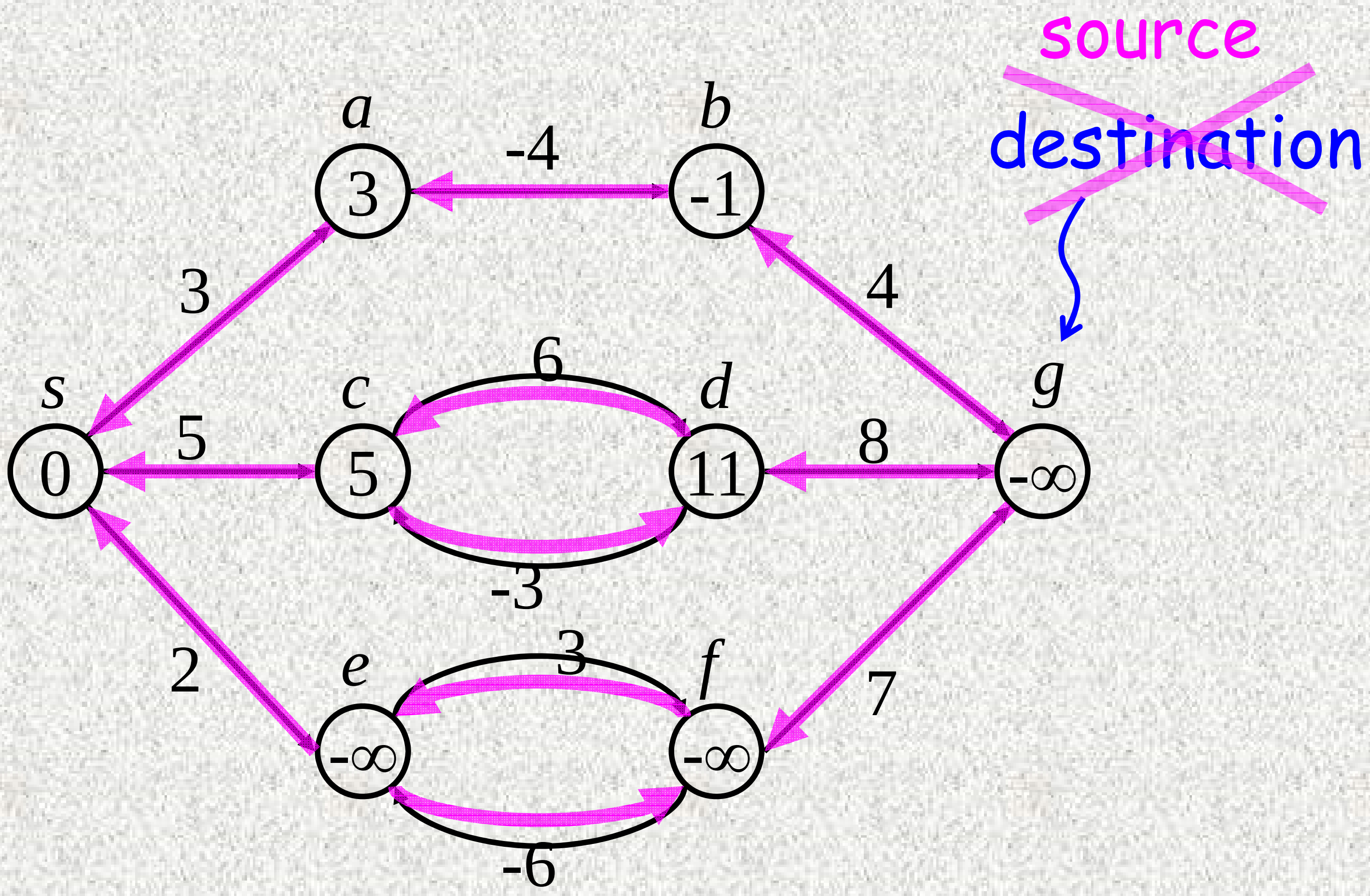
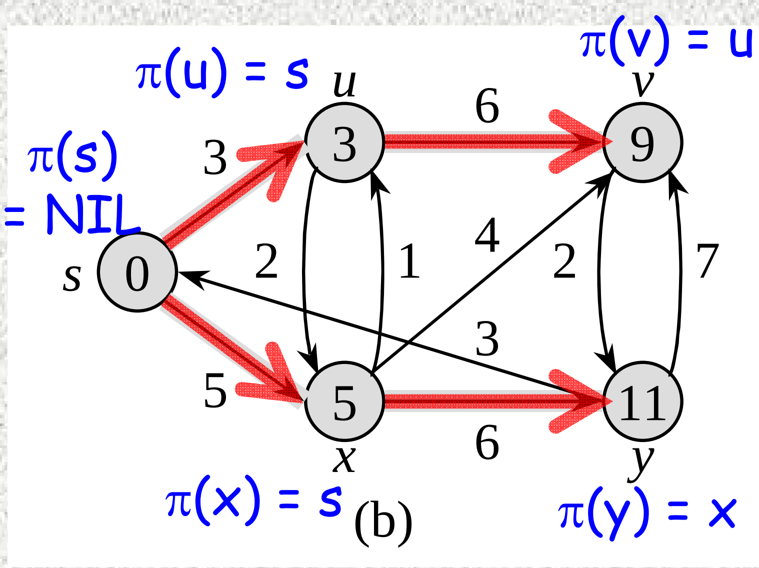
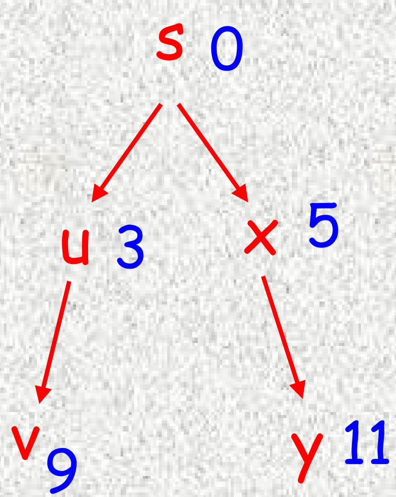
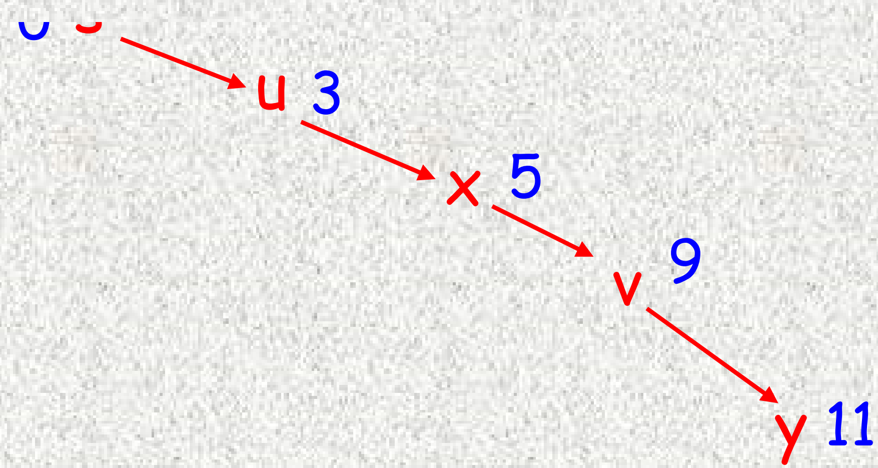

source
~~destination~~
a
b
3
-4
-1
3
4
6
s
c
d
g
5
8
0
5
11
-∞
-3
2
3
7
e
f
-∞
-∞
-6

Shortest-paths tree



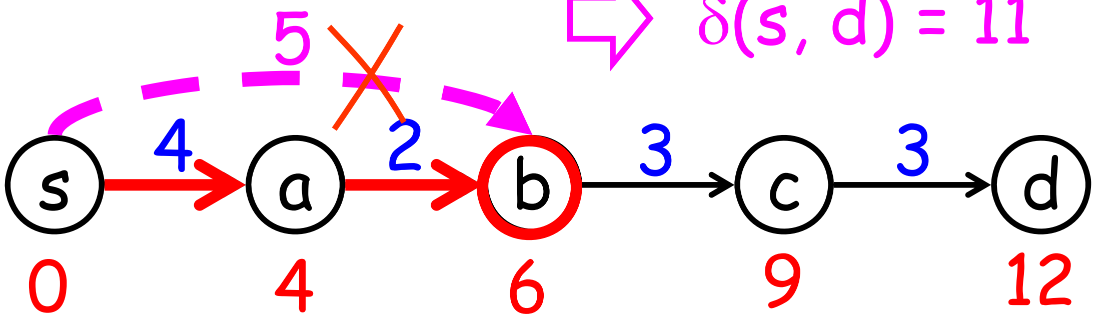




Main Idea ----- 1

$\Rightarrow$ $\delta(s, d) = 11$

If



$\pi(d) = c$

is a shortest path from s to d

Then

(i) all subpaths are shortest **optimal substructure !**

(ii) After $\delta(s, \pi(v))$ is known,
we can get $\delta(s, v)$ by Relax($\pi(v)$, v, w)

e.g. After $\delta(s, c) = 9$ is known,
we have $\delta(s, d) = 9 + w(c, d) = 12$

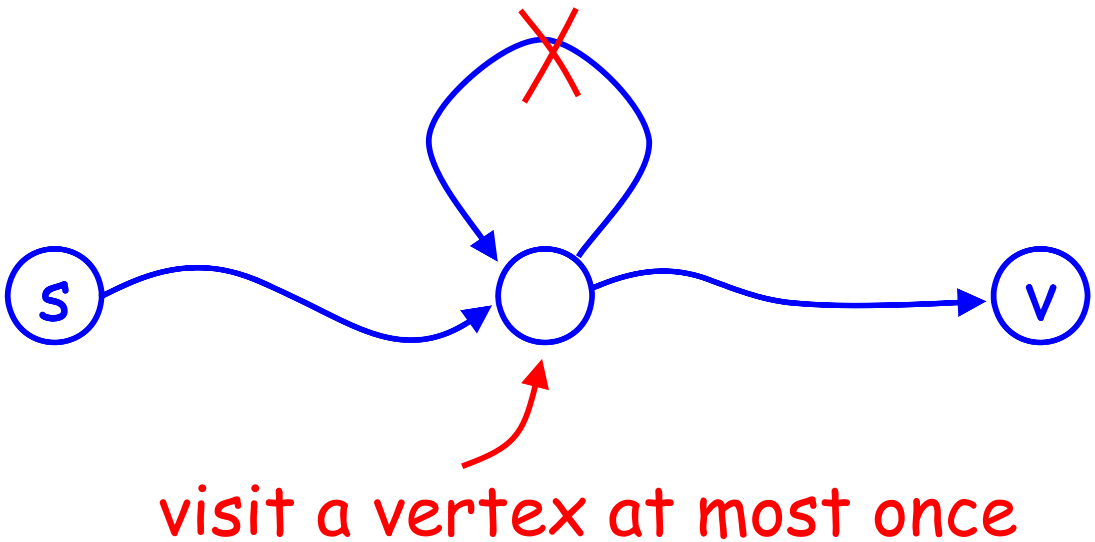
Relax(c, d, w)

Main Idea ----- 2

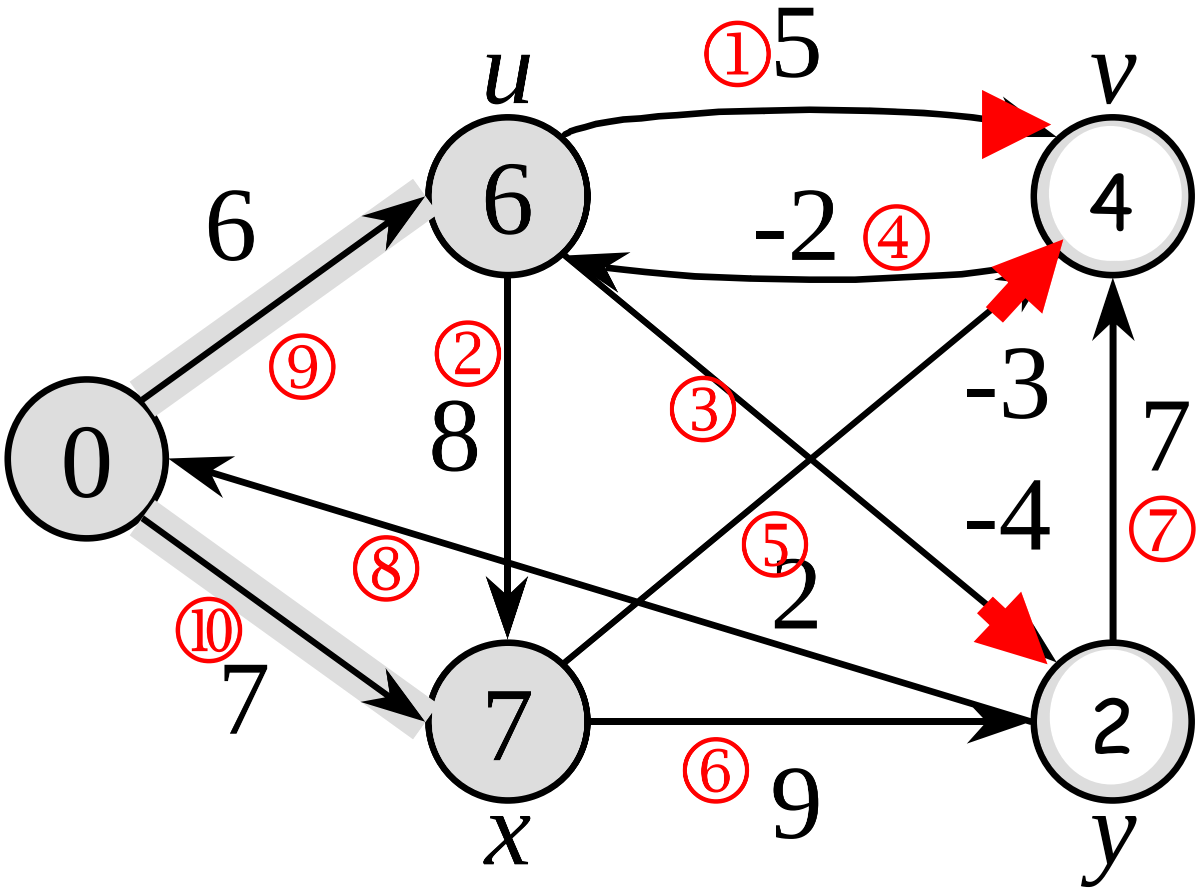
If G contains no negative cycles,

(i) every shortest path is a simple path

(ii) every shortest path has at most $n - 1$ edges



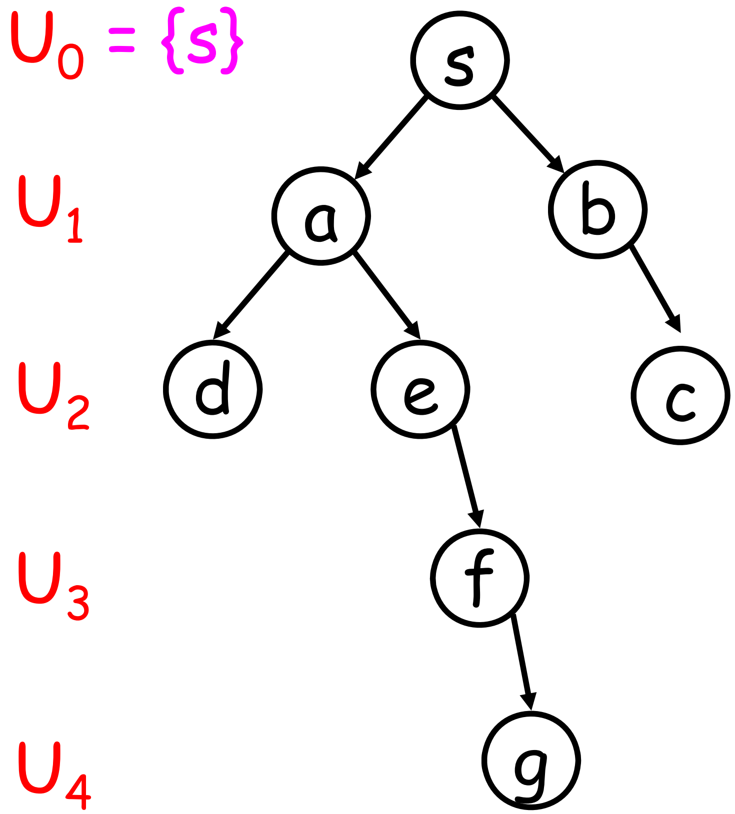
(For ease of discussion, assume that there are no 0-cycles)



(b)

Main Idea: Bellman-Ford
(no negative cycles)

shortest path tree



* U_i: vertices whose shortest paths having i edges

* U_0 ok $\xrightarrow{\text{phase 1}}$ U_1 ok $\xrightarrow{\text{phase 2}}$ U_2 ok $\longrightarrow$ • • •

main idea 1 - correctness

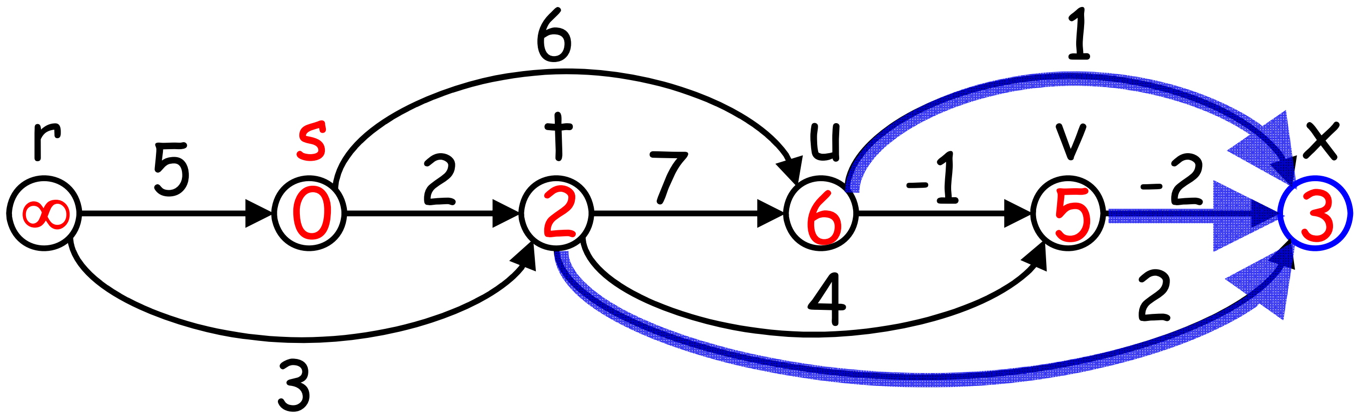
* A simple path has at most n – 1 edges

⇨ $U_n = U_{n+1} = U_{n+2} = \ldots = \varnothing$

⇨ n – 1 phases is sufficient!

main idea 2 - time complexity

Traditional approach: DP (See 15-14a)



$$\begin{cases} d(s) = 0 \\ d(v) = \underset{(u,v) \in E}{\text{MIN}} \{d(u) + w(u, v)\} \end{cases}$$

bottom-up computation
(left-to-right)

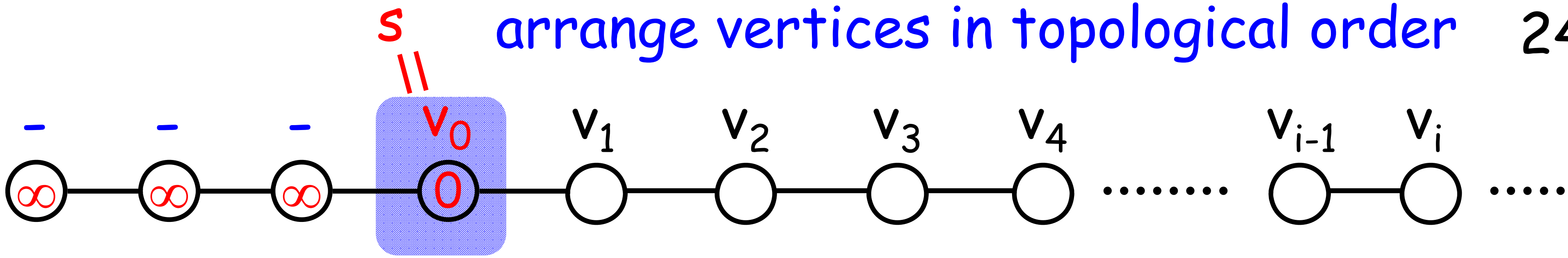
$\pi(x) \in \{t, u, v\}$

$$d(x) = \begin{cases} d(t) + 2, \\ d(u) + 1, \\ d(v) + (-2) \end{cases}$$

DP: 有 答 案 的 存 起 來 等 別 人 問 (t, u, v 等 x 來 問 答 案)

24.2: 有 答 案 的 主 動 去 修 正 有 需 要 的 人 (t, u, v 主 動 用 答 案 修 正 x)

arrange vertices in topological order



* all edges are from left to right →

* $\pi(v_i)$ is one of $v_0, v_1, v_2, \ldots, v_{i-1}$ (or NIL)

* Once $v_0, v_1, v_2, \ldots, v_{i-1}$ ok $\Rightarrow$ v_i ok!

* Initially, $d(v_0)$ is correct

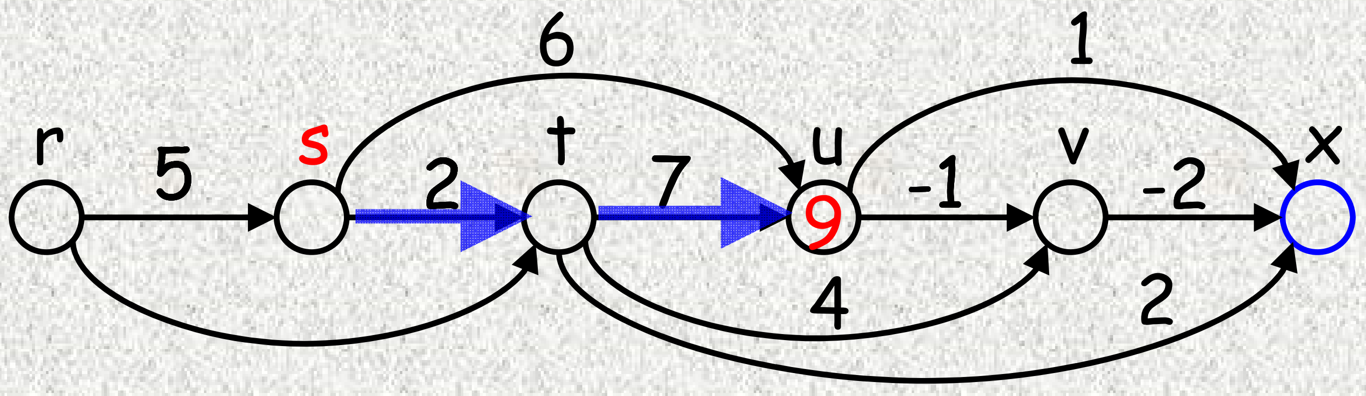
v_0 does "relax" with correct $d(v_0)$ $\Rightarrow$ $d(v_1)$ is correct

$\Rightarrow$ v_1 does "relax" with correct $d(v_1)$ $\Rightarrow$ $d(v_2)$ is correct

$\Rightarrow$ v_2 does "relax" with correct $d(v_2)$ $\Rightarrow$ $d(v_3)$ is correct

$\Rightarrow$ • • • all $d(v_i)$ are correct (by induction)

The longest path problem on a DAG



Negating the edge weights

* edge weights: 5, -2, 7, -1, ... ⇨ -5, +2, - 7, +1, ...

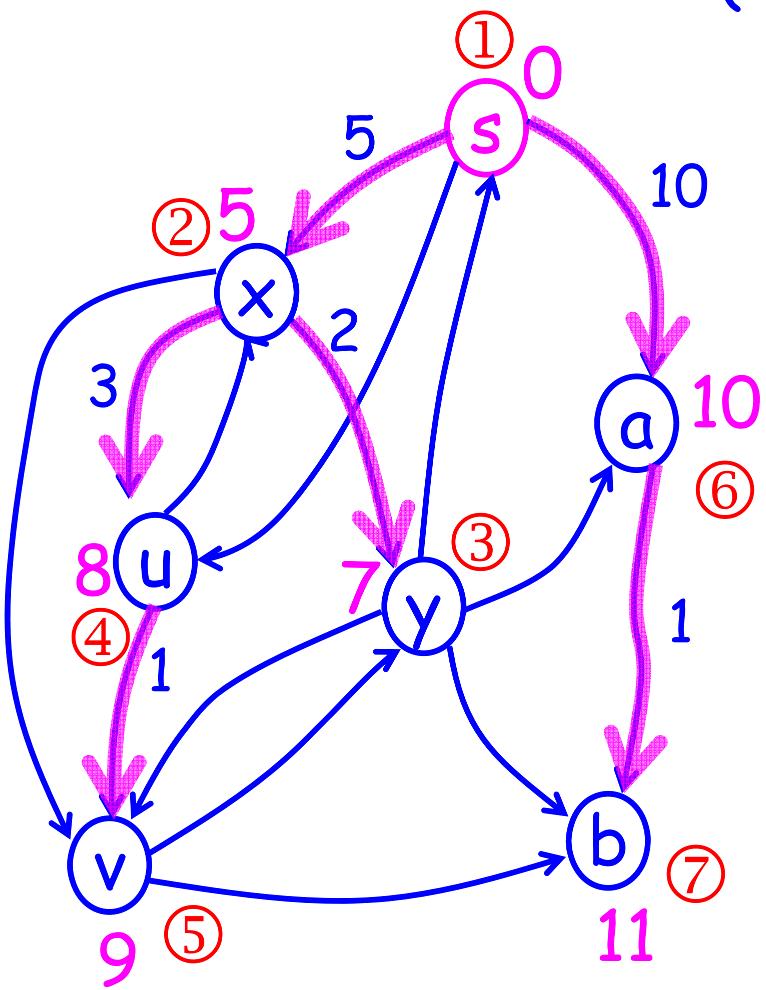
* path lengths: -3, 12, 73, 24, ... ⇨ +3, -12, -73, -24, ...

longest

shortest

Main Idea: Dijkstra

(no negative edge)



No negative edge → $\delta(v) > \delta(\pi(v))$

⇨ rank(v) > rank(π(v))

⇨ Once 1 2 3 • • • k ok,

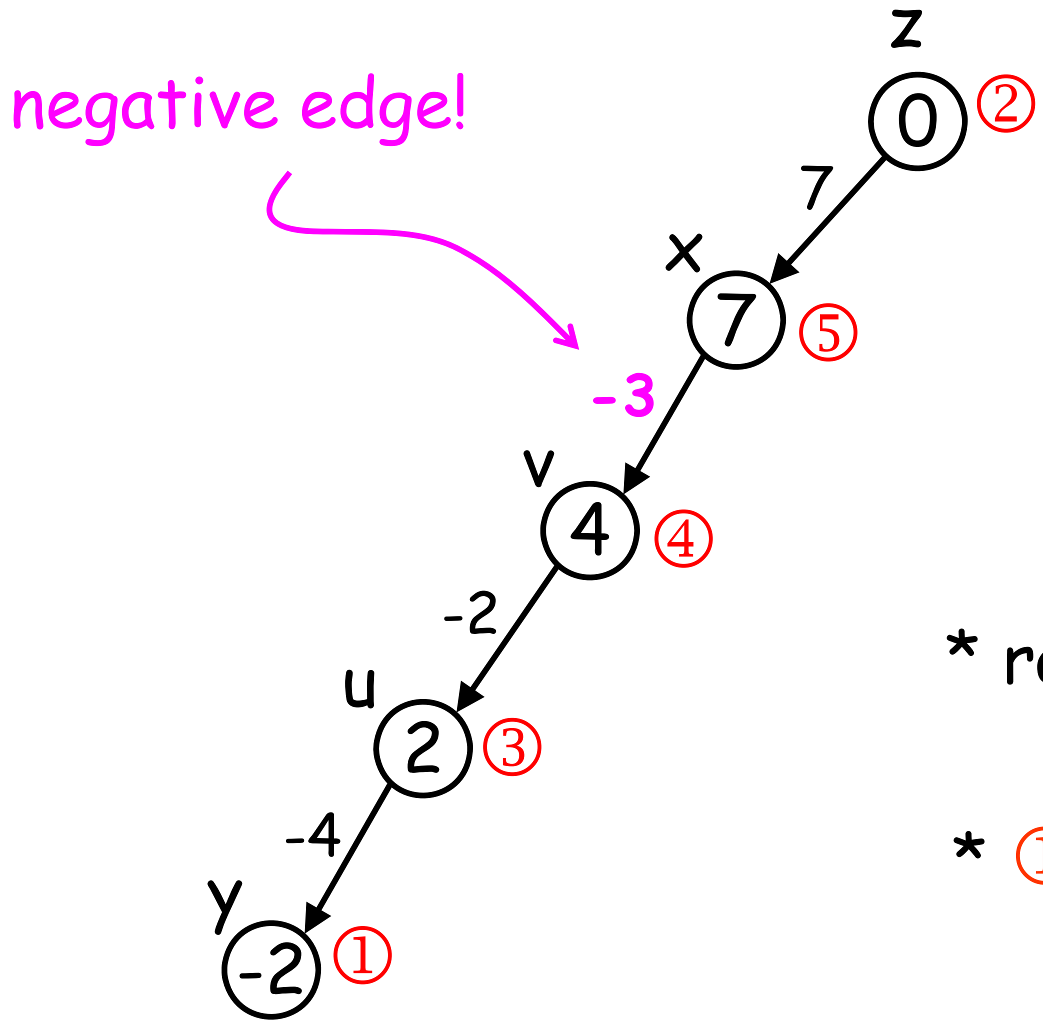
k+1 can be computed.

⇨ 1 → 2 → 3 → 4 → • • •

ok ok ok ok

必 然 是 s

Why all weights should be nonnegative?



(shortest path tree of 24-5 Fig.)

Dijkstra's idea :

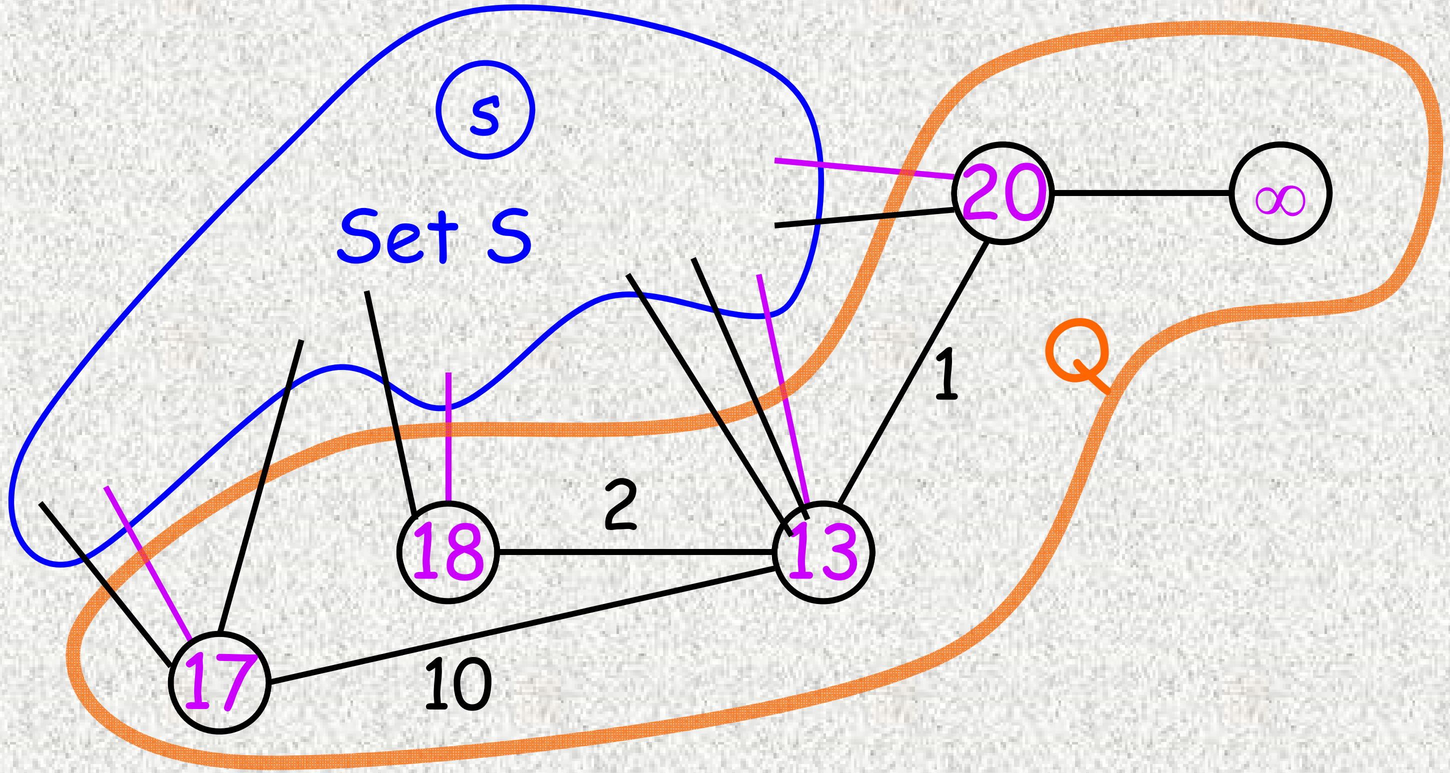
$\cancel{\text{rank}(v) > \text{rank}(\pi(v))}$

* $\text{rank}(v) < \text{rank}(\pi(v))$
  ④ ⑤

* ① ② ③ ok $\Rightarrow$ v not ok
  ④

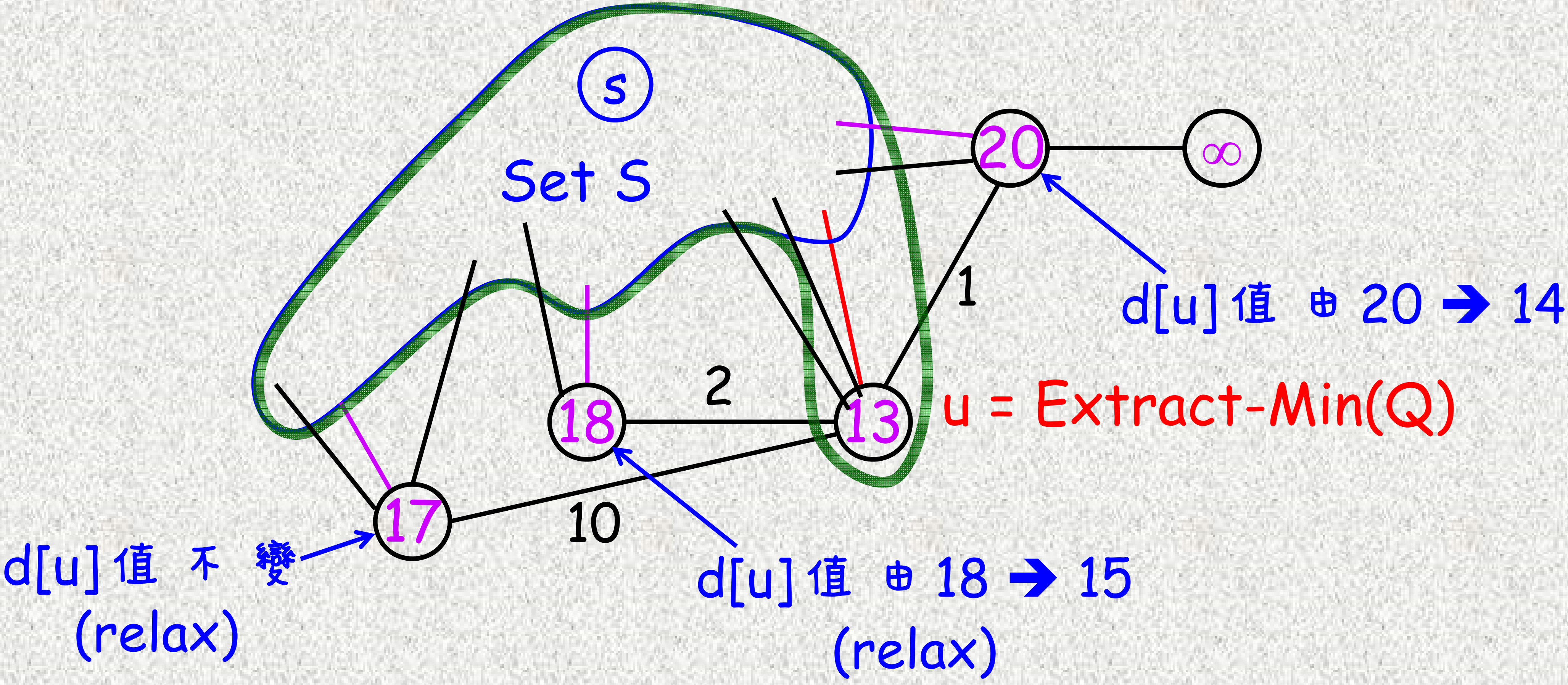
Dijkstra's shortest path algorithm

* d[u] 記住 u 和 s 之間目前已知的最短距離 (π[u] 記住目前的 predecessor)



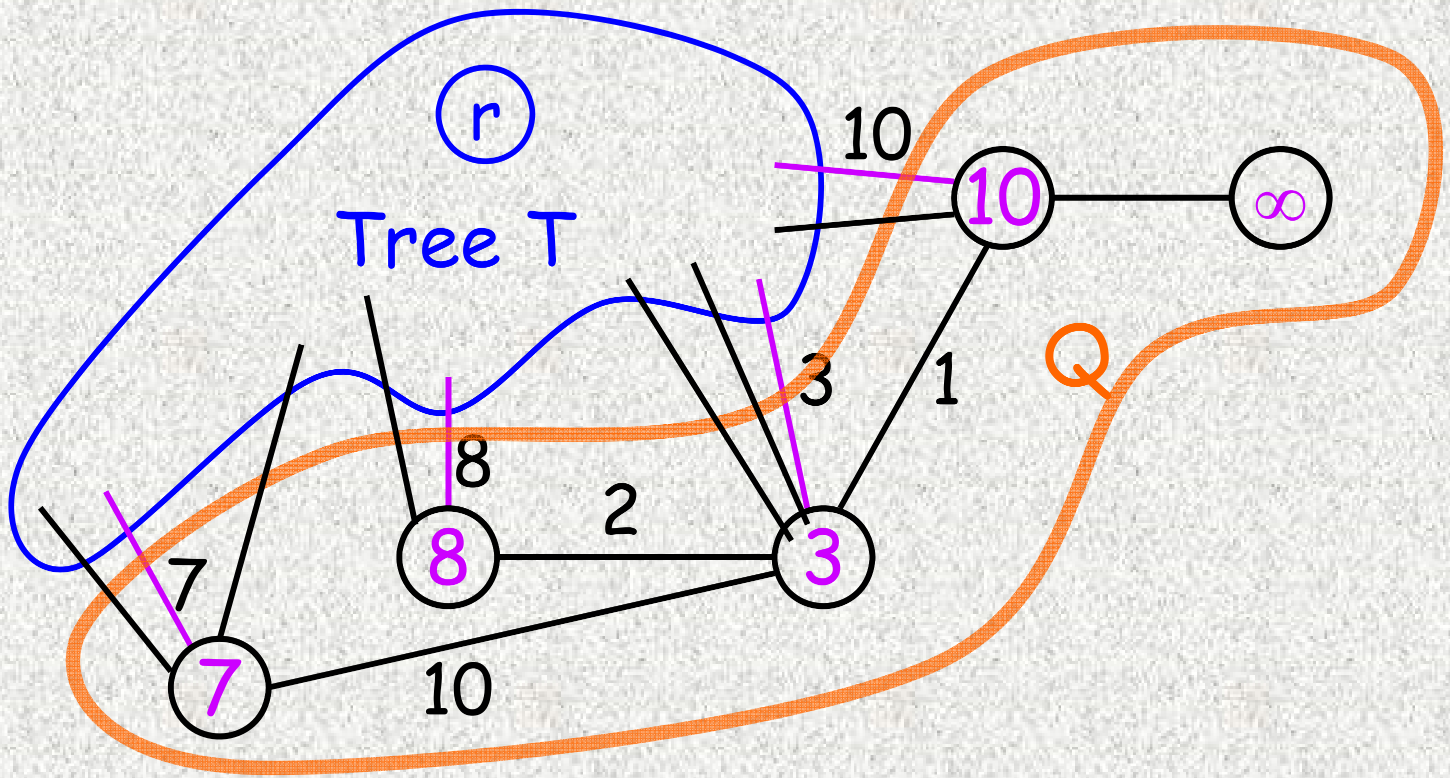
Dijkstra's shortest path algorithm

* d[u] 記住 u 和 s 之間目前已知的最短距離
(π[u] 記住目前的 predecessor)



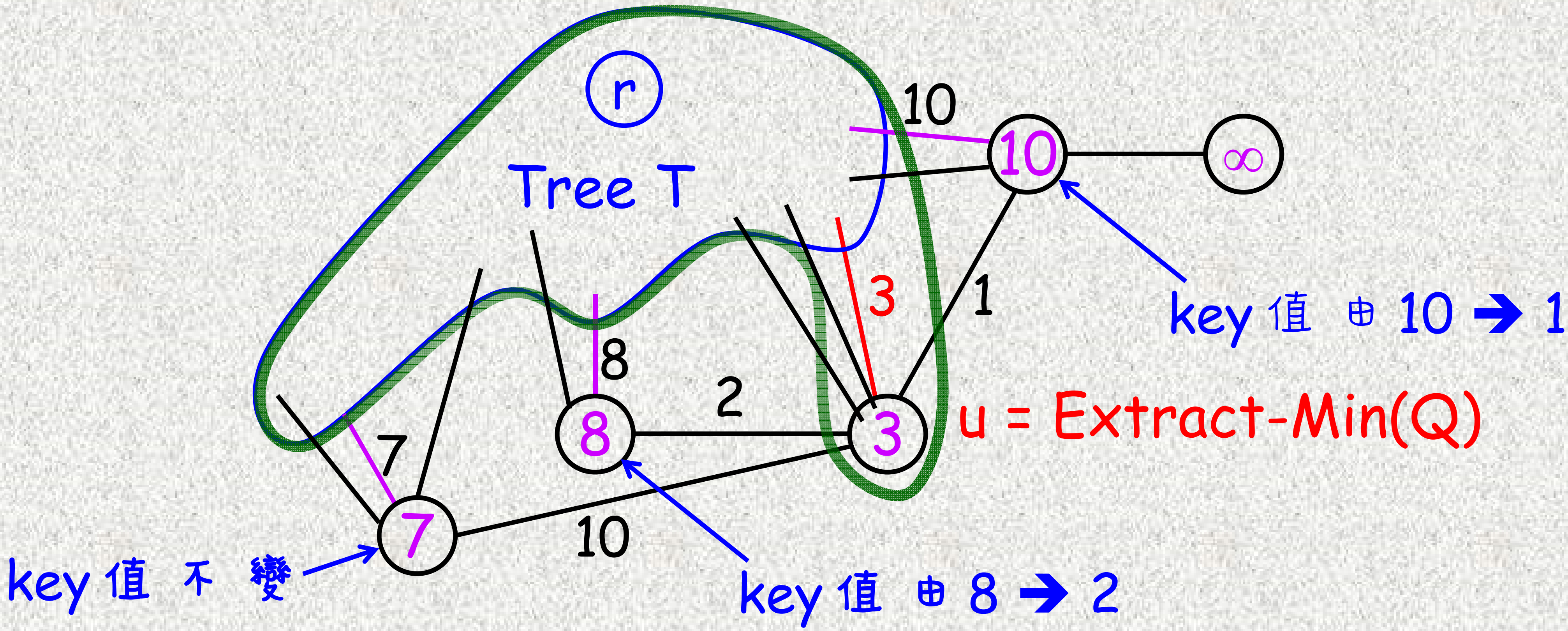
Prim's MST

* key[u] 記住 u 和 T 之間最短的一條 edge

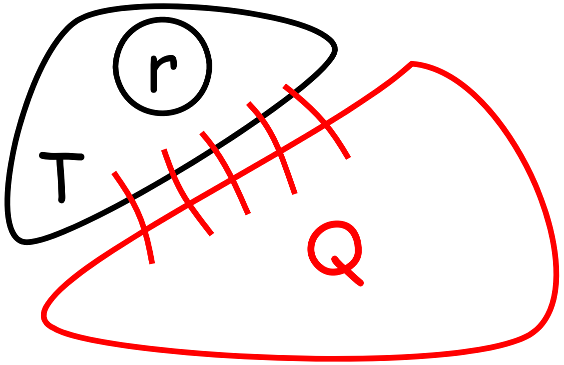


Prim's MST

* key[u] 記住u和T之間最短的一條edge



Prim's MST



key[v] : shortest edge to T

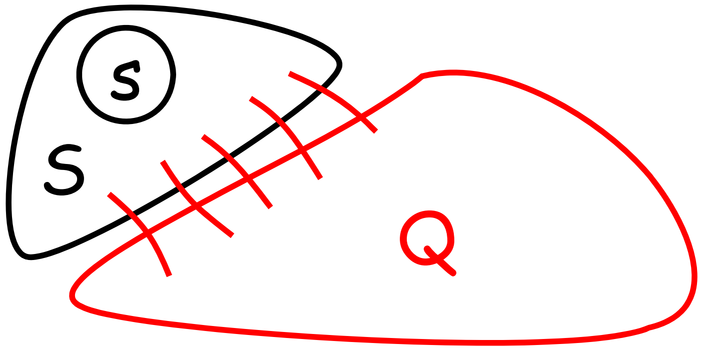
π[v] : nearest vertex in T

$u \leftarrow$ ExtractMin(Q)

$T \leftarrow T \cup \{u\}$

reduce key[$\cdot$] of Adj(u)
(decrease-key)

Dijkstra's shortest path



d[v] : known shortest distance to s

π[v] : current predecessor

$u \leftarrow$ ExtractMin(Q)

$S \leftarrow S \cup \{u\}$

relax d[$\cdot$] of Adj(u)
(decrease-key)

		array	b. heap	f. heap
Steps 1~3:	Build Q	O(V)	O(V)	O(V)
Step 5:	V times Extract-Min	$O(V^2)$	O(V lg V)	O(V lg V)
Steps 7~9:	E times Decrease-Key	O(E)	O(E lg V)	O(E)
		$O(V^2+E)$	O(E lg V)	O(E + Vlg V)

Procedure	Binary heap (worst-case)	Fibonacci heap (amortized)	array
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$	O(1)
INSERT	$\Theta(\lg n)$	$\Theta(1)$	O(1)
MINIMUM	$\Theta(1)$	$\Theta(1)$	O(n)
EXTRACT-MIN	$\Theta(\lg n)$	$O(\lg n)$	O(n)
UNION	$\Theta(n)$	$\Theta(1)$	O(n)
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(1)$	O(1)
DELETE	$\Theta(\lg n)$	$O(\lg n)$	O(1)
build	O(n)	O(n)	O(n)

(See 22-1)

Single-Source Shortest Paths Algorithms - Review

Main Ideas

Optimal substructure: $\underset{\text{ok}}{\pi(v)} \xrightarrow[\text{relax}]{} \underset{\text{ok}}{v}$

No negative cycles: simple path (at most n-1 edges)

Bellman-Ford (no negative cycles, can detect) O(VE)

$\underset{\text{ok}}{U_0} \overset{=\{s\}}{} \rightarrow \underset{\text{ok}}{U_1} \rightarrow \underset{\text{ok}}{U_2} \rightarrow \underset{\text{ok}}{U_3} \rightarrow \dots \rightarrow \underset{\text{ok}}{U_{n-1}}$

Dijkstra (no negative edges) O(Vlg V+E)

$\underset{\text{ok}}{\text{rank}(1)} \overset{=\{s\}}{} \rightarrow \underset{\text{ok}}{\text{rank}(2)} \rightarrow \underset{\text{ok}}{\text{rank}(3)} \rightarrow \dots \rightarrow \underset{\text{ok}}{\text{rank}(n)}$

Two important special cases

Single-Source on un-weighted graph $O(V+E)$

BFS

Single-Source on a DAG: shortest/longest $O(V+E)$

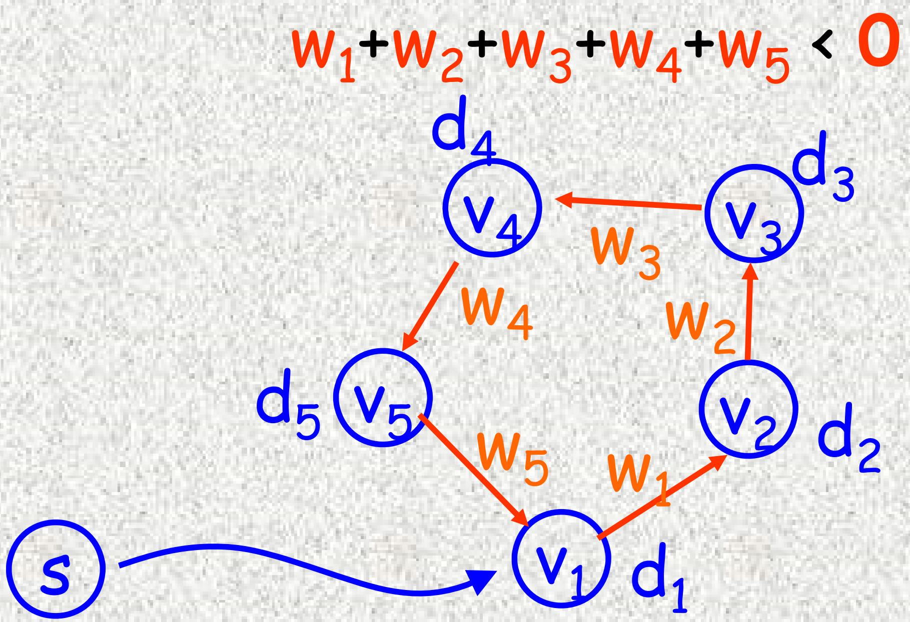
(1) Bellman-Ford: one phase – left to right

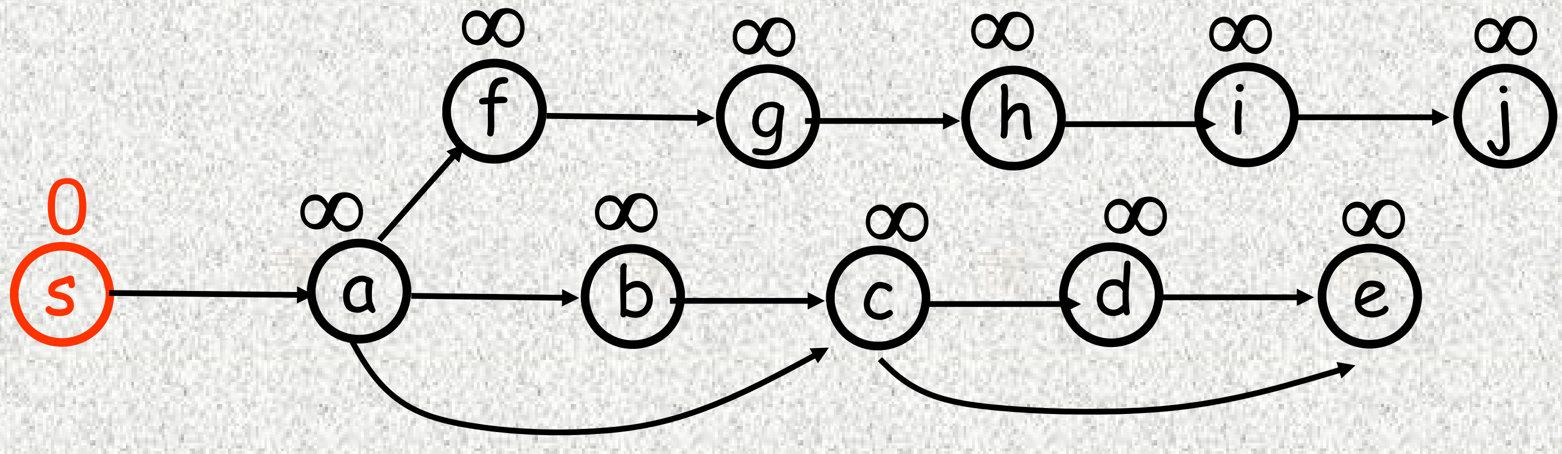
(2) classical: DP

Bellman-Ford with negative cycles

Assume that G contains a negative cycle C reachable from s

After n-1 iterations, every $d_i \neq \infty$





$d(a) \neq \infty$ after ? iteration
$d(b) \neq \infty$ after ? iteration
$d(d) \neq \infty$ after ? iteration
$d(j) \neq \infty$ after ? iteration

⇨ after n-1 iterations,
any vertex v reachable from s has $d(v) \neq \infty$

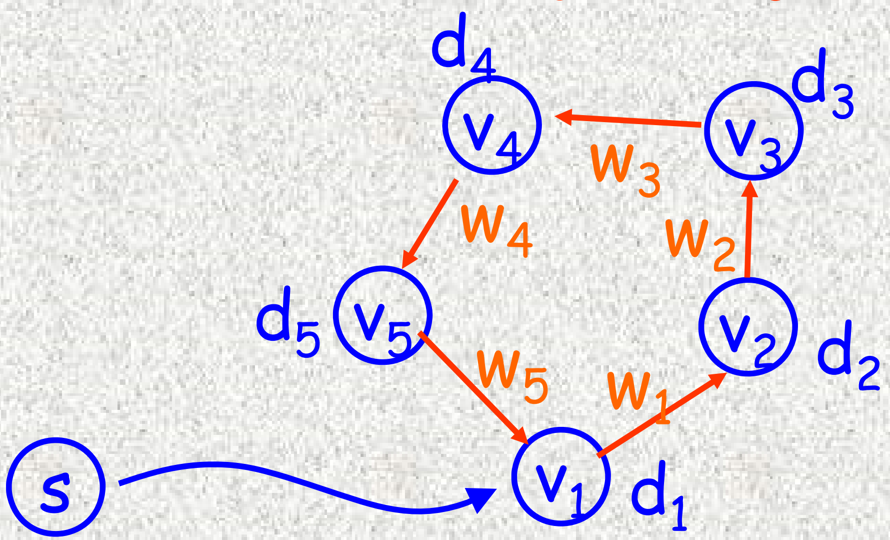
Bellman-Ford with negative cycles

Assume that G contains a negative cycle C reachable from s

After n-1 iterations, every $d_i \neq \infty$

At iteration n, **at least one** v_i accepts "Relax"

$w_1+w_2+w_3+w_4+w_5 < 0$



By contradiction:

$$\begin{array}{rrcl}
 & \cancel{d_1} + w_1 & \geq & \cancel{d_2} \\
 & \cancel{d_2} + w_2 & \geq & \cancel{d_3} \\
 & \cancel{d_3} + w_3 & \geq & \cancel{d_4} \\
 & \cancel{d_4} + w_4 & \geq & \cancel{d_5} \\
+ & \cancel{d_5} + w_5 & \geq & \cancel{d_1} \\
\hline
 & w_1+w_2+w_3+w_4+w_5 & \geq & 0
\end{array}$$

$\Rightarrow$ C is not negative !!!