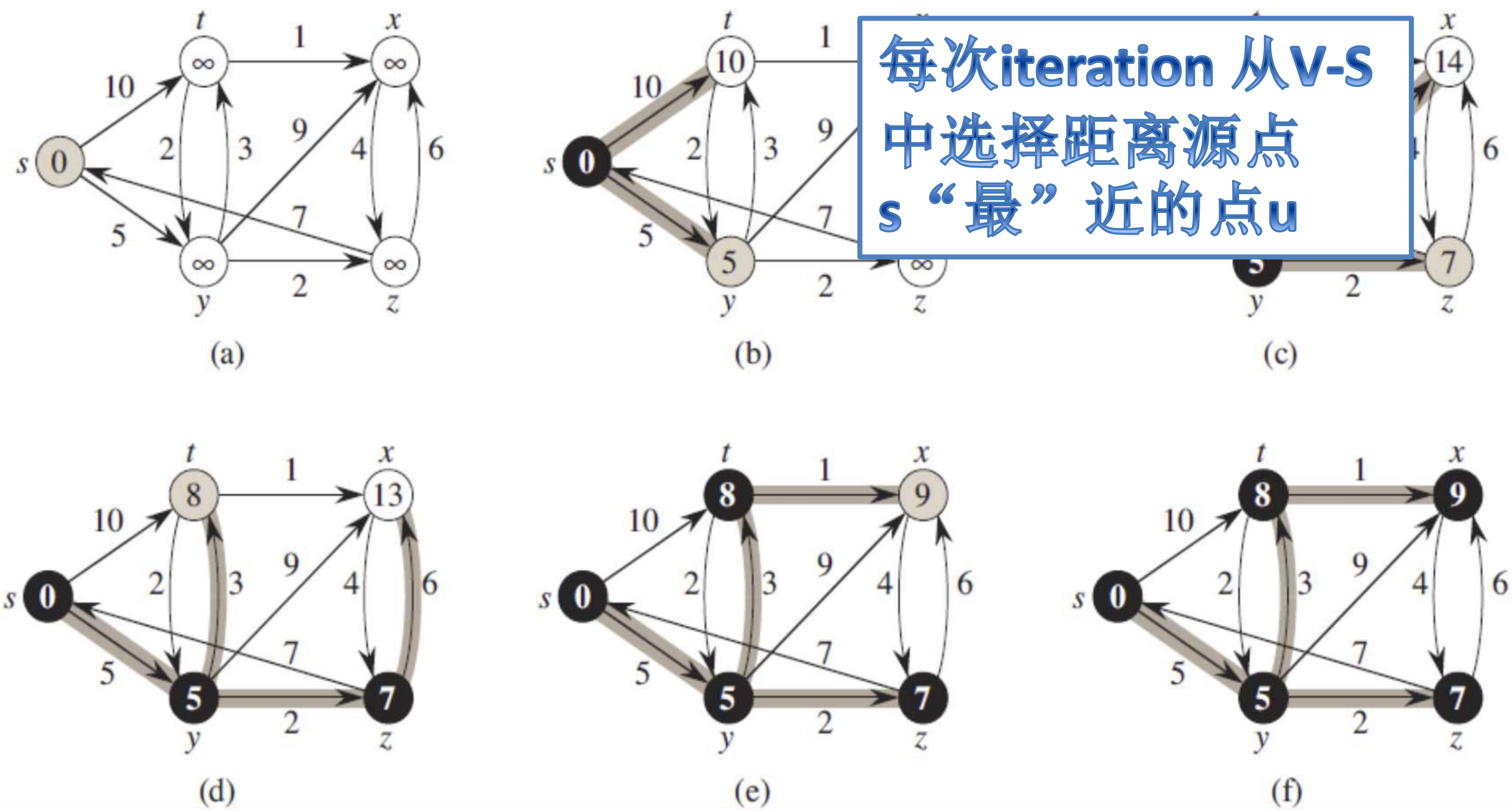
Dijkstra算法 正确性

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

问题10：为什么这被认为是一个Greedy算法?



每次iteration 从V-S中选择距离源点s “最” 近的点u

Dijkstra算法 正确性

```
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)
```

Corollary 24.7

If we run Dijkstra's algorithm on a weighted, directed graph $G = (V, E)$ with nonnegative weight function w and source s, then at termination, the predecessor subgraph G_π is a shortest-paths tree rooted at s.

predecessor-subgraph property

Theorem 24.6 (Correctness of Dijkstra's algorithm)

Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $u.d = \delta(s, u)$ for all vertices $u \in V$.

Dijkstra算法 正确性

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

Theorem 24.6 (Correctness of Dijkstra's algorithm)
Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $u.d = \delta(s, u)$ for all vertices $u \in V$.

Proof We use the following 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 for each vertex $u \in V$, we have $u.d = \delta(s, u)$ at the time when u is added to set S. Once we show that $u.d = \delta(s, u)$, we rely on the upper-bound property to show that the equality holds at all times thereafter.

```
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算法 正确性

- 初始阶段(**Initialization**)：
  - $S = \emptyset$, 不变式显然成立
- 运行期间(**Maintenance**):
  - We wish to show that in each iteration, $\boldsymbol{u.d = \delta(s, d)}$ **for the vertex $\boldsymbol{u}$ added to set S**.
- 终止时刻(**Termination**)

Termination: At termination, $Q = \emptyset$ which, along with our earlier invariant that $Q = V - S$, implies that $S = V$. Thus, $u.d = \delta(s, u)$ for all vertices $u \in V$. ■

运行期间(Maintenance)

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

假设： let u be the first vertex for which $u.d \neq \delta(s,d)$ when it is added to set S.

显然$u \neq s$

$S \neq \emptyset$(至少$s \in S$)

目标：找冲突 $u.d = \delta(s,d)$

策略

证明$y.d = \delta(s,y) = \delta(s,d) = u.d$

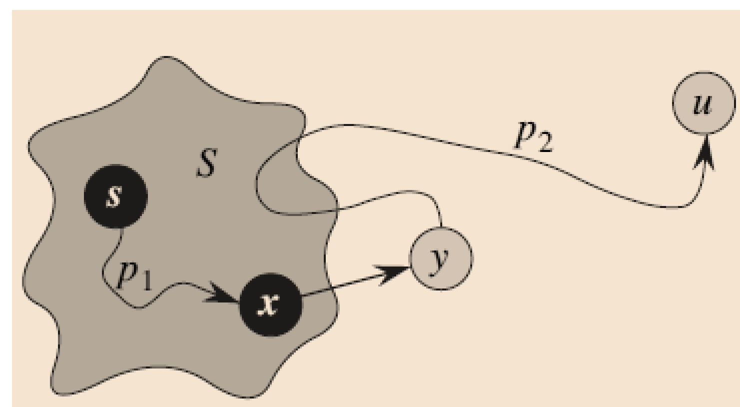
u, s之间一定存在通路

u, s之间一定存在某条最短通路P

令y是通路P上属于$V - S$的第一个点，x为y在P上的前驱节点，显然$x \in S$

P可以进一步划分为: $s \xrightarrow{P_1} x \rightarrow y \xrightarrow{P_2} u$

Either of paths p1 or p2 may have no edges



证明$y.d = \delta(s,y) = \delta(s,d) = u.d$

Convergence property (Lemma 24.14)
If $s \rightsquigarrow u \to v$ is a shortest path in G for some $u, v \in V$, and if $u.d = \delta(s,u)$ at any time prior to relaxing edge (u,v), then $v.d = \delta(s,v)$ at all times afterward.

- $y.d = \delta(s,y)$

when x is added to S: $x.d = \delta(s,x)$; Edge (x,y) was relaxed

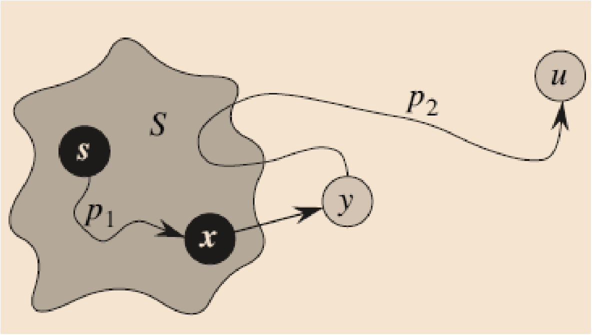
P是$s \to u$最短路径 → $p_1 + (x \to y)$是$s \to y$最短路径

→ $y.d = \delta(s,y)$ (Convergence property)

$$\begin{aligned} y.d &= \delta(s,y) \\ &\le \delta(s,u) \\ &\le u.d \end{aligned}$$

$u.d \le y.d$ (Dijkstra算法保证)

$$\begin{aligned} y.d &= \delta(s,y) \\ &= \delta(s,d) \\ &= u.d \end{aligned}$$



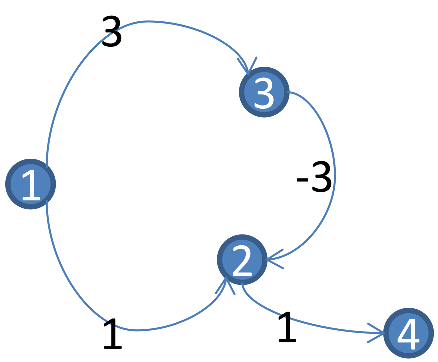
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

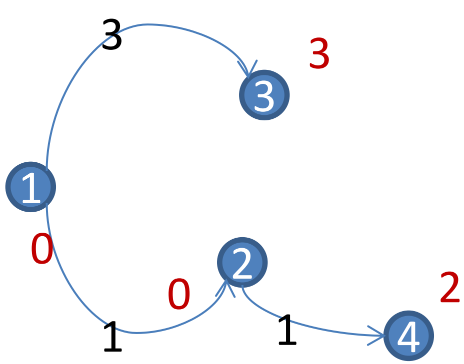
Proof We use the following loop invariant:

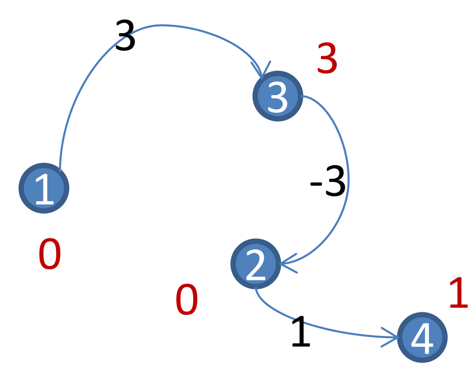
> At the start of each iteration of the **while** loop of lines 4–8, $v.d = \delta(s, v)$ for each vertex $v \in S$.

问题11:

Dijstra算法对每条边最多relax一次，而且不要求输入是DAG，它付出的代价是什么？为什么必须如此？







```
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)
```

显性或者隐性的
优先队列操作

问题12：

为什么说Dijstra算法的复杂度与其实现方法有关？

问题13：

你能比较一下Dijstra算法与计算最小生成树的Prim算法吗？Dijstra算法的结果是否一定是一个最小生成树？