Algorithms
演算法

Single Source Shortest Path
— 7.3 Dijkstra's Algorithm —

Professor Chien-Mo James Li 李建模
Electrical Engineering Department
National Taiwan University

Outline

- **Single Source Shortest Paths, CH24**
  - Bellman-Ford Algorithm [1956~1958]
  - **Dijkstra's Algorithm [1959]**
    - **Algorithm**
    - **Proof**
  - Directed Acyclic Graph [1972]
  - Linear Programming
- All-pairs Shortest Paths, CH25

	complexity	notice
B-Ford	$O(VE)$	Negative weight ok
Dijkstra	$O(E \lg V)$	No negative weight
DAG	$\Theta(V+E)$	DAG only

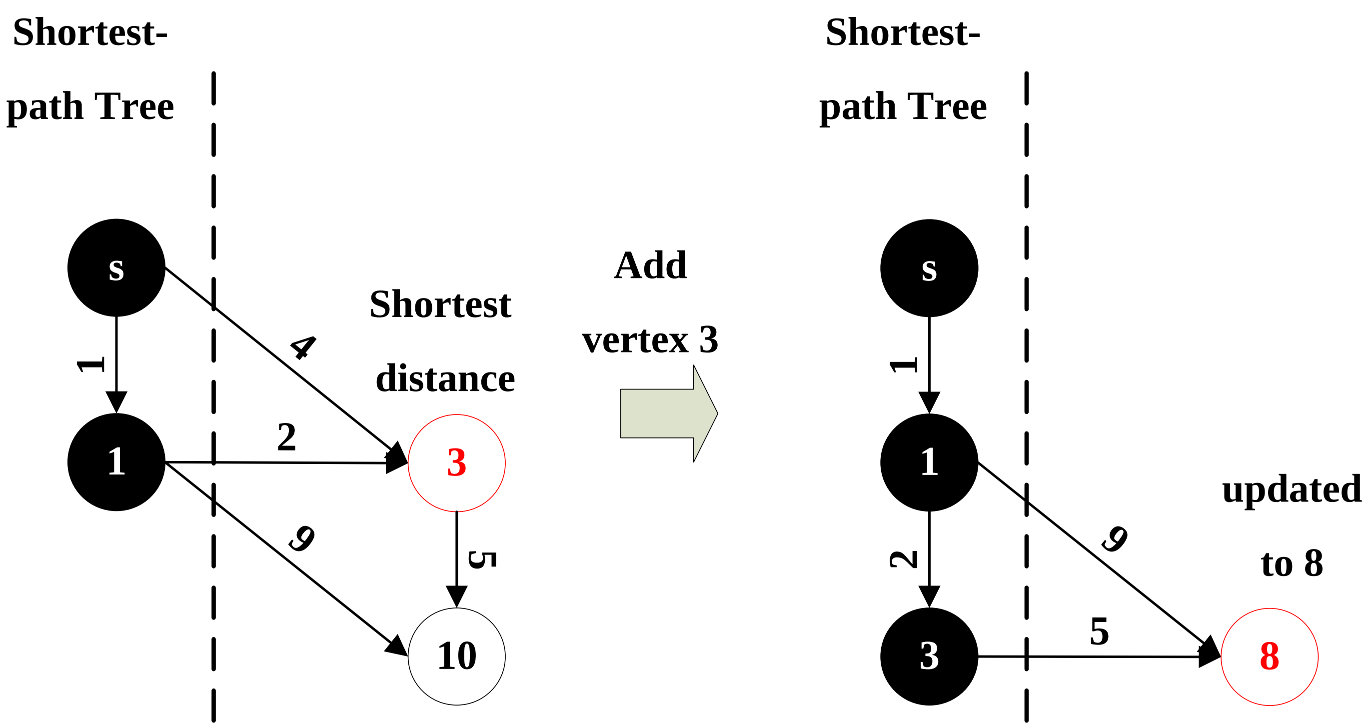
Edsger W. Dijkstra (1930~2002)

- Dutch computer scientist and mathematical scientist.
- Professor of mathematics at Eindhoven University of Technology
- 1972 ACM Turing Award Recipient

"The question of whether computers can think is like the question of whether submarines can swim."

Dijkstra's Idea

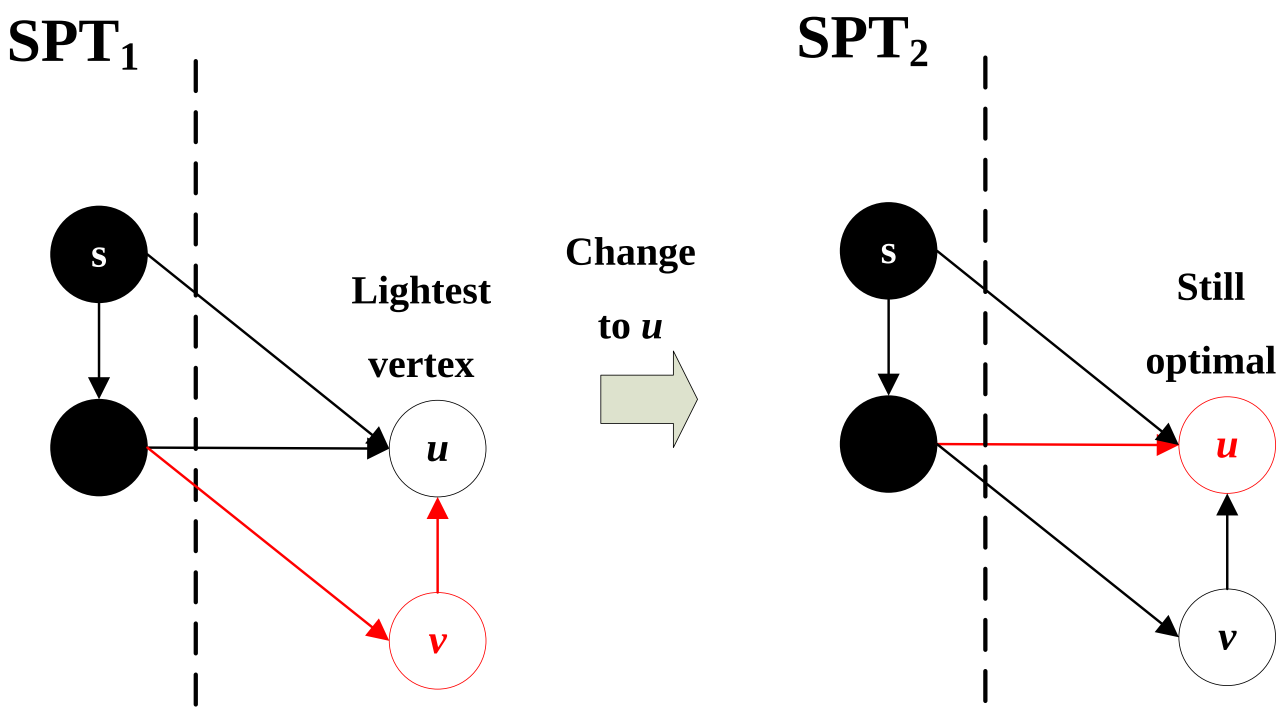
- Idea: Every time choose one lightest vertex into shortest-path tree
  - similar to Prim's MST
- Why greedy-choice is ok? because all weights are non-negative



Dijkstra Is Greedy Algorithm

Greedy Choice Property

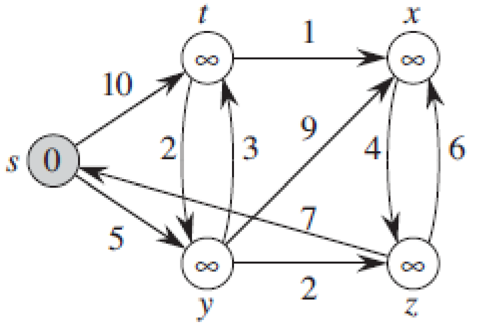
- Proof: cut and paste
- Suppose SPT_1 is optimal solution
  - *u* is lightest vertex crossing the cut
  - but SPT_1 connects to *v* rather than *u*
- Replace *v* by *u* to obtain SPT_2, which is also an optimal solution



Dijkstra's Algorithm (1)

- Greedy algorithm: choose *lightest vertex* in V-S to add to set S
  - S = set of vertices already in shortest-path tree
  - V-S = set of vertices not yet determined

```
DIJKSTRA(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    S = ∅
    Q = G.V
    while Q ≠ ∅
        u = EXTRACT-MIN(Q)
        S = S ∪ {u}
        for each vertex v ∈ G.Adj[u]
            RELAX(u, v, w)
```

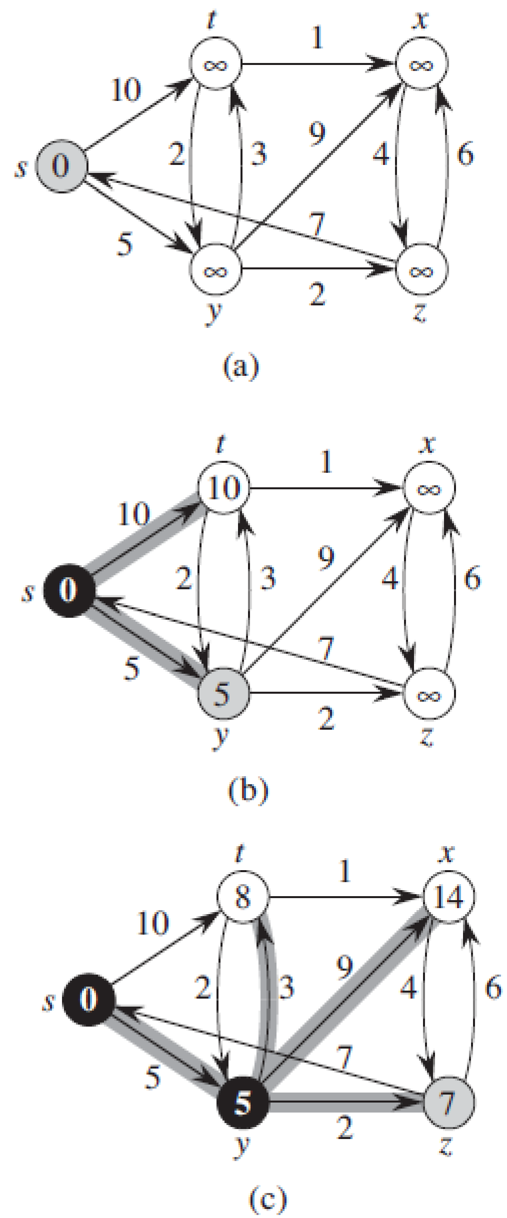


(a)

Fig 24.6

Dijkstra's Algorithm (2)

- **Black vertices** $\in S$
- **Gray vertices are lightest vertices** $\in V\text{-}S$



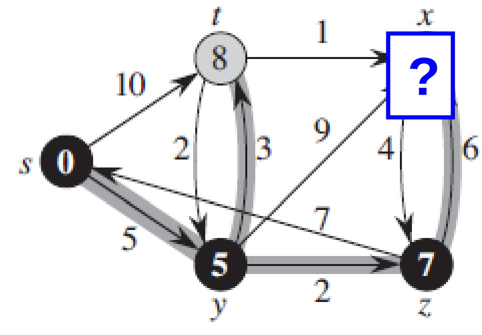
u=s

u=y

```
DIJKSTRA(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    S = ∅
    Q = G.V
    while Q ≠ ∅
        u = EXTRACT-MIN(Q)
        S = S ∪ {u}
        for each vertex v ∈ G.Adj[u]
            RELAX(u, v, w)
```

Dijkstra's Algorithm (3)

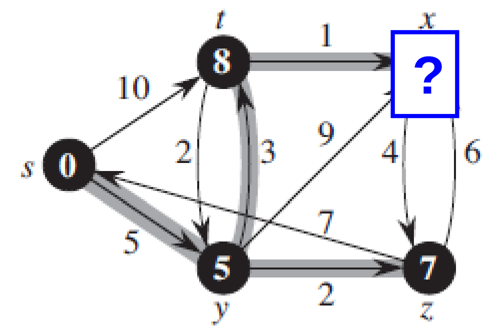
- **Black vertices** $\in S$
- **Gray vertices are lightest vertices** $\in V\text{-}S$



(d)

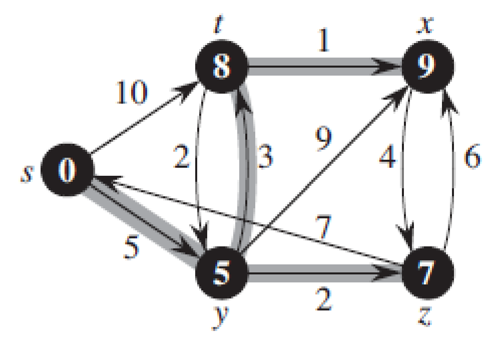
u=z

u=t



(e)

u=x



(f)

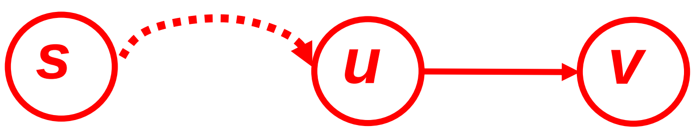
```
DIJKSTRA(G, w, s)
   INITIALIZE-SINGLE-SOURCE(G, s)
   S = ∅
   Q = G.V
   while Q ≠ ∅
       u = EXTRACT-MIN(Q)
       S = S ∪ {u}
       for each vertex v ∈ G.Adj[u]
           RELAX(u, v, w)
```

Convergence Property

(Lemma 24.14) **Let $s \rightsquigarrow u \rightarrow v$ be a shortest path.**
If $u.d = \delta(s, u)$ prior to RELAX(u, v, w), then $v.d = \delta(s, v)$ after RELAX(u,v,w)

- **Proof:**

$$
\begin{aligned}
v.d &\le u.d + w(u,v) && \text{(after RELAX)} \\
&= \delta(s,u) + w(u,v) \\
&= \delta(s,v) && \text{(by Lemma 24.1)}
\end{aligned}
$$



Dijkstra is Correct

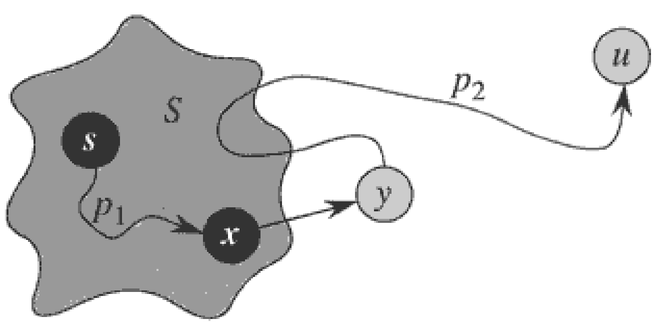
- (Theorem 24.6) Loop invariance:

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

  - **Initialization:** Initially, $S = \emptyset$, so trivially true
  - **Termination:** At end, $Q = \emptyset$, $S = V$, $v.d = \delta(s, v)$ for each vertex $v \in V$
  - **Maintenance:**
    - Let u be lightest vertex when u is added to S
    - Let p = shortest path from $s \sim> u$
    - Decompose p into p_1 and p_2
      - y = first vertex along p that's in V-S
      - $x \in S$ be y's predecessor

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Fig 24.7



Dijkstra Is Correct (2)

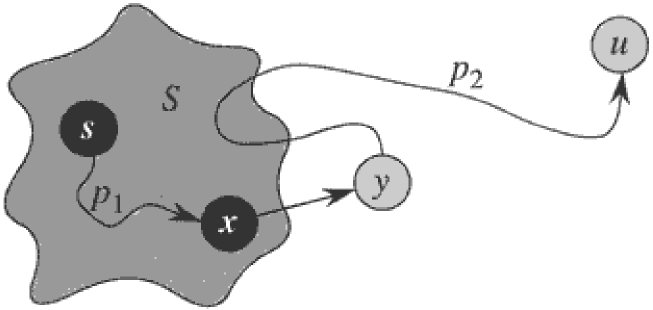
- when u is added to S,
  - $x.d = \delta(s, x)$
  - Relaxed (x, y), so $\boxed{y.d = \delta(s, y)}$ (convergence property)
- because y is before u on the shortest path p

$$\boxed{y.d = \delta(s,y) \le \delta(s,u) \le u.d}$$

- because u is lightest vertex (chosen before y)

$$\boxed{u.d \le y.d}$$

- So $\boxed{y.d = \delta(s, y) = \delta(s,u) = u.d}$ **QED**



Time Complexity

- **Similar to Prim's MST algorithm, keep a min-priority queue**
  - **but compute $v.d$, and use SP weights as keys**

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

	Binary Heap (min-heap)	Fibonacci heap
Insert, $\|V\|$ times	O(V)	O(V)
extract-min, $\|V\|$ times	O(V lg V)	O(V lg V)
Decreasekey $\|E\|$ times	O(E lg V)	O(E)
total	O((V+E) lg V) =O(E lg V)	O(E+ V lg V)

Dijkstra is O(E lg V)

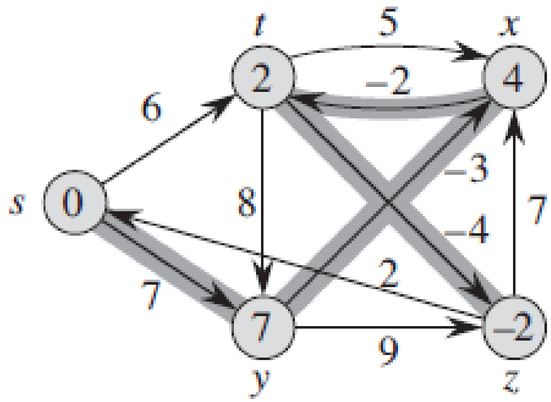
Summary

- **Single Source Shortest Paths, CH24**
  - Bellman-Ford Algorithm [1956~1958]
  - **Dijkstra's Algorithm [1959]**
    - $O(E \lg V)$
    - no negative weight allowed
  - Directed Acyclic Graph [1972]
  - Linear Programming
- All-pairs Shortest Paths, CH25

	complexity	notice
B-Ford	$O(VE)$	Negative weight ok
Dijkstra	$O(E \lg V)$	No negative weight
DAG	$\Theta(V+E)$	DAG only

FFT

- **Q: for graph with negative weight, can we add a positive number to all edges? so we can solve it faster**
  - **e.g. add 4 to all edges**



(e)