Shortest Path Algorithms

Kuan-Yu Chen (陳冠宇)

2019/06/05 @ TR-310-1, NTUST

Dijkstra's Algorithm

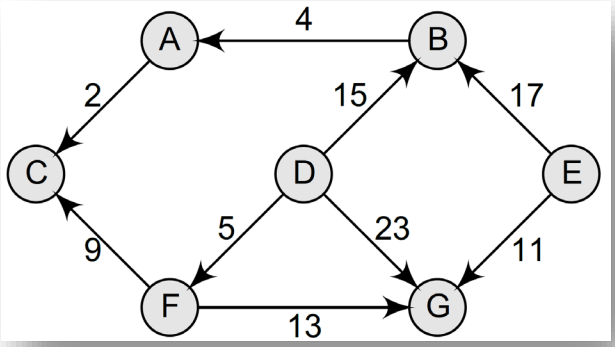
- Dijkstra's algorithm solves the single-source shortest-paths problem on a weighted, directed graph $G = (V, E)$ for the case in which all edge weights are nonnegative
  - $w(u, v) \geq 0$ for each edge (u, v)

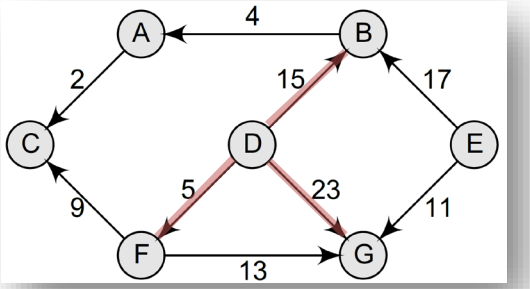
- Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined
  - Selects the vertex $u \in V - S$ with the minimum shortest path estimate
  - Adds u to S
  - Relaxes all edges leaving u

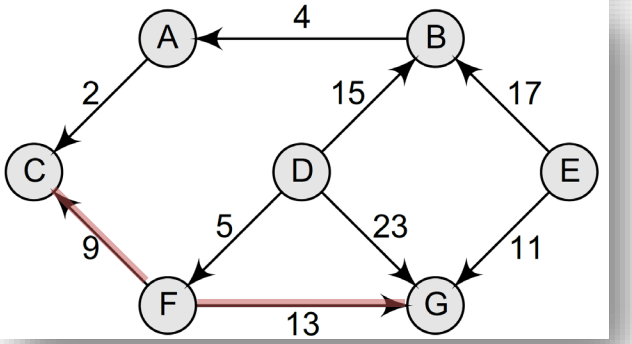
```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Example.

- Given a graph G, please take D as the initial node, and execute the Dijkstra's algorithm on it
  - Step 1:
    - Set $D.d = 0$
  - Step 2:
    - Select D and $S = \{D\}$
    - By RELAX, $B.d = 15$, $G.d = 23$, and $F.d = 5$
    - $B.d = 15$, $G.d = 23$, and $F.d = 5$
  - Step 3:
    - Select F and $S = \{D, F\}$
    - By RELAX, $\boldsymbol{G.d = 18}$ and $C.d = 14$
    - $B.d = 15$, $G.d = 18$ and $C.d = 14$

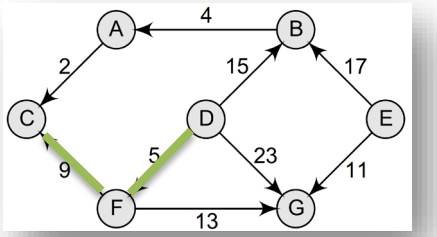




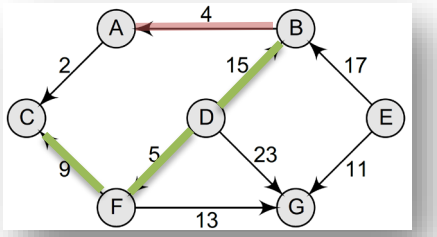


Example.

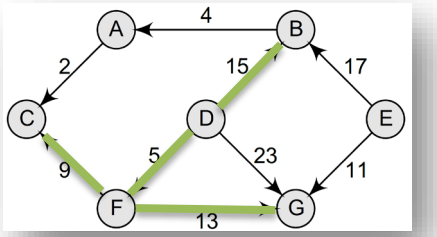
- Step 4:
  - Select C and $S = \{D, F, C\}$
  - By RELAX, no value should be change!
  - $B.d = 15$, $G.d = 18$



- Step 5:
  - Select B and $S = \{D, F, C, B\}$
  - By RELAX, $A.d = 19$
  - $G.d = 18$ and $A.d = 19$



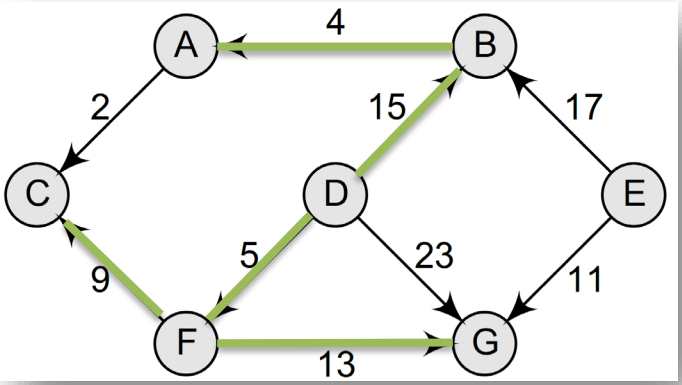
- Step 6:
  - Select G and $S = \{D, F, C, B, G\}$
  - By RELAX, no value should be change!
  - $A.d = 19$



Example.

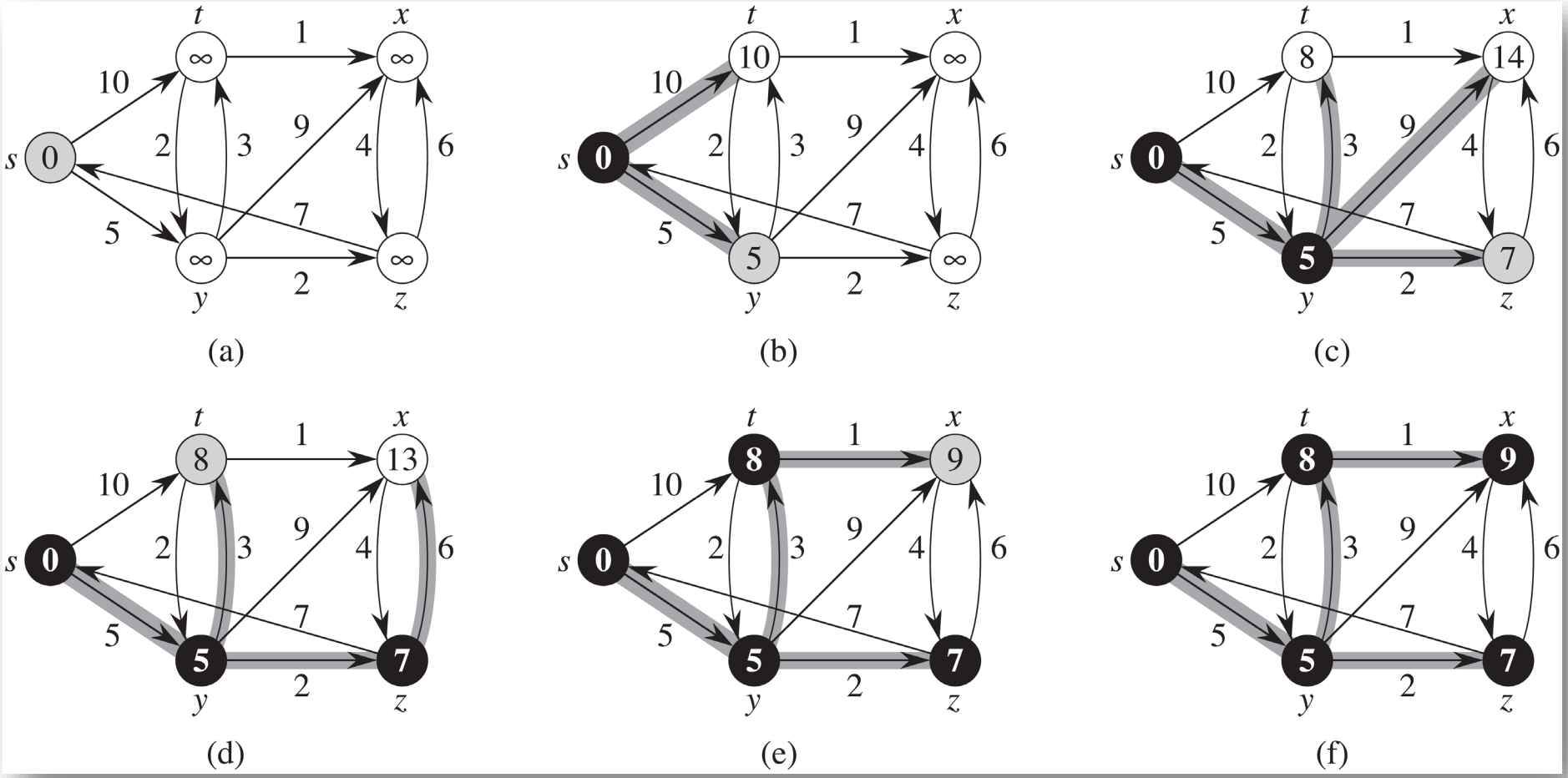
- Step 7:
  - Select A and $S = \{D, F, C, B, G, A\}$
  - By RELAX, no value should be change!

- Step 8:
  - Select E and $S = \{D, F, C, B, G, A, E\}$
  - By RELAX, no value should be change!
  - Terminated!



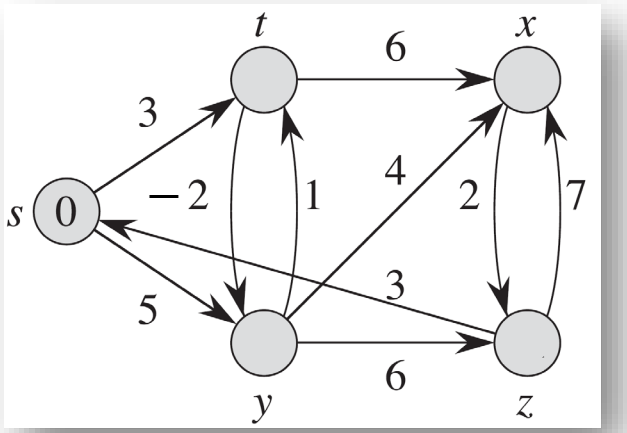
Example..

- The source s is the leftmost vertex



Bellman-Ford Algorithm

- The ***Bellman-Ford algorithm*** solves the single-source shortest-paths problem in the general case in which edge weights may be negative
  - The Bellman-Ford algorithm returns a boolean value indicating whether or not there is a negative-weight cycle that is reachable from the source
    - If there is such a cycle, the algorithm indicates that no solution exists (return false)
    - If there is no such cycle, the algorithm produces the shortest paths and their weights



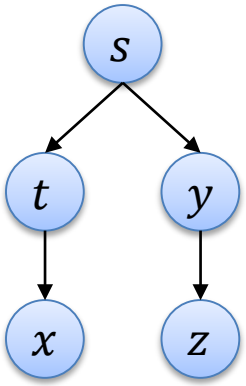

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

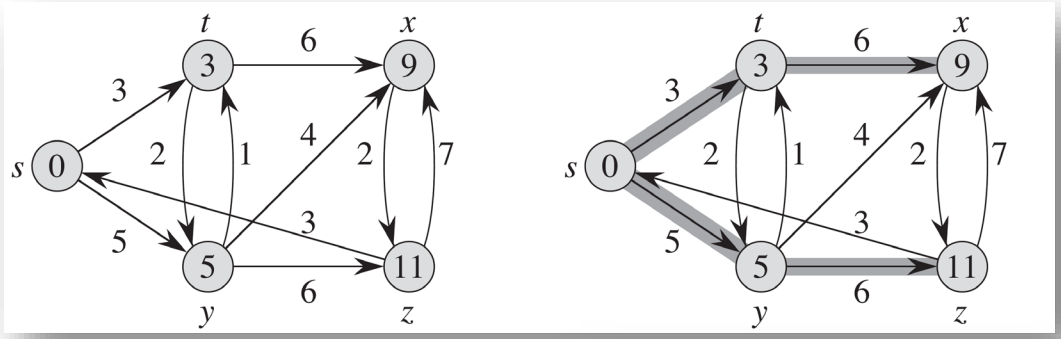
Correctness

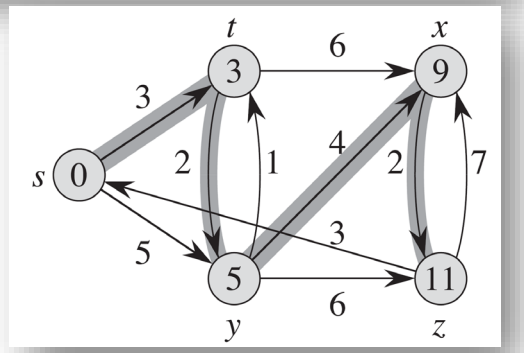
- To prove the correctness of the Bellman-Ford algorithm, we start by showing that if there are no negative-weight cycles
  - Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w: E \to \mathbb{R}$, and assume that G contains no negative-weight cycles that are reachable from s. Then, after the $|V| - 1$ iterations of the relaxing for all edges, we have $v.d = \delta(s, v)$ for all vertices v that are reachable from s
  - Proof:
    - Let $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from v_0 to v_k
    - It is easy to understand that p has at most $|V| - 1$ edges
    - Since the Bellman-Ford algorithm performs $|V| - 1$ iterations of the relaxing of all the edges, we thus say it performs a sequence of relaxation steps in order for p, i.e., $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$
    - By the path-relaxation property, after the Bellman-Ford algorithm, $v_k.d = \delta(v_0, v_k) = \delta(s, v_k)$

Shortest Path Tree

- The shortest path can be treated as a tree, where the source vertex is the root node
  - Since the shortest paths from source to all other vertexes are not unique, the shortest path trees for a graph are also not unique

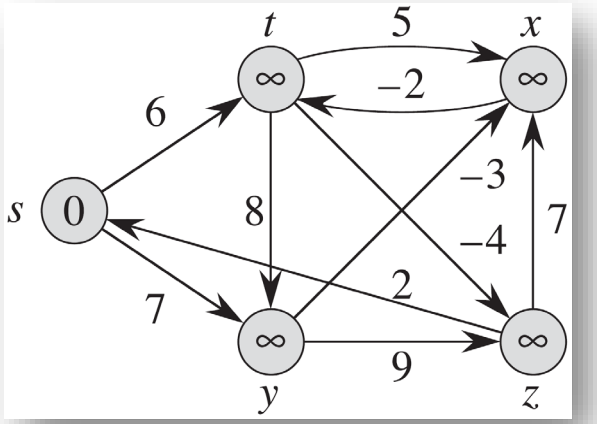




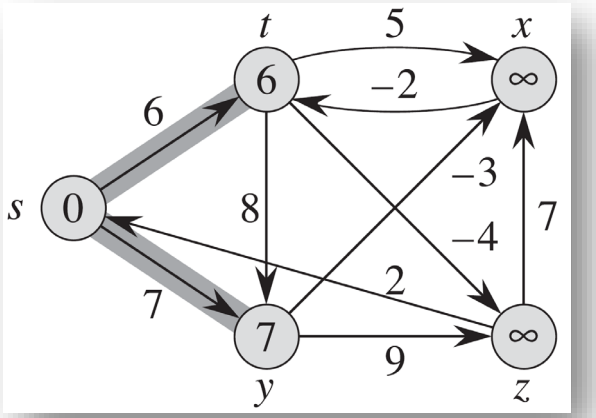


Example.

- The source is vertex s, and we assume that each pass relaxes the edges in the order $(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

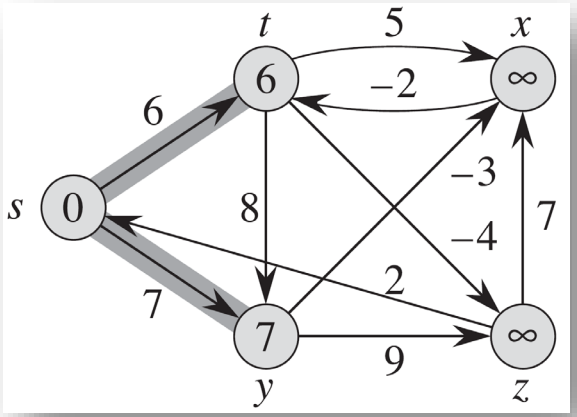


  - Iteration1:
    - (s,t): $t.d = 6$
    - (s,y): $y.d = 7$

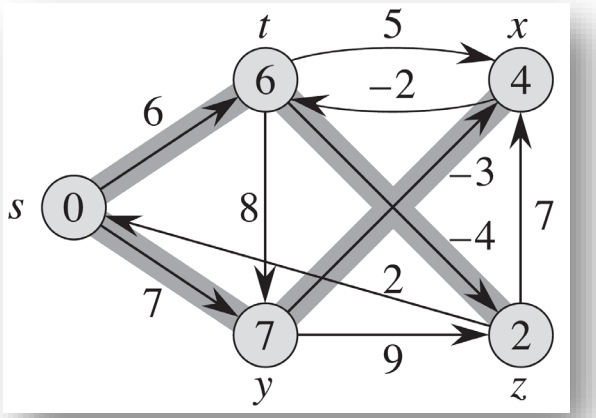


Example..

- The source is vertex s, and we assume that each pass relaxes the edges in the order $(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$

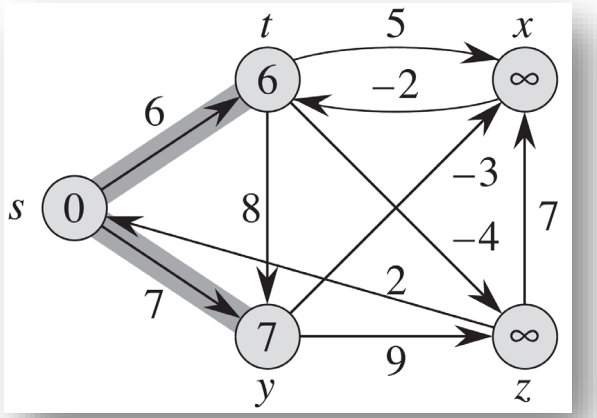


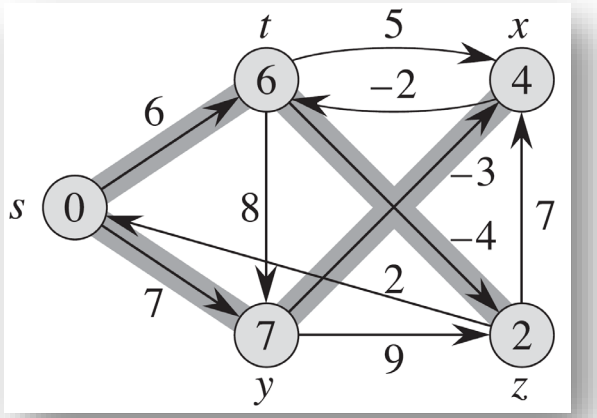
  - Iteration2:
    - (t,x): $x.d = 11$
    - (t,y): $y.d = 7$ *unchange*
    - (t,z): $z.d = 2$
    - (x,t): $t.d = 6 < 11 - 2$ *unchange*
    - …



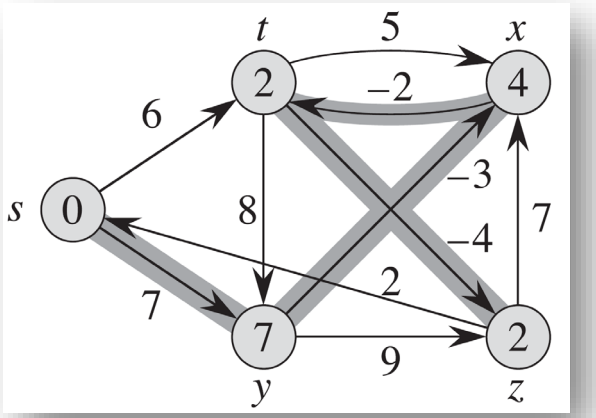
Example...

- The source is vertex s, and we assume that each pass relaxes the edges in the order $(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$



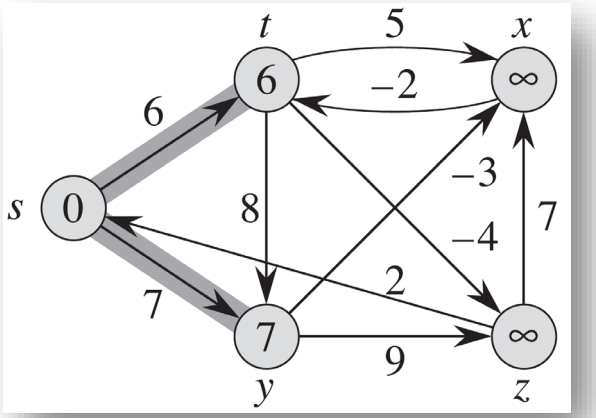


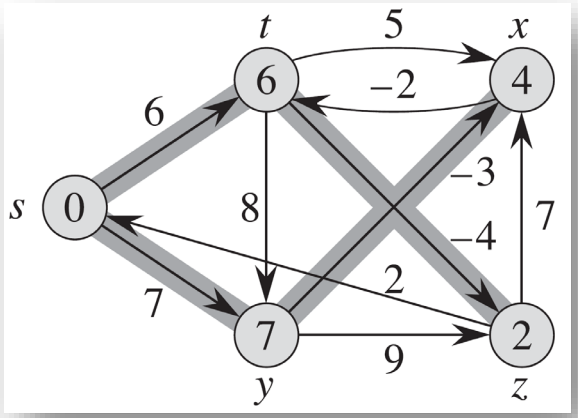
  - Iteration3:
    - (t,x): $x.d = 4 < 2 + 5$ *unchange*
    - (t,y): $y.d = 7 < 2 + 8$ *unchange*
    - (t,z): $z.d = 2 = 6 - 4$ *unchange*
    - (x,t): $t.d = 2 = 4 - 2 < 6$
    - ...

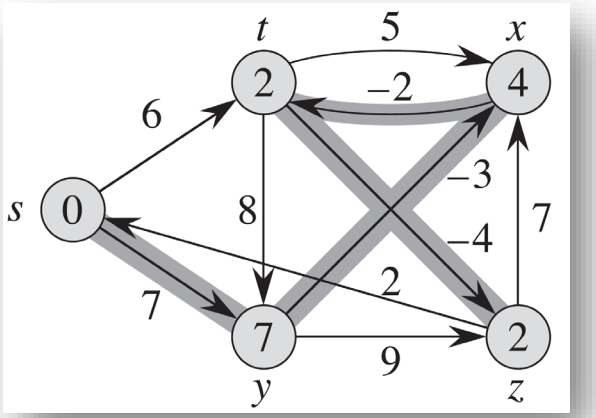


Example….

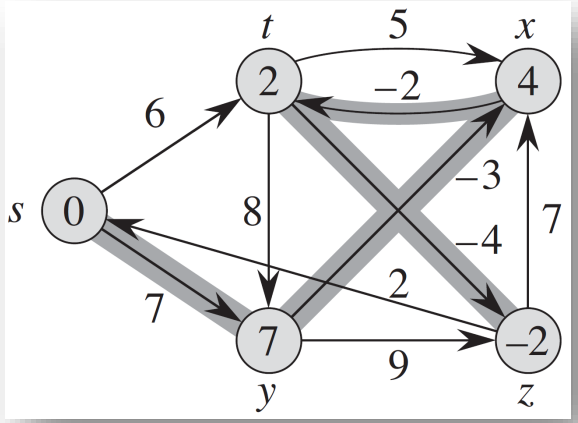
- The source is vertex s, and we assume that each pass relaxes the edges in the order
  $(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$







- Iteration4:
  - (t,x): $x.d = 4 < 2 + 5$ *unchange*
  - (t,y): $y.d = 7 < 2 + 8$ *unchange*
  - (t,z): $z.d = -2 = 2 - 4$
  - (x,t): $t.d = 2 = 4 - 2$ *unchange*
  - …



Questions?

kychen@mail.ntust.edu.tw