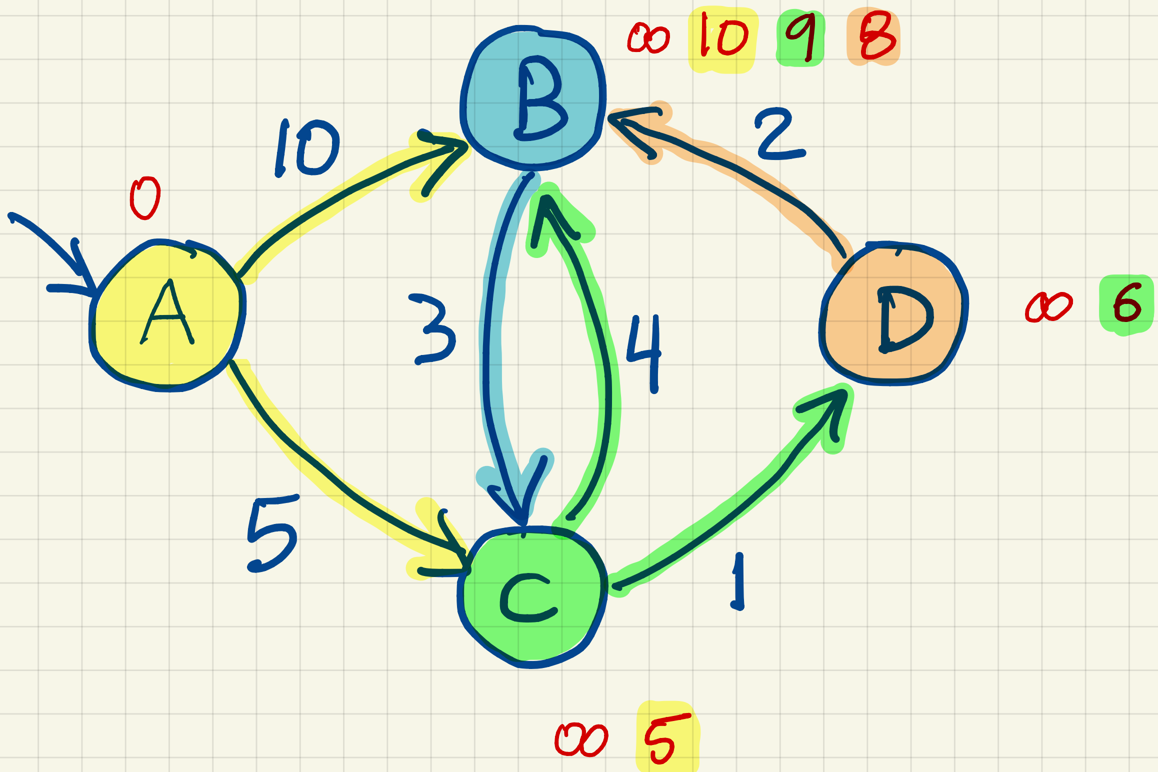
$O(V)$ Extract_Min

$O(E)$ Decrease-Key

Running time:

like Prim's

Example: Run it.

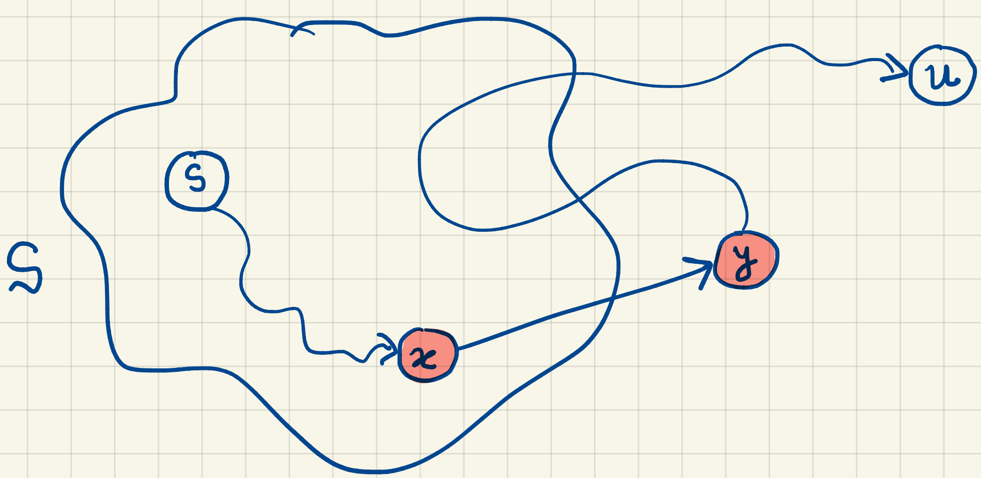


Why does it work?

As before $d[v] \geq \delta(s,v)$ (still just doing relaxations)

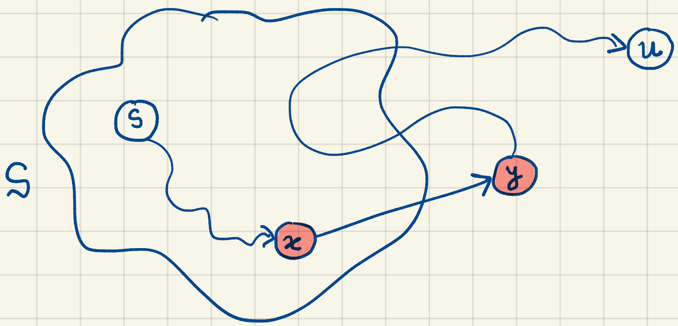
Correctness: When u added to S, $d[u] = \delta(s,u)$

Proof by contradiction: Assume u is first to violate above and look at situation just before adding u to S



- There must be a path from s to u; otherwise $\delta(s,u) = \infty = d[u]$

- Pick shortest path which crosses S with edge (x,y)
(s could be x, and y could be u)



Claim:

$d[y] = \delta(s,y)$

- $s \rightsquigarrow y$ is subpath of shortest path
- $d[x] = \delta(s,x)$ ($x \in S$, u is first to violate this)
- $d[y] = d[x] + w(x,y)$ (when edge (x,y) relaxed)

Now: $d[u] \neq \delta(s,u) \Rightarrow$

$$d[u] > \delta(s,u) \quad \text{(Lemma)}$$

$$= \delta(s,y) + \underbrace{\delta(y,u)}$$

$$= d[y] + (\geq 0) \quad \text{(no negative weights)}$$

$$\geq d[y] \quad ,$$ contradicts moving u to S.