Shortest paths — from Chapter 25 introduction and Section 25.1

- *Generalization of breadth-first-search (from last lecture) to handle weighted graphs.*
- Directed graph $G = (V, E)$, weight function $w : E \to \mathbf{R}$
  (BFS — $w(e) = 1 \ \forall e$)
- e.g. street map (with distances on roads between intersections)
- Weight of 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 = path of minimum weight

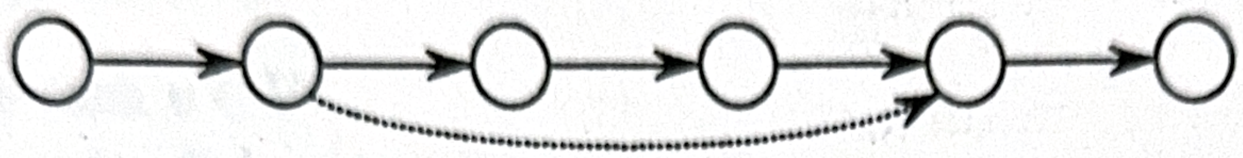
- **Optimal substructure**

  (Thus, will see greedy and dynamic-programming algorithms.)

  Theorem: Subpaths of shortest paths are shortest paths.

  ⟨⟨Lemma 25.1, with both statement and proof in English rather than in mathematics.⟩⟩

  Proof: By "cut and paste"



  If some subpath were not a shortest path, could substitute the shorter subpath and create a shorter total path.

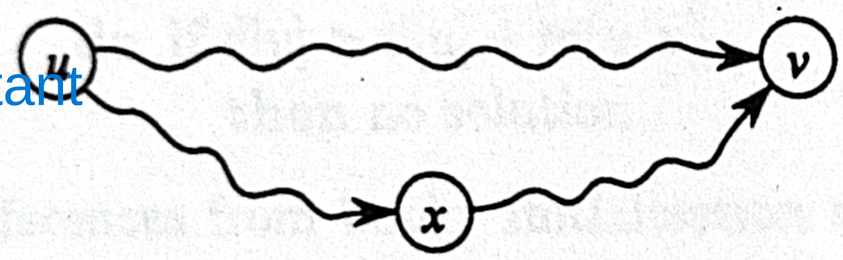
- Def: $\delta(u, v) =$ weight of a shortest path from u to v
- **Triangle inequality**

  Theorem: $\delta(u, v) \leq \delta(u, x) + \delta(x, v)$

  ⟨⟨Generalization of Lemma 25.3⟩⟩

  Proof:

Definition Important

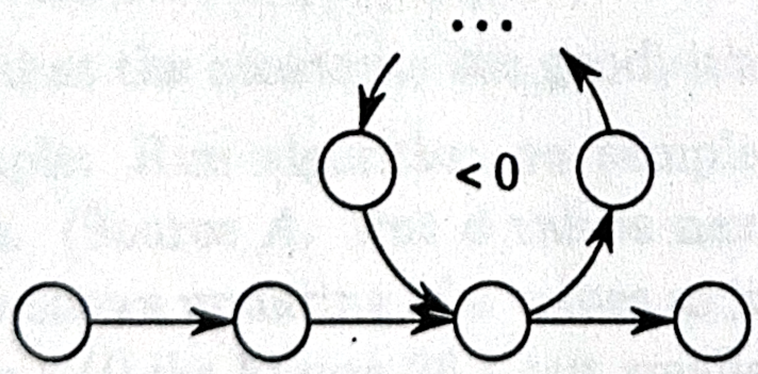


  Shortest path $u \rightsquigarrow v$ is no longer than any other path — in particular, the path that takes the shortest path $u \rightsquigarrow x$, then the shortest path $x \rightsquigarrow v$.

- **Well-definedness**

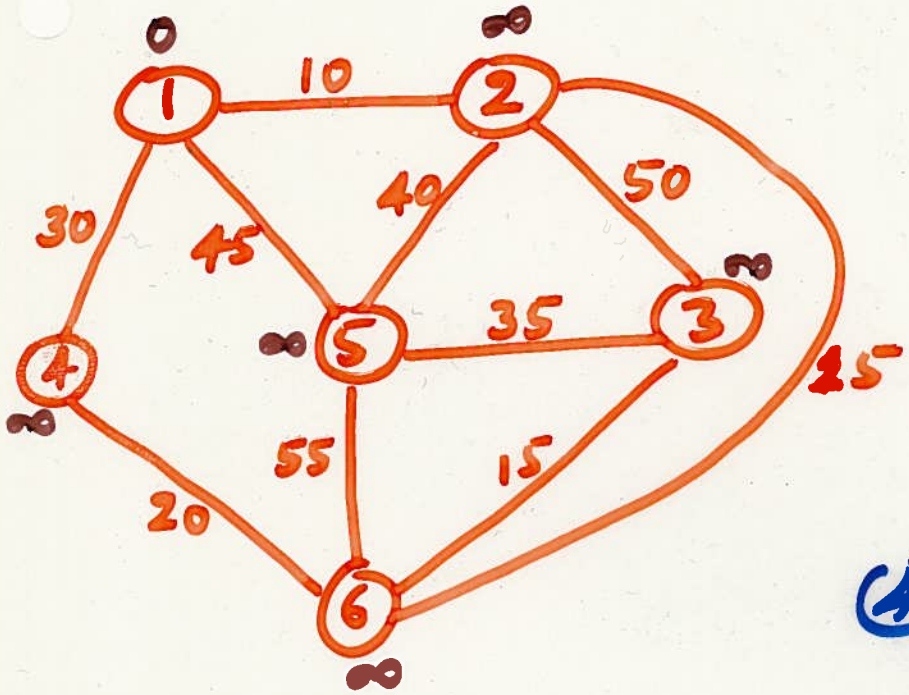
  Negative-weight cycle in graph ⇒ some shortest paths may not exist.

  Argument: Can always get a shorter path by going around the cycle again.

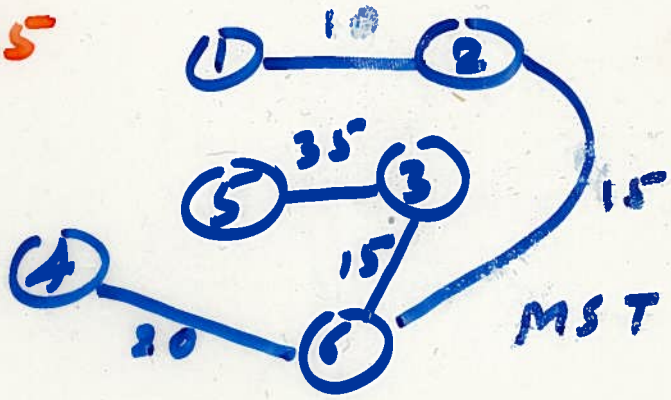


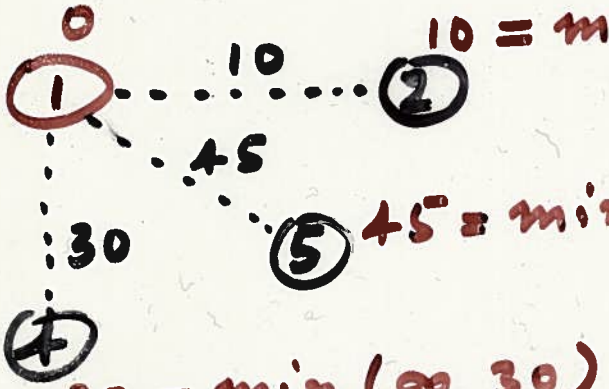
- (Above were high points of theory — rest of theory is in book.)

SHORTEST PATH



Q: Find a shortest path from 1 to 3



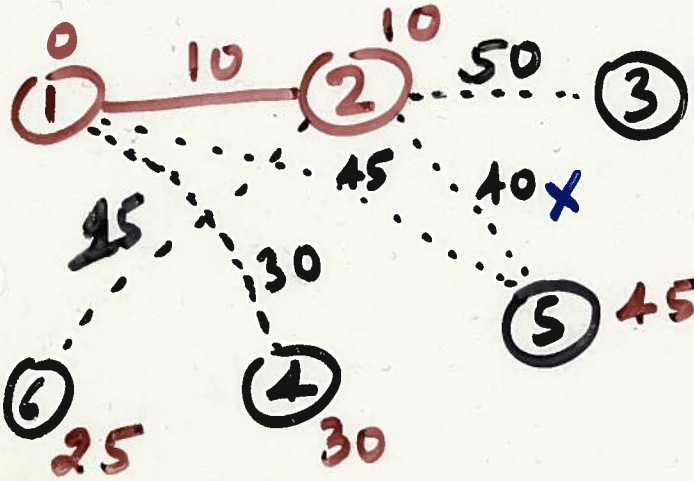


$10 = \min(\infty, 10)$

$45 = \min(\infty, 45)$

$30 = \min(\infty, 30)$

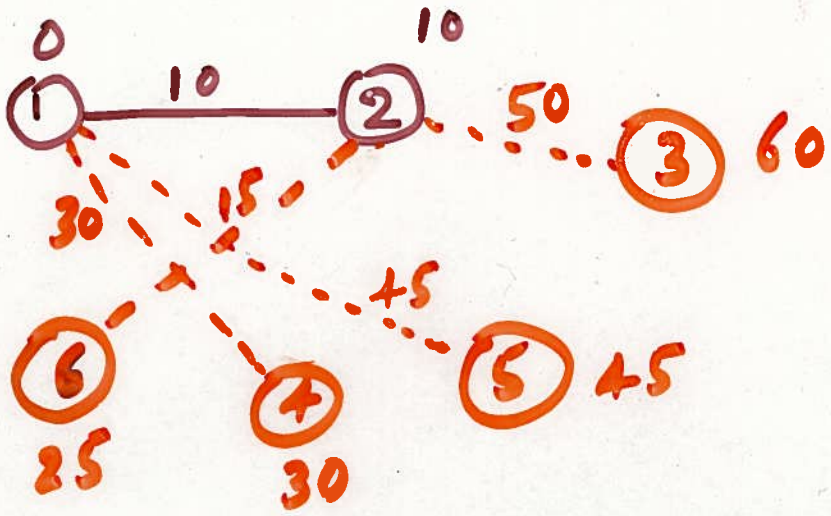
choose 2



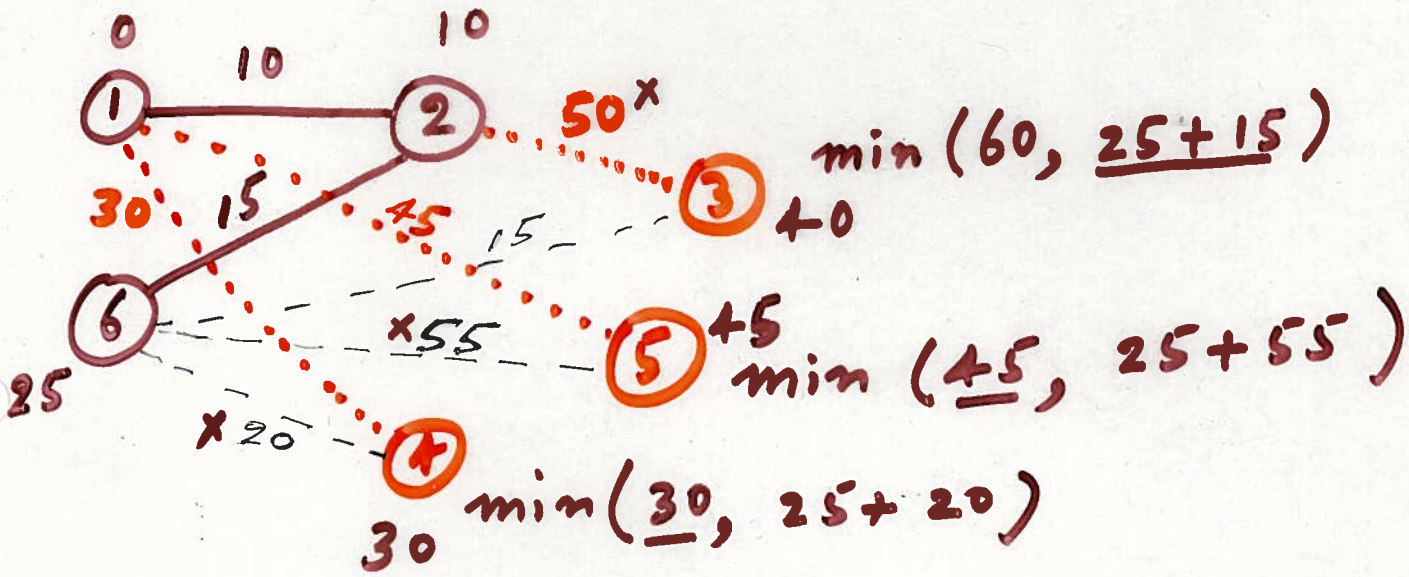
$60 = \min(\infty, \underline{10+50})$

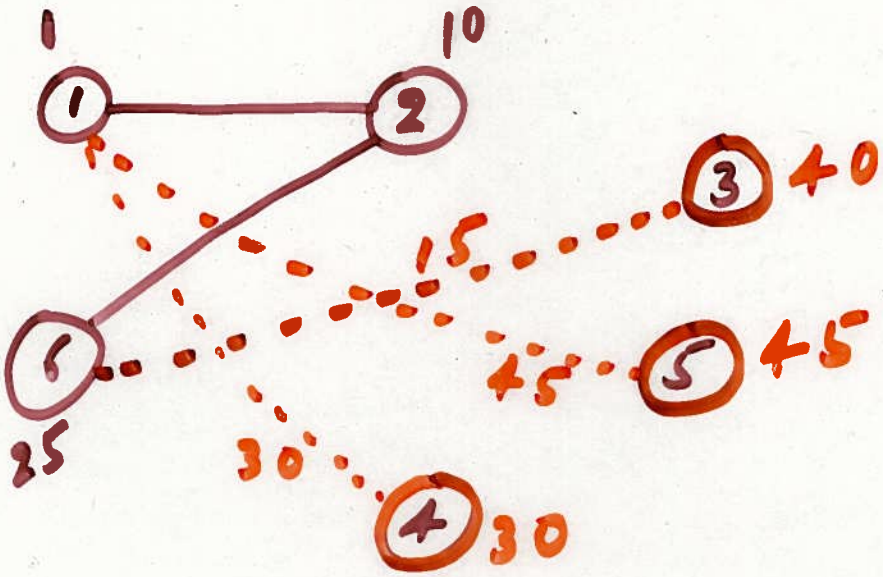
$45 = \min(\underline{45}, 10+40)$

choose 6

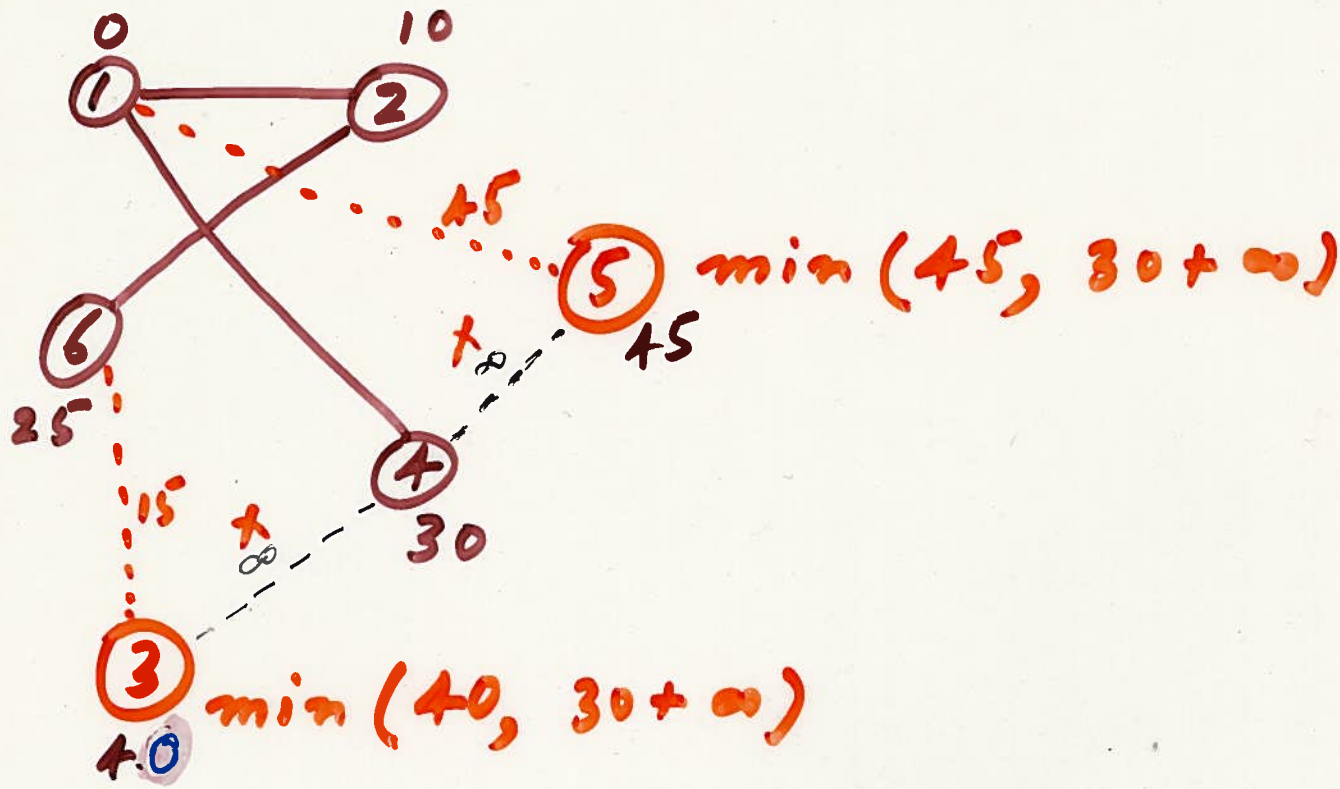


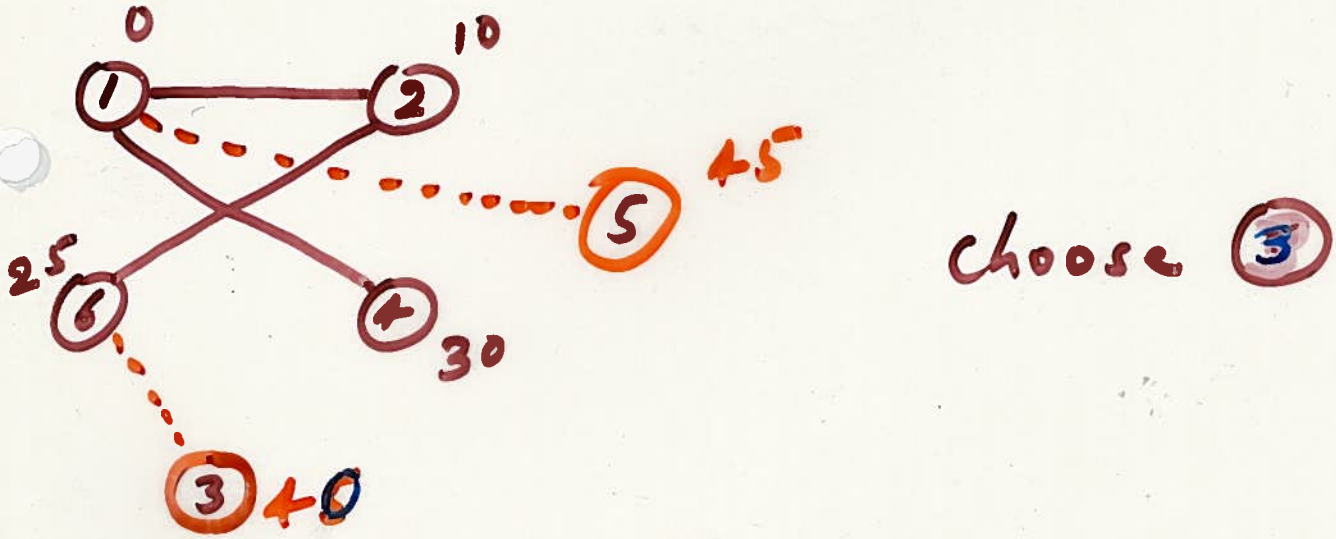
Choose 6



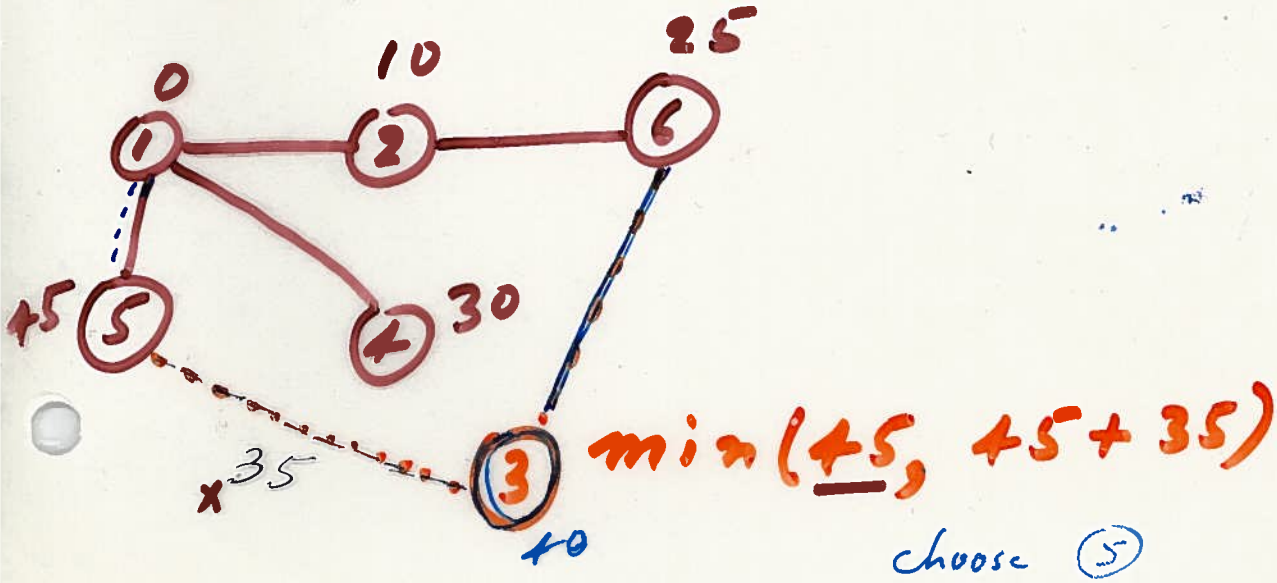


Choose 4

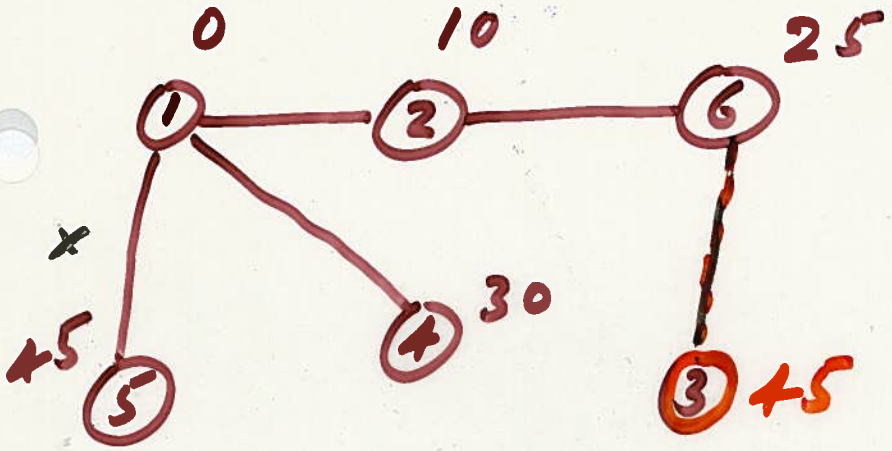




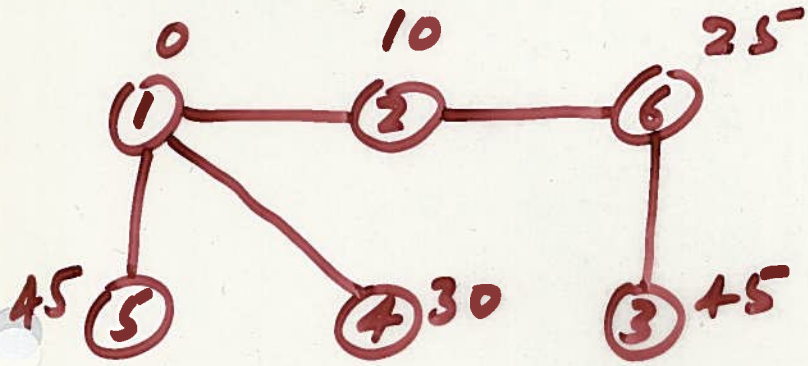
choose (3)



choose (5)



Choose 3



Shortest Path Tree with root node 1

Assumption: No negative weight.

```
DIJKSTRA(G)
    for each v ∈ V
        do d[v] ← ∞
    d[s] ← 0
    S ← ∅
    Q ← V
    while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
           S ← S ∪ {u}
           for each v ∈ Adj[u]
               do if d[v] > d[u] + w(u, v)
                     then d[v] ← d[u] + w(u, v)
```

Q : Priority Queue

* DECREASE KEY (v)

Q	n* Extract-Min + m* Decrease-key	
array	$n * O(n) + m * O(1)$	$= O(n^2)$
heap	$n * \log(n) + m * \log n$	$= O(m \log n)$
fibonacci	$n * \log(n) + m * O(1)$	$= O(n \log n + m)$

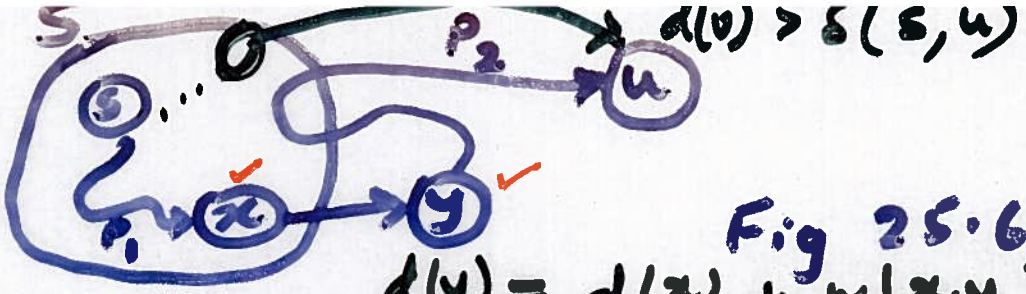


Fig 25.6

Correctness: Prove that whenever u is added to S, $d[u] = \delta(s,u)$
⟨⟨Theorem 25.10⟩⟩

Proof: ⟨⟨Basically same as in book, but derives a different contradiction.⟩⟩

- Figure 25.6
- Note that $\forall v\ d[v] \geq \delta(s,v)$
  (because lemma proved for Bellman-Ford above was just about relaxation, didn't depend on order of relaxing edges)
- Let u be first vertex picked such that $\exists$ shorter path than $d[u]$
  $\Rightarrow d[u] > \delta(s,u)$ — (i)
- Let y be first vertex $\in V - S$ on actual shortest path from s to u
  $\Rightarrow d[y] = \delta(s,y)$ — (ii)
  Because:
  - $d[x]$ is set correctly for y's predecessor $x \in S$ on the shortest path (by choice of u as first choice for which that's not true)
  - when put x into S, relaxed (x,y), giving $d[y]$ correct value,

if $y = u$, we are done.

$$\begin{aligned} d[u] &> \delta(s,u) && \text{— (i)} \\ &= \delta(s,y) + \delta(y,u) && \text{(optimal substructure)} \\ &= d[y] + \delta(y,u) && \text{(ii)} \\ &\geq d[y] && \text{(no negative weights)} \end{aligned}$$

- But $d[u] > d[y]$ $\Rightarrow$ algorithm would have chosen y to process next, not u.
  Contradiction.

Bellman. Ford Algorithm

1. **for** each $v \in V$
   do $d[v] \leftarrow \infty$
   $d[s] \leftarrow 0$

$O(nm)$

2. **for** $i \leftarrow 1$ **to** $|V| - 1$
   do for each edge $(u, v) \in E$
   do if $d[v] > d[u] + w(u, v)$
   then $d[v] \leftarrow d[u] + w(u, v)$

3. **for** each edge $(u, v) \in E$
   do if $d[v] > d[u] + w(u, v)$
   then no solution

Relaxation

Lemma: $d[v] \geq \delta(s, v)$ always.

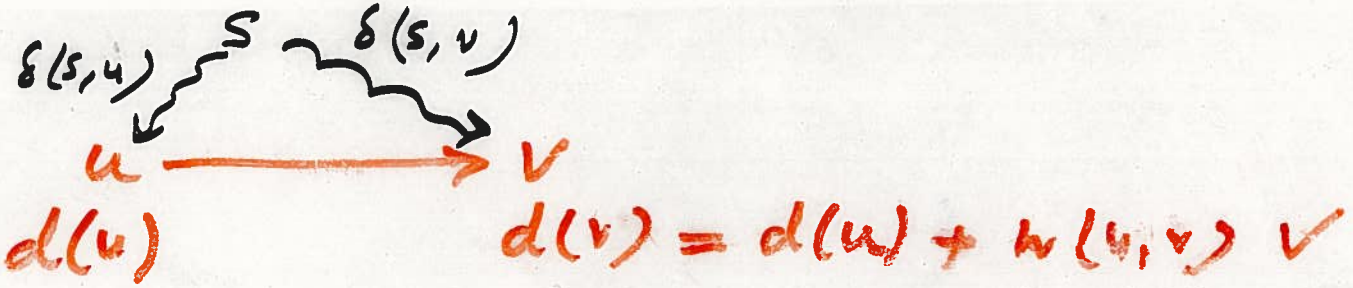
⟨⟨First part of Lemma 25.5, Section 25.1⟩⟩

- Initially true
- Let v be first vertex for which $d[v] < \delta(s, v)$, and let u be vertex that caused $d[v]$ to change:
- $d[v] = d[u] + w(u, v)$ — (i)
- Then

$$\begin{aligned} d[v] &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) && \text{(Triangle inequality)} \\ &\leq d[u] + w(u, v) && (v \text{ is first violation}) \end{aligned}$$

$\Rightarrow d[v] < d(u) + w(u,v)$ — (ii)

contradicts $d[v] = d[u] + w(u, v)$ (above).



- Bellman-Ford is correct (after $|V|-1$ passes, all the d values are correct) ⟨⟨Lemma 25.12⟩⟩

  Proof: ⟨⟨Informal version of book's proof.⟩⟩

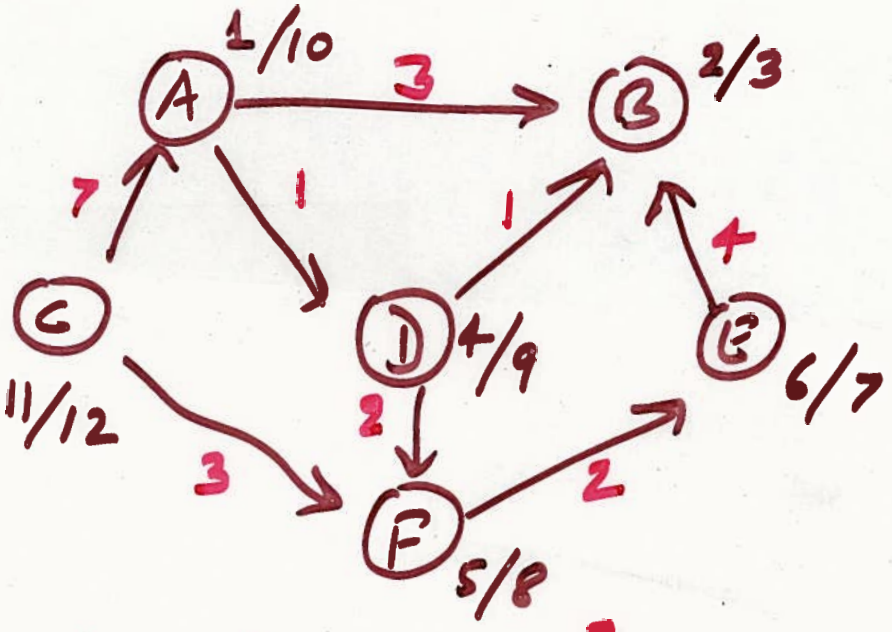
  Let v be a vertex, and consider shortest path from s to v (assuming no neg-weight cycles):

  $s \to v_1 \to v_2 \to v_3 \to v_4 \to v$

  - Initially, $d[s] = 0$ is correct (and doesn't change thereafter — by previous lemma and fact that code never increases d)
  - After 1 pass through edges, $d[v_1]$ is correct (and doesn't change...) ($d[s]$ is correct, and by optimal substructure, shortest distance is $w(s, v_1)$. 1st pass sets $d[v_1] = d[s] + w(s, v_1)$, which is right answer.)
  - After 2 passes through edges, $d[v_2]$ is correct (and doesn't change...)

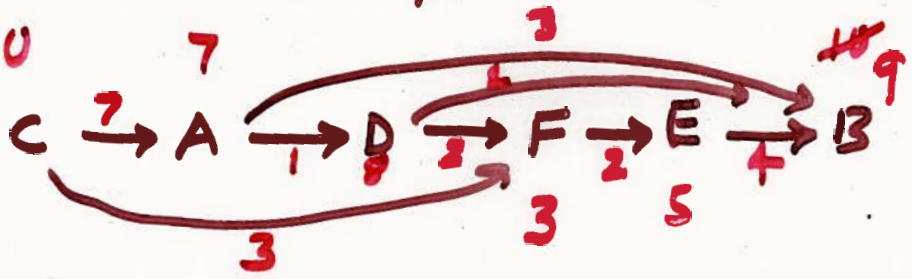
    ⋮

TOPOLOGICAL SORTING



DAG & DFS

(u)
visiting time
finishing time



SHORTEST PATH IN DAGs

1. TOPOLOGICAL SORT USING DFS
2. ONE ITERATION OF BELLMAN-FORD ALGORITHM

$O(n+m)$

1 2 3 4
1 3 2 4