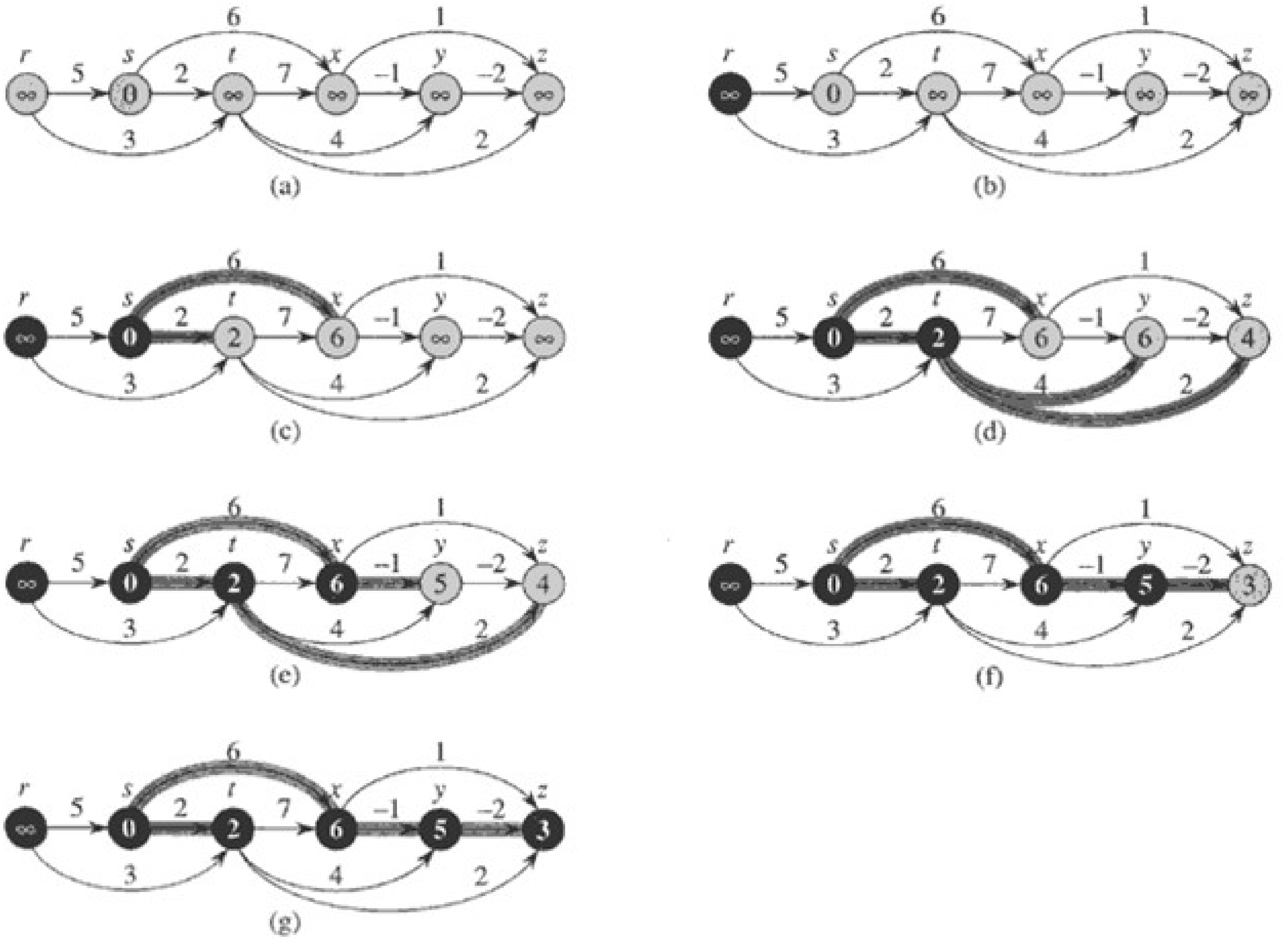
Lecture 8
Single-Source Shortest Paths Problems Continued

1. Single-Source Shortest Paths in Directed Acyclic Graphs
2. Dijkstra’s Algorithm
3. Difference constraints

Single-Source Shortest Paths in Directed Acyclic Graphs

```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u, taken in topologically sorted order
4       do for each vertex v ∈ Adj[u]
5              do RELAX(u, v, w)
```

Time Complexity: $O(V + E)$


(a)
(b)
(c)
(d)
(e)
(f)
(g)

Correctness of the Algorithm

- Theorem 24.5
  - If a weighted, directed graph $G = (V, E)$ has source vertex s and no cycles, then at the termination of the DAG-SHORTEST-PATHS procedure, $d[v] = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph G_π is a shortest-paths tree rooted at s.
- Proof.
  - We first show that $d[v] = \delta(s, v)$ for all vertices $v \in V$ at the termination.
  - If v is not reachable from s, then $d[v] = \delta(s, v)=\infty$ by the no-path property.

Proof Continued

- Now, Suppose that there is a shortest path $P=<v_0,v_1,\ldots,v_k>$, where $v_0=s$ and $v_k=v$. Since we process the vertices in topological order, the edges on P are relaxed in the order (v_0,v_1), (v_1,v_2), …., (v_{k-1},v_k).The path relaxation property implies that $d[v_i]=\delta(s, v_i)$ for all i.
- By the predecessor-subgraph property, G_π is a shortest-paths tree.

Application: Determine Critical Path

- Pert Chart:
  - edges represent jobs and weights represent the times required to perform particular jobs.
  - If edge (u,v) enters v and edge (v,w) leaves v, then job (u,v) must be performed prior to job (v,w).
  - A path through this dag represents a sequence of jobs that must be performed in a particular order.
  - A critical path is a longest path through the dag.

Two method

- Negating the edge weight.
- Replace “∞” with “- ∞” and “>” with “<”.

Dijkstra's Algorithm

- Problem solved
- Correctness
- Time complexity
- Similarities to BFS and PRIM

Problem Solved by Dijkstra's

- Single-source shortest-paths problem
- Edge weight >=0

- Input: A graph $G=(V, E)$ and a source s, and a nonnegative function w: $E \rightarrow R^+$
- Output: For each vertex v, shortest path weight $\delta(s,v)$, and a shortest path if exists.

The Dijkstra Algorithm

```
DIJKSTRA(G,w,s)
Initialize-Single-Source(G,s)
S←∅
Q←V[G]
while Q≠∅
do u←EXTRACT-MIN(Q)
S←S∪{u}
for each v ∈Adj[u]
      do RElAX(u,v,w)
```

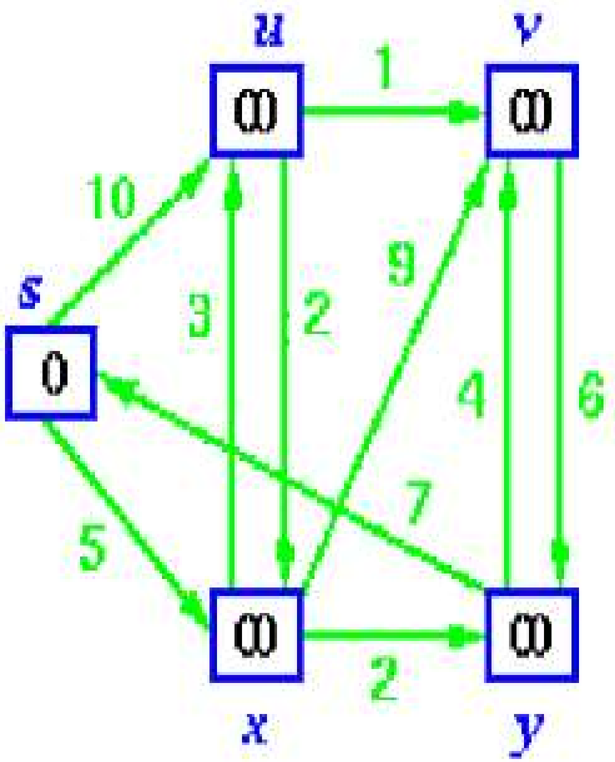
```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ V[G]
      do d[v] ← ∞
         π[v] ← NIL
d[s] ← 0
```

```
RELAX(u, v, w)
if d[v] > d[u] + w(u, v)
   then d[v] ← d[u] + w(u, v)
        π[v] ← u
```

Dijkstra's Algorithm - example

- Initial Graph

Source Mark 0



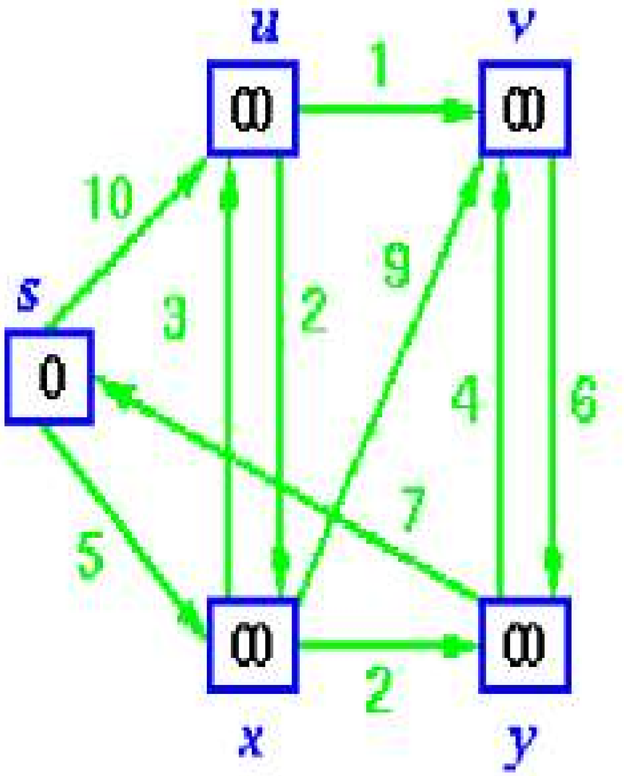
Distance to all nodes marked ∞

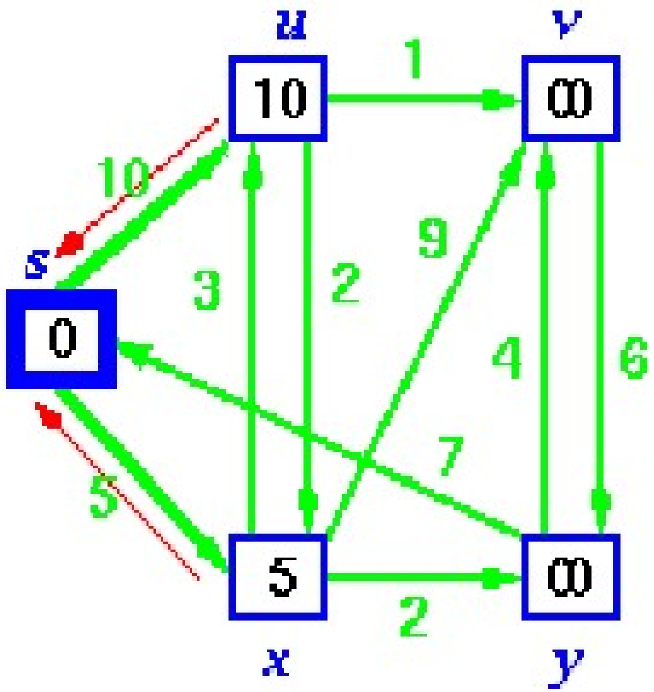
Dijkstra's Algorithm - Operation

- Initial Graph

S={s}

Source

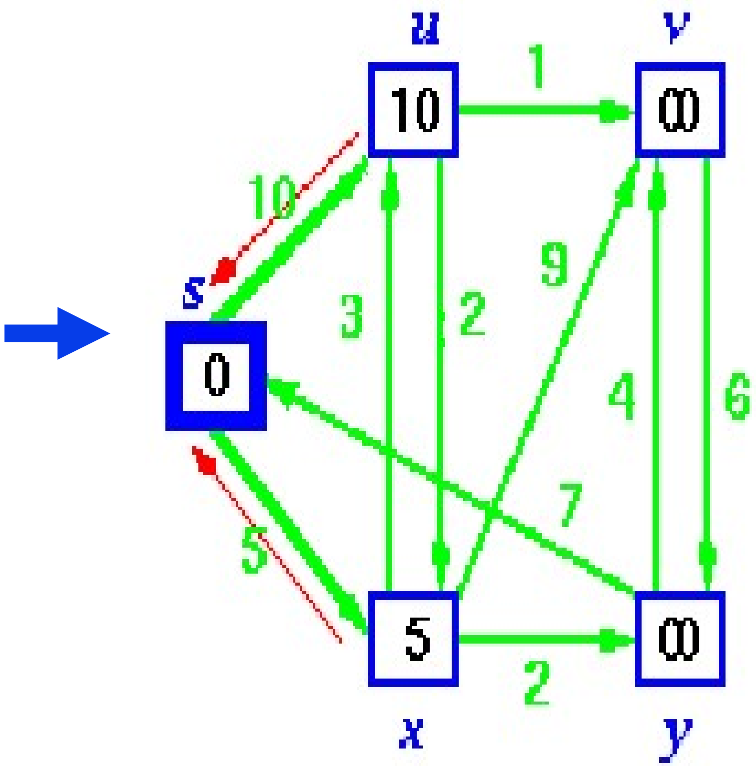




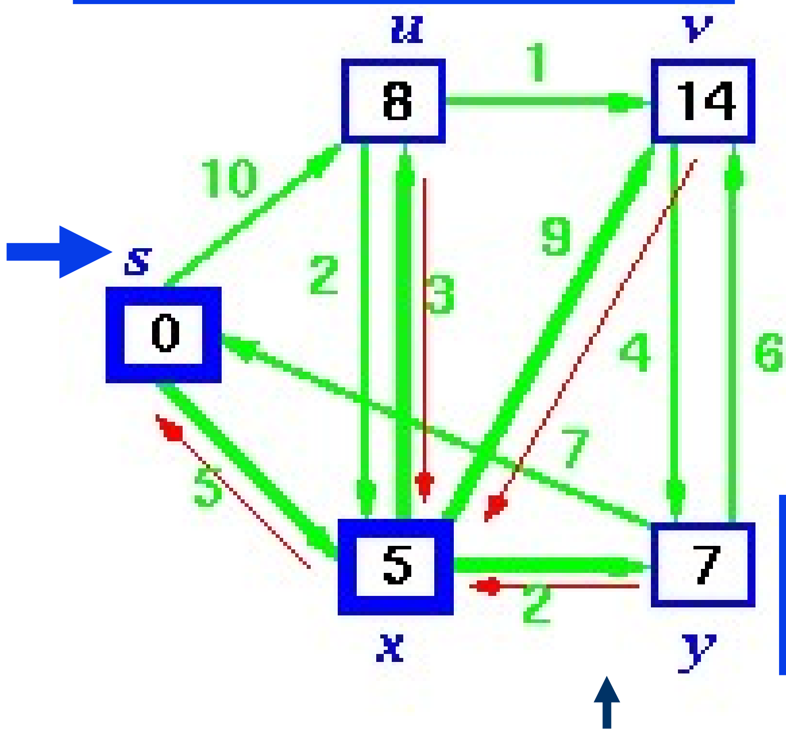
Relax edges leaving source

S= ∅

Dijkstra's Algorithm - Operation



Relax *u* and a shorter path via *x* exists



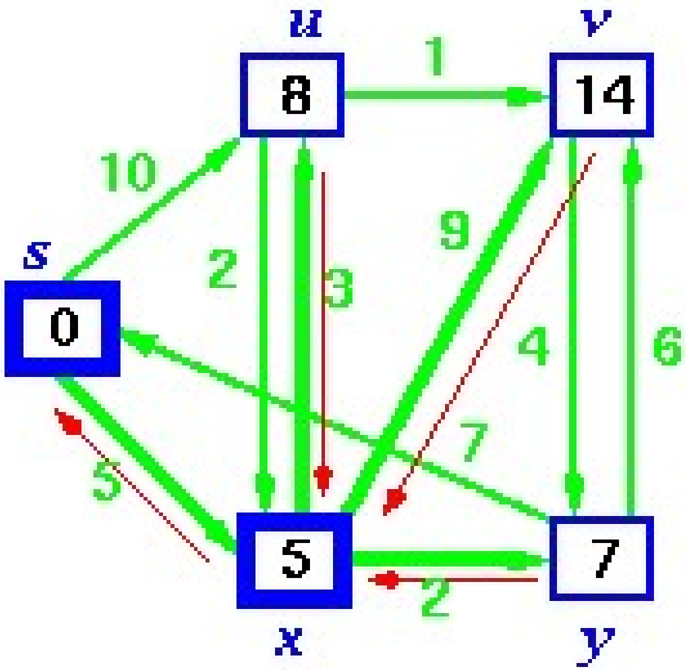
Relax *y*

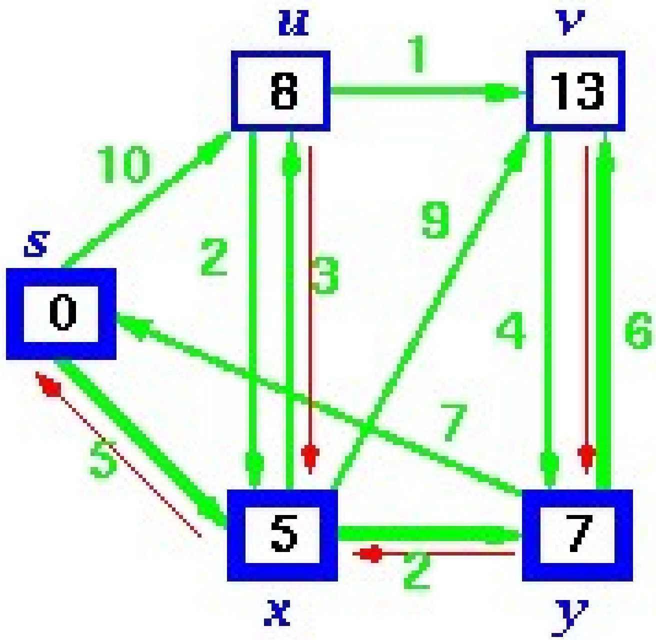
Red arrows show predecessors

Sort vertices and choose closest

Dijkstra's Algorithm - Operation

Relax v and a shorter path via y exists



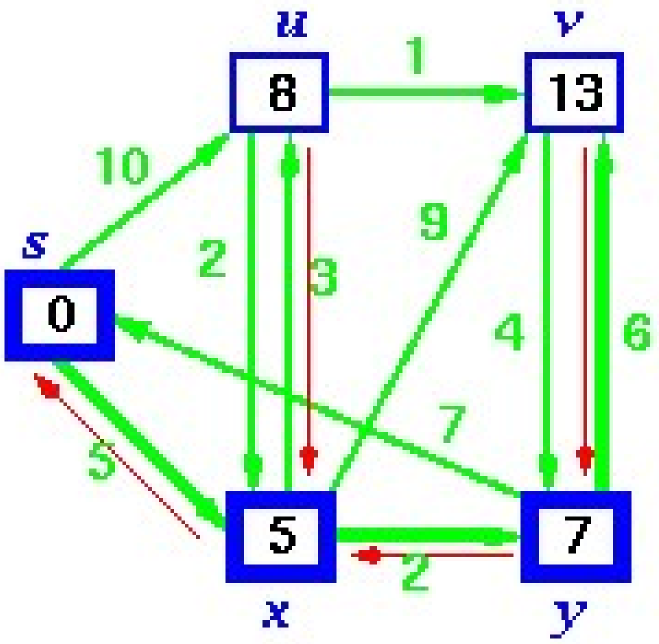


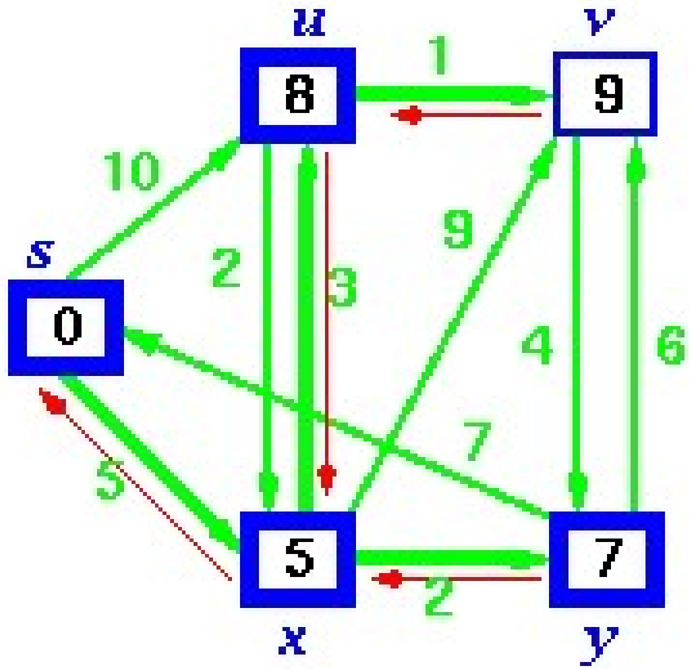
S is now $\{ s, x \}$

Sort vertices and choose closest

Dijkstra's Algorithm - Operation

Update d[v]

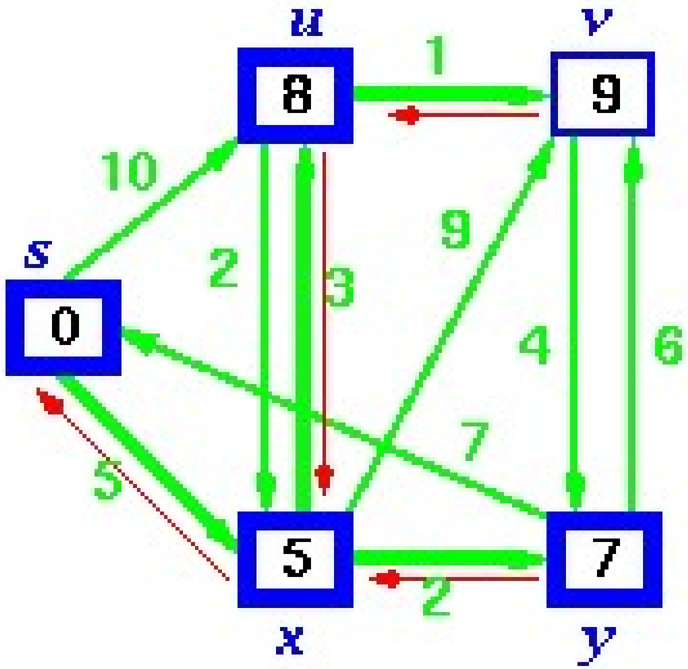




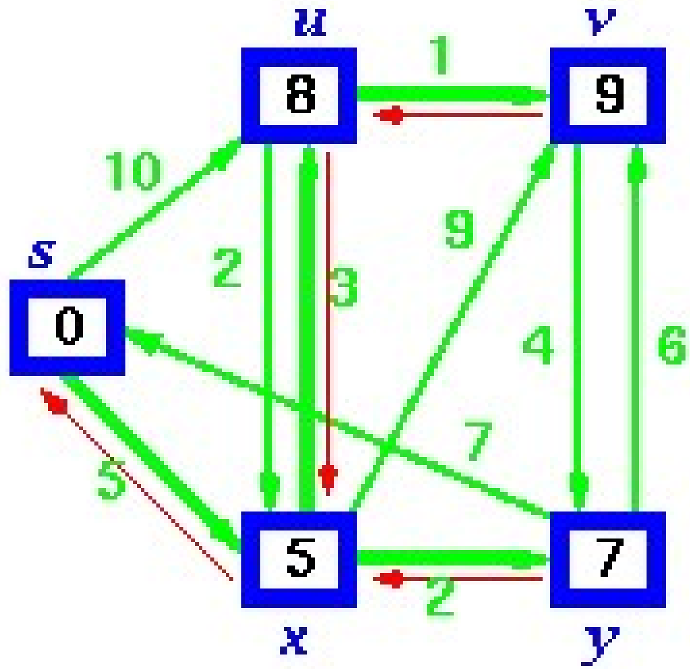
S is now { s, x, y }

Sort vertices and choose closest, u

Dijkstra's Algorithm - Operation

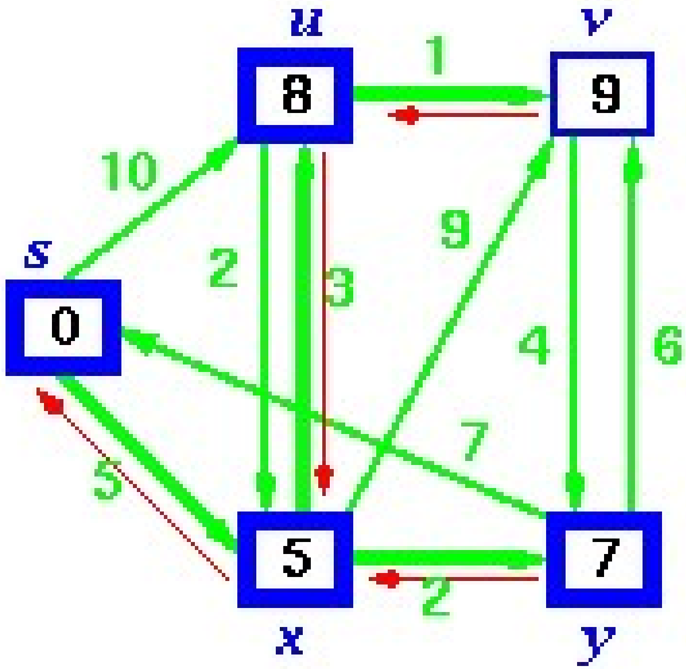


S is now { *s*, *x*, *y*, *u* }

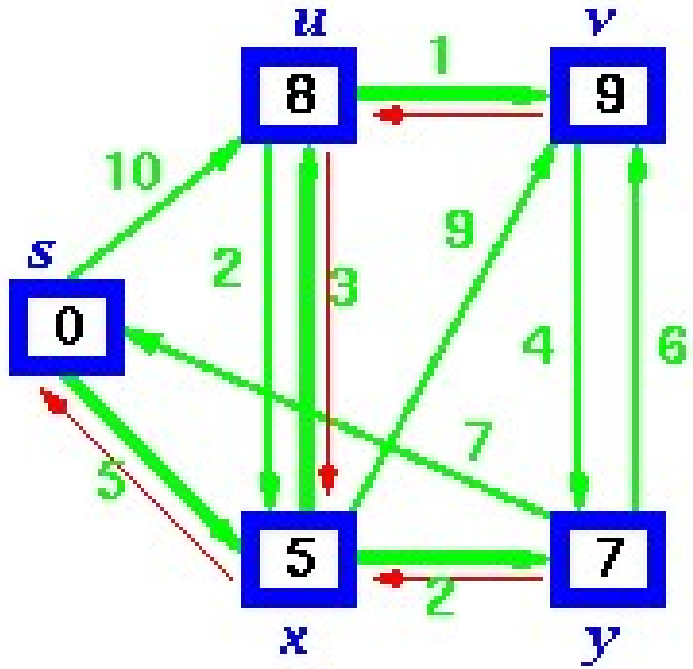


Finally add *v*

Dijkstra's Algorithm - Operation



S is now { *s, x, y, u* }



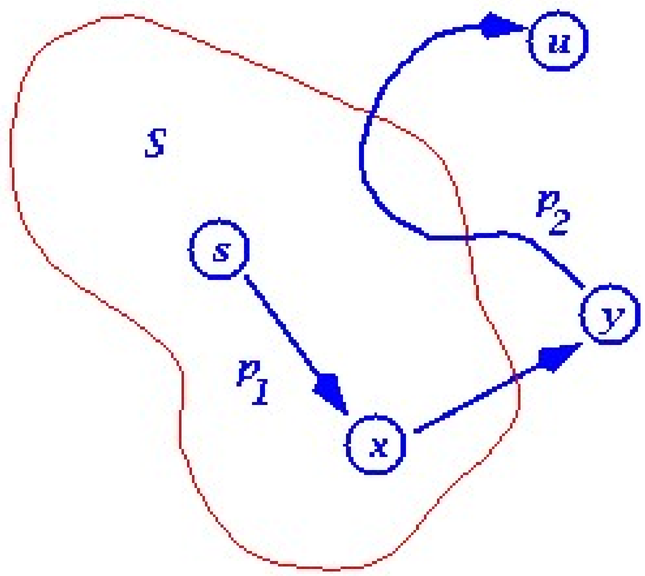
Pre-decessors show shortest paths sub-graph

Correctness of Dijkstra's

- Theorem 24.6
  - Dijkstra's algorithm, run on a weighted,directed graph $G=(V, E)$, with non-negative weight function w and source s, terminates with $d[u]= \delta(s,u)$ for all vertices $u \in V$.
- Proof *(by contradiction)*
  - Since $S=V$ in the end, we only need to prove that for each vertex v added to S, there holds $d[v]= \delta(s, v)$ when v is added to S.
  - Suppose that ***u* is the first** vertex for which $d[u] \neq \delta(s, u)$ when it was added to S
  - Note
    - u is not s because $d[s] = 0= \delta(s, s)$
    - There must be a path $s \rightarrow \ldots \rightarrow u$, since otherwise $d[u]= \delta(s, u) = \infty$.
    - Since there's a path, there must be a shortest path (note there is no negative cycle).

Dijkstra's Algorithm - Proof

- Let $s \rightsquigarrow x \rightarrow y \rightsquigarrow u$ be a shortest path from s to u,
  where at the moment u is chosen to S,
  x is in S and y is the first outside S
- When x was added to S, $d[x] = \delta(s, x)$
- Edge $x \rightarrow y$ was relaxed at that time,
  so at time u is chosen, $d[y] = \delta(s, y)$

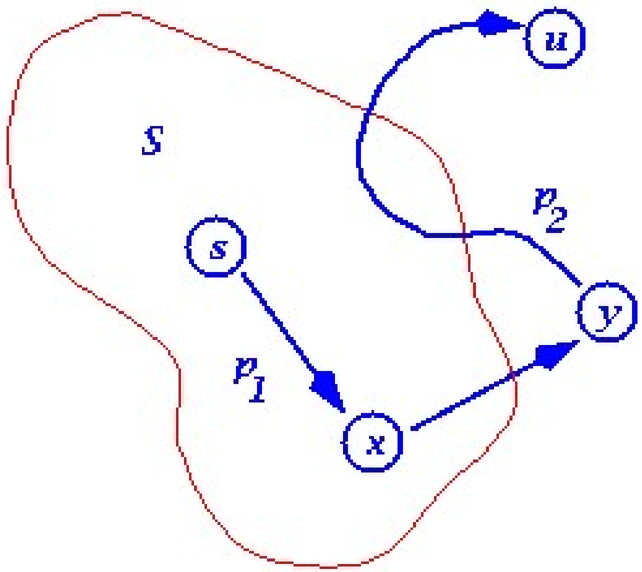


Dijkstra's Algorithm - Proof

- So, $d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$
- But, when we chose u,
  both u and y are in Q,
  so $d[u] \leq d[y]$
  (otherwise we would have chosen y)
- Thus the inequalities must be equalities

$\therefore\ d[y] = \delta(s, y) = \delta(s, u) = d[u]$

- And our hypothesis ($d[u] \neq \delta(s, u)$) is contradicted!



- 1.Why the red inequality holds? How about negative edges exist?
- 2.When u is added to S, a shortest path s→u in S is found.

Dijkstra's Algorithm - Time Complexity

- Like that of PRIM's
  - If using arrays
    - $O(V^2)$
  - Using special data structure
    - $O(V \lg V + E)$ or less

Compared to BFS

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

```
BFS(G,s)
1.  For each vertex u ∈V,
      color[u]=W; d[u]=∞;
      π[u]=NIL;  color[s]=G; d[s]=0;
2.  Q={s}.
3.  while Q is nonempty
      u=dequeue(Q)
      for each v ∈ adj[u],
          if(color[v]==W), do
              color[v]=G;
              d[v]=d[u]+1;
              π[v]=u;
              enqueue(Q,v);
      color[u]=B
```

Note: vertices in Q are in non-decreasing order of d, and there are only two values d, and d+1 in Q.

Compared to PRIM

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

```
MST-PRIM(G, w, r)
for each u ∈ V[G]
    do key[u] ← ∞
       π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
       for each v ∈ Adj[u]
           do if v ∈ Q and w(u, v) < key[v]
                 then π[v] ← u
                      key[v] ← w(u, v)
```

Linear Programming

- Linear programming problem:
  - Input: matrix A_{mn}, vector b_m, and vector c_n.
  - Output: vector x_n, such that

$$\text{maximize} \quad cx$$
$$\text{subject to} \quad Ax \le b.$$

- Notes
  - This problem has polynomial time solution.
  - Many problems can be converted into LP.

An Instance

单位产品所需原料数量（公斤）	产品 Q1	产品 Q2	产品 Q3	原料供用量（公斤/日）
原料P1	2	3	0	1500
原料P2	0	2	4	800
原料P3	3	2	5	2000
单位产品的利润（千元）	3	5	4	

LP

$$\max \quad 3x_1 + 5x_2 + 4x_3$$

s.t.

$$2x_1 + 3x_2 \leq 1500$$

$$2x_2 + 4x_3 \leq 800$$

$$3x_1 + 2x_2 + 5x_3 \leq 2000$$

$$x_1, x_2, x_3 \geq 0$$

Feasibility problem of LP

- Feasible solution: x_n, subject to $Ax<=b$.
- Feasibility problem of LP:
  - Given Matrix A $m \times n$, b m-vector
  - Output: either a feasible solution *x* if one exists, or a judgment that no feasible solution exists.

Systems of difference constraints

- Each row of A contains exactly one "1" and one "-1". All other elements are "0".
- Each row is a difference constraint:
  - $x_j - x_i <= b_k$, $1 <= k <= m$.
- An example

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

$x_1 - x_2 <= 0$
$x_1 - x_5 <= -1$
$x_2 - x_5 <= 1$
$x_3 - x_1 <= 5$
$x_4 - x_1 <= 4$
$x_4 - x_3 <= -1$
$x_5 - x_3 <= -3$
$x_5 - x_4 <= -3$

Solution x=(-5,-3,0,-1,-4); x'=(-5+d,-3+d,0+d,-1+d,-4+d)
(0,2,5,4,1)

Relations between solutions to DC

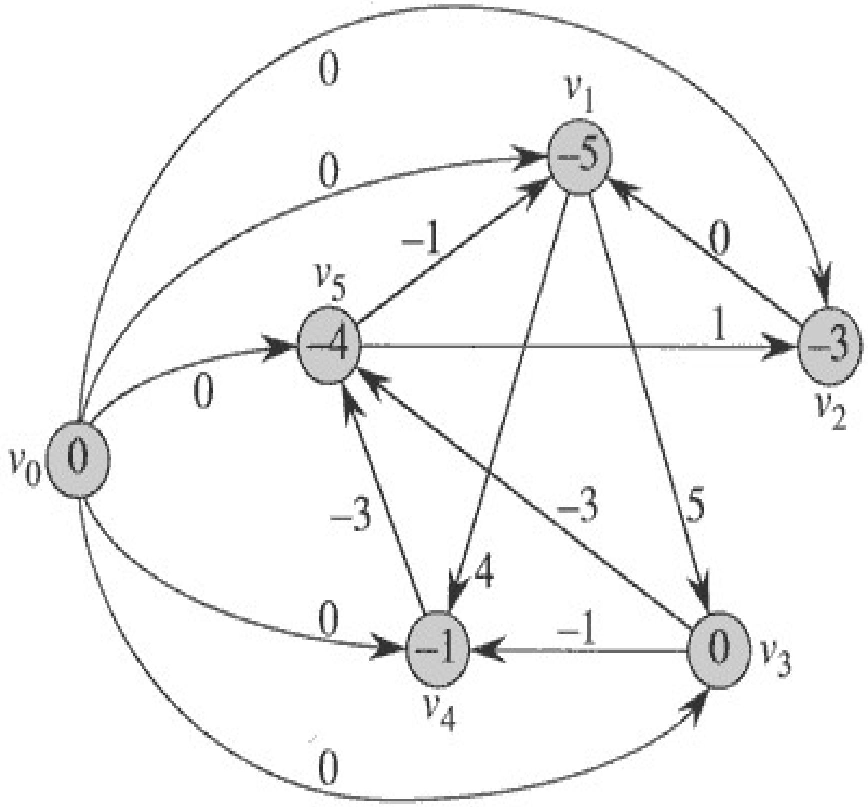
- Lemma 24.8: Let $x=(x_1,x_2,\ldots,x_n)$ be a solution to a system $Ax<=b$ of difference constraints, and let d be any constant. Then $x+d=(x_1+d, \ldots, x_n+d)$ is a solution to $Ax<=b$ as well.

- Proof. Easy(Book236)

Constraints graph

- Given a system $Ax \le b$ of difference constraints
- The constraint graph is a weighted, directed graph $G=(V, E;w)$, where:
  - $V=\{v_0,v_1,\ldots,v_n\}$ //v_0 is an extra vertex
  - $E=\{(v_i,v_j):x_j-x_i \le b_k$ is a constraint$\}$ $\cup\{(v_0,v_1),(v_0,v_2),\ldots,(v_0,v_n)\}$. //each vertex is reachable from v_0
  - $w(v_i,v_j)=b_k$, if $x_j-x_i \le b_k$ is a constraint;
  - $w(v_0,v_i)=0$.

Constraint graph of example



$$
\begin{pmatrix}
1 & -1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & -1 \\
-1 & 0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 \\
0 & 0 & -1 & 1 & 0 \\
0 & 0 & -1 & 0 & 1 \\
0 & 0 & 0 & -1 & 1
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}
\leq
\begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}
$$

Figure 24.8 The constraint graph corresponding to the system (24.3)–(24.10) of difference constraints. The value of $\delta(v_0, v_i)$ is shown in each vertex v_i. A feasible solution to the system is $x = (-5, -3, 0, -1, -4)$.

Solve Difference Constraints by SSSP in constraint graph

- Thoerem24.9: Given a system Ax<=b of difference constraints, let G=(V, E) be the corresponding constraint graph.
  - If G contains no negative-weight cycles, then

    $$x=(\delta(v_0,v_1), \delta(v_0,v_2), \ldots, \delta(v_0,v_n))$$

    is a feasible solution for the system.
  - If G contains a negative-weight cycle, then there is no feasible solution for the system.

Correctness of Theorem 24.9

- Proof. ($\Rightarrow$)If G does not contain negative-weight cycles. Then shortest path is well defined. And since there is an edge from v_0 to each vertex, all shortest paths from v_0 is finite. Consider any edge (v_i,v_j)(any constraint). By the triangle inequality, $\delta(v_0,v_j) \leq \delta(v_0,v_i)+w(v_i,v_j)$ or, equivalently, $\delta(v_0,v_j)- \delta(v_0,v_i) \leq w(v_i,v_j)$. Thus, letting $x_i= \delta(v_0,v_i)$ and $x_j= \delta(v_0,v_j)$ satisfies the difference constraint $x_j-x_i \leq b_k =w(v_i,v_j)$ that corresponds to edge (v_i,v_j).

Proof continued

- If G contains a negative-weight cycle $C=<v_1,v_2,\ldots,v_k>$, where $v_1=v_k$. We will show that there is no feasible solution. Cycle *C* corresponds to the following difference constraints: $x_2-x_1<=w(v_1,v_2)$; $x_3-x_2<=w(v_2,v_3)$; …, $x_k-x_{k-1}<= w(v_{k-1},v_k)$, $x_1-x_k<=w(v_k,v_1)$. If there is a feasible solution x, it will satisfy all these k inequalities. It also satisfies the summation of the inequalities, i.e., $0<=w(C)$, a contradiction to C is negative-weighted.

Property of DC

- Observation(According to Theorem24.9):
  - A difference constraints system has a solution $\Leftrightarrow$ constraint graph contains no negative cycle, and $x=(\delta(v_0,v_1), \delta(v_0,v_2), \ldots, \delta(v_0,v_n))$ is a solution.
  - A difference constraints system has no solution $\Leftrightarrow$ constraint graph contains negative cycles.

Bellman-Ford solution to DC

Algorithm:

- Compute constraint graph G;
- Run Bellman-Ford on constraint graph G;
- If Bellman-Ford returns True, then $x=(\delta(v_0,v_1), \ldots, \delta(v_0,v_n))$ is a solution to DC; if Bellman-Ford returns false, DC has no solution.

Conclusion

- Relaxation and its properties
- Bellman-Ford algorithm
- Single-source shortest paths in DAG
- Dijkstra's algorithm
- Difference constraints

Homework

- 24.3-2
- 24.3-3

Next Class

- All-pairs shortest paths problem

实验五

- 给定一个有向加权完全图，每条边的权重为正，求最小的TSP圈。
  - TSP圈：经过所有点，且每个点仅经过一次的圈。
  - 方法自选，动态规划、分支定界、枚举都可以。