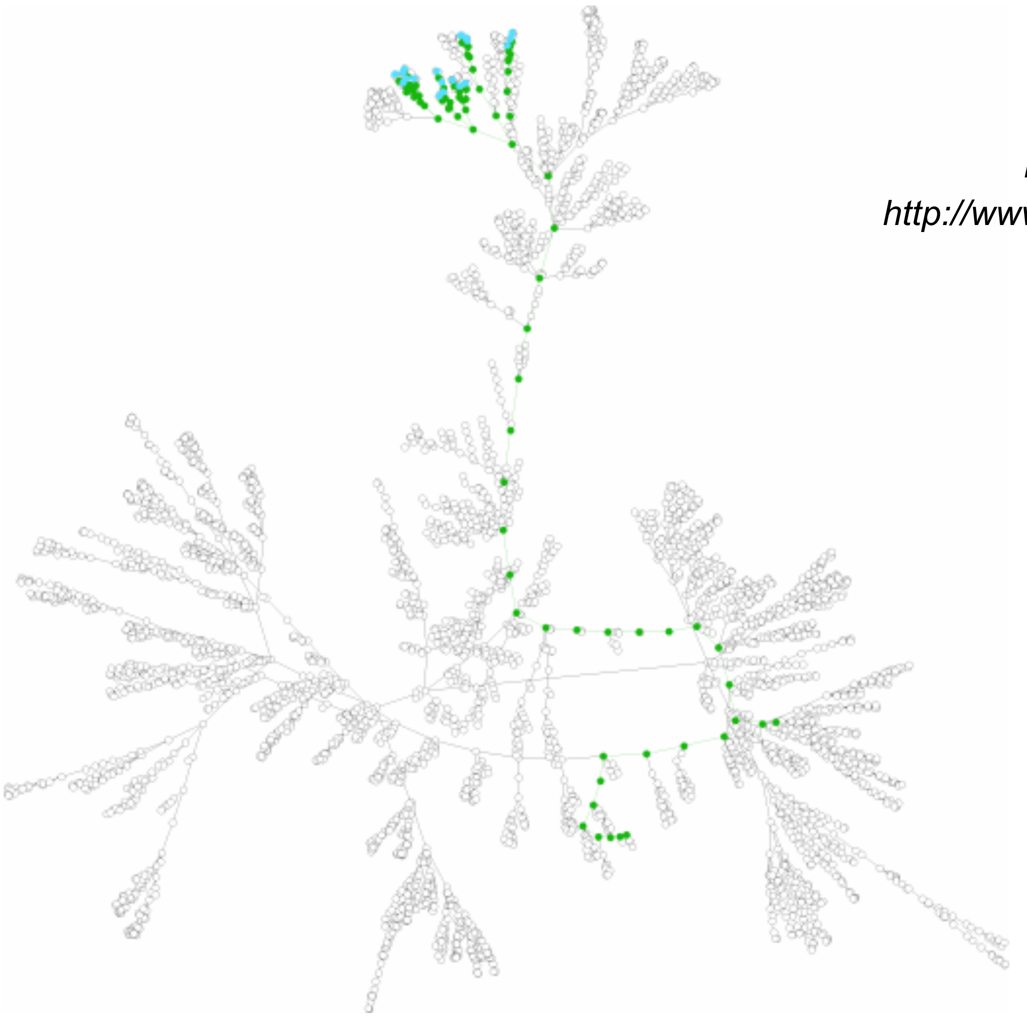
Shortest paths



solution graph of
rush hour puzzle
http://www.puzzles.com/products/rushhour.htm

Shortest paths

Single-source shortest paths (or single-destination)
depending on the assumptions:
Dijkstra's algorithm $O(|S|^2)$
or $O(|S| + |A| \cdot \log |S|)$
Bellman-Ford algorithm $O(|A| \cdot |S|)$

All-pairs shortest paths
Floyd-Warshall algorithm $O(|S|^3)$

Problem

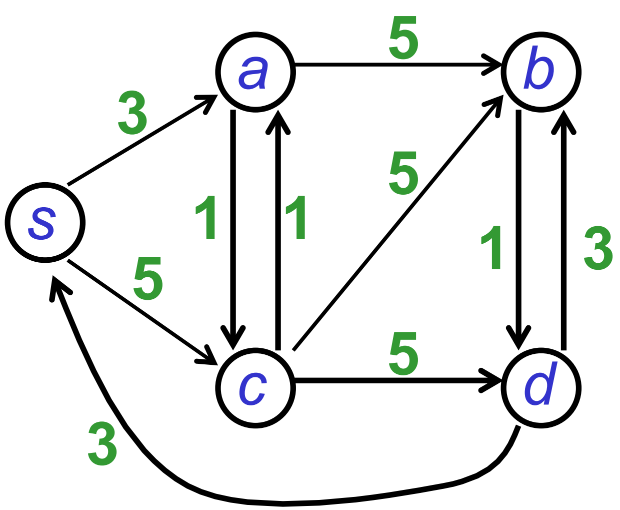
Weighted graph: $G = (S, A, v)$ where $v : A \to \mathbf{R}$ (weight/cost)

Source : $s \in S$

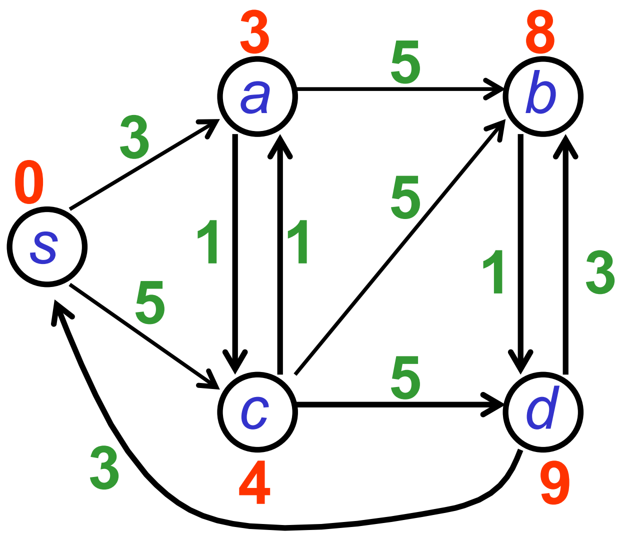
Problem: for all $t \in S$, compute

$$\delta(s, t) = \min \{ v(c) \; ; \; c \text{ path from } s \text{ to } t \} \cup \{+\infty\}$$

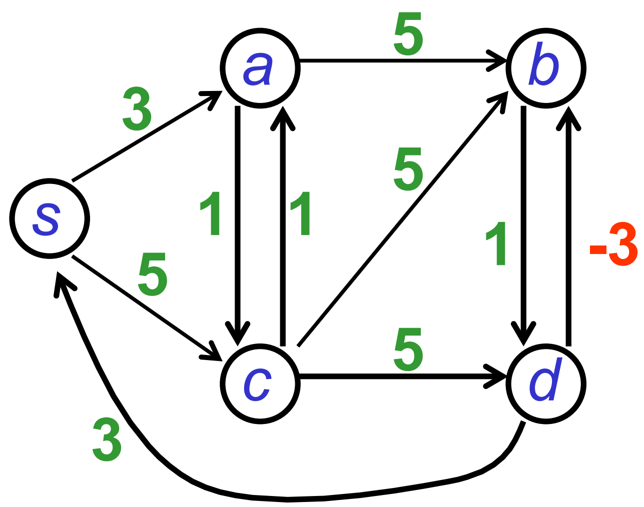
Example:



Costs δ:



Properties of shortest paths



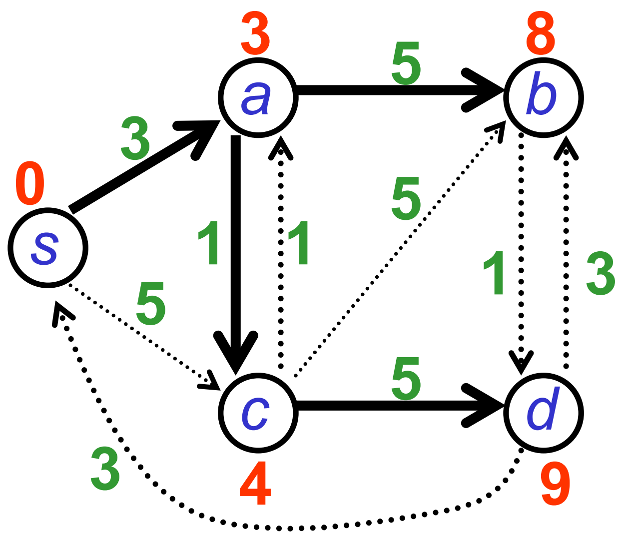
Proposition 1 (existence):
for all $t \in S$, $\delta(s, t) > -\infty$ **iff** the graph does not have a cycle of cost < 0 reachable from *s*

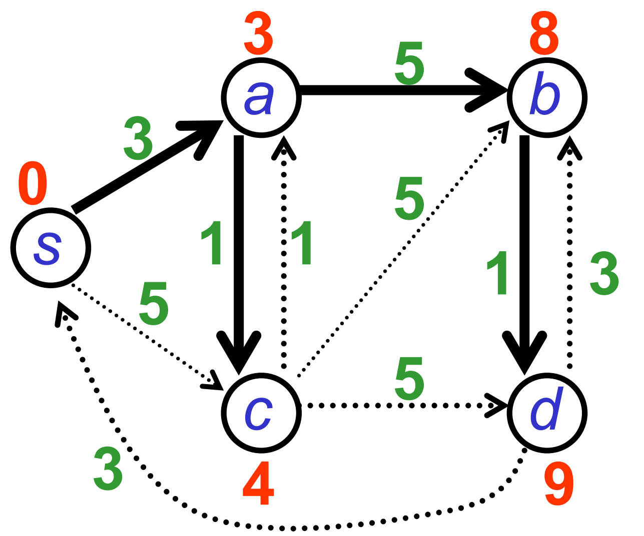
Proposition 2: a shortest path cannot contain cycles of positive or negative cost

Proposition 3: if there exists a shortest path from *s* to *t*, then there exists one with no more than |S|-1 edges

Trees of shortest paths

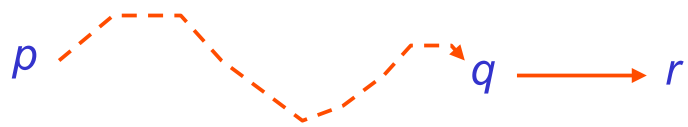
Trees rooted at *s* representing shortest paths





Main properties

Property 1: $G = (S, A, v)$
let c be a shortest path from p to r
and q be the node preceding r in c.
Then $\delta(p, r) = \delta(p, q) + v(q, r)$.



Property 2: A subpath of a shortest path is a shortest path

Property 3: $G = (S, A, v)$ let c be a path from p to r and q be the node preceding r in c. Then $\delta(p, r) \leq \delta(p, q) + v(q, r)$.

Relaxation

Compute $\delta(s,t)$ by successive approximations

$t \in S$ $d(t)$ = estimate (from above) of $\delta(s, t)$

$\pi(t)$ = predecessor of t on

a path from s to t of cost $d(t)$

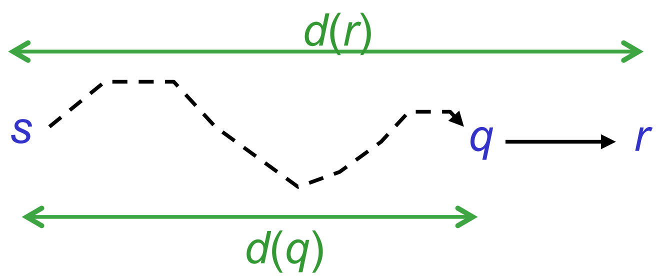
Initialization of d and π

INIT

for all $t \in S$ **do**

{ $d(t) \leftarrow \infty$; $\pi(t) \leftarrow$ **nil** ; }

$d(s) \leftarrow \mathbf{0}$;



Relaxation of the edge (q, r)

RELAX(q, r)

if $d(q) + v(q, r) < d(r)$

then { $d(r) \leftarrow d(q) + v(q, r)$; $\pi(r) \leftarrow q$; }

Relaxation (cont)

Proposition :
the property « for all $t \in S$, $d(t) \geq \delta(s, t)$ »
is an invariant of **`relax`**

Proof by induction on the number of executions of **`relax`**

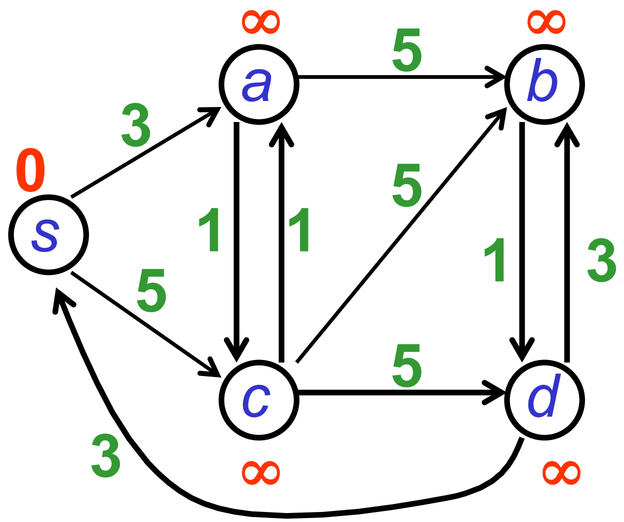
Dijkstra's algorithm

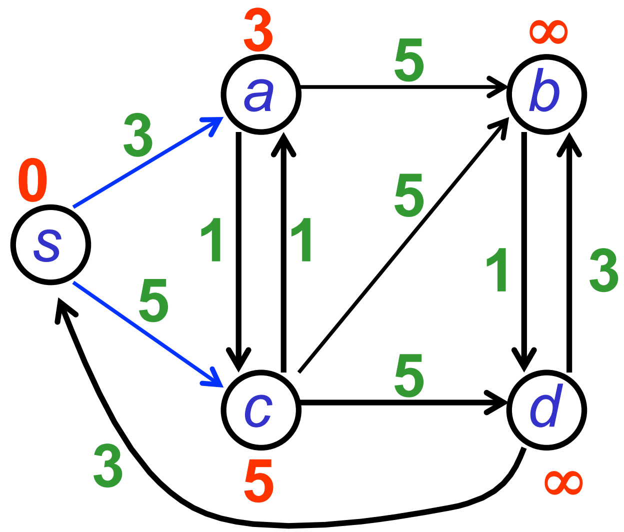
Assumption: $v(p, q) \geq 0$ for all edges (p, q)

```
begin
     INIT;
     Q ← S ;
     while Q ≠ ∅  do {
          q ← MIN_d(Q) ;  Q ← Q - {q} ;
          for all r successor of q  do
               RELAX(q, r) ;
     }
end
```

Greedy algorithm

Example

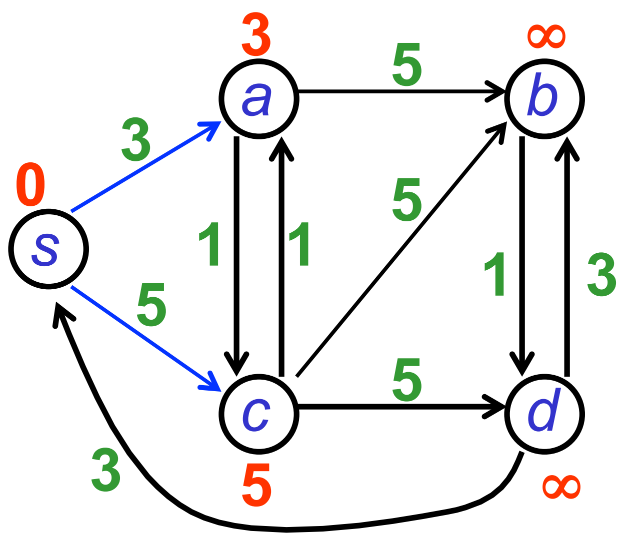


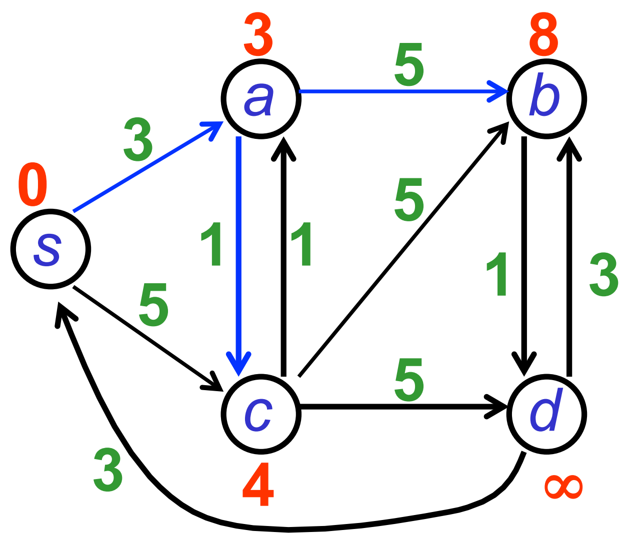


$Q = \{ s, a, b, c, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = \text{nil}$
$\pi(b) = \text{nil}$
$\pi(c) = \text{nil}$
$\pi(d) = \text{nil}$

$Q = \{ a, b, c, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = s$
$\pi(b) = \text{nil}$
$\pi(c) = s$
$\pi(d) = \text{nil}$

Example (cont)

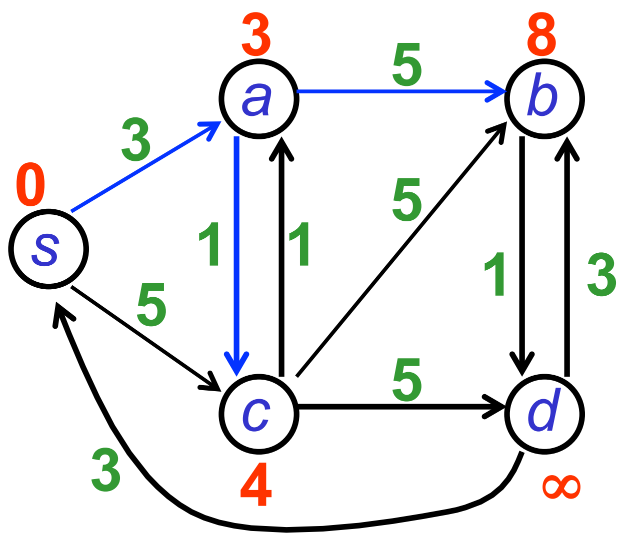




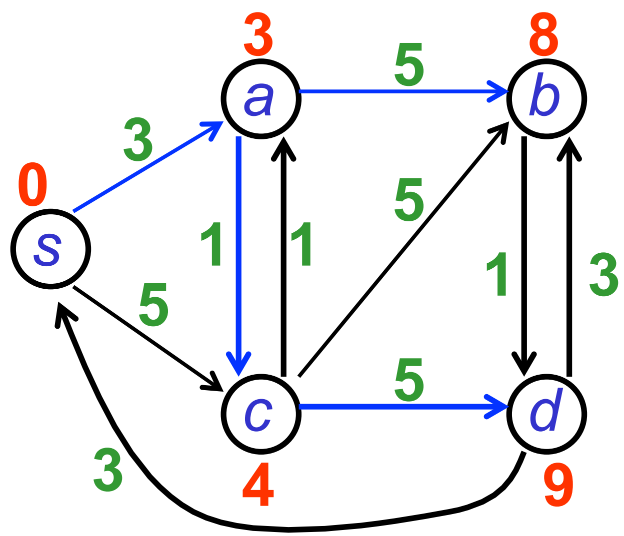
$Q = \{ a, b, c, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = s$
$\pi(b) = \text{nil}$
$\pi(c) = s$
$\pi(d) = \text{nil}$

$Q = \{ b, c, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = \text{nil}$

Example (cont)

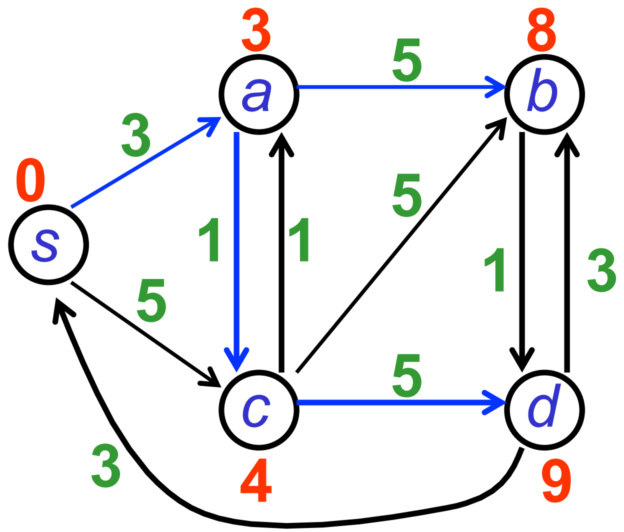


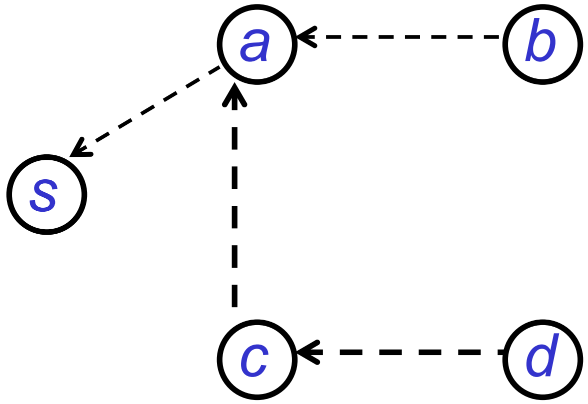
$Q = \{ b, c, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = \text{nil}$



$Q = \{ b, d \}$
$\pi(s) = \text{nil}$
$\pi(a) = s$
$\pi(b) = a$
$\pi(c) = a$
$\pi(d) = c$

Example (cont)





$Q = \{\ b,\ d\ \}$, $Q = \{\ d\ \}$ then $Q = \varnothing$

$\pi(s) = \text{nil}$

$\pi(a) = s$

$\pi(b) = a$

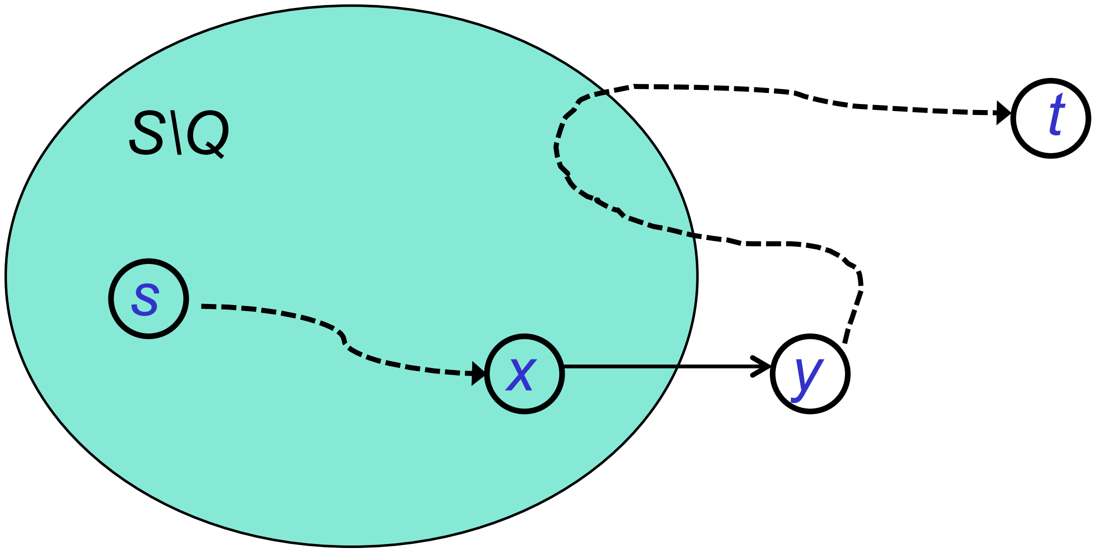
$\pi(c) = a$

$\pi(d) = c$

Correctness of Dijkstra's algorithm

Proposition : After the execution of Dijkstra's algorithm on a graph $G = (S, A, v)$, $d(t) = \delta(s, t)$ for all $t \in S$.

Proof by contradiction: let $d(t) \neq \delta(s, t)$



Implementation

With adjacency matrix

time $O(|S|^2)$

With adjacency lists

Q : priority queue

if implemented by binary heaps:

$|S|$ building a heap of $|S|$ elements: $O(|S|)$

$|S|$ operations **MIN**$_d$: $O(|S| \cdot \log |S|)$

$|A|$ operations **RELAX** : $O(|A| \cdot \log |S|)$

total time $O((|S|+|A|) \cdot \log |S|)$

time can be improved to $O(|S| \cdot \log |S|+|A|)$ using *Fibonacci heaps*

Breadth-first, Dijkstra, Best-first and A*

• **BFT** explores *the whole graph* and finds shortest paths to all nodes under assumption that all moves have equal cost. It uses a *queue*.

• **Dijkstra's alg** explores *the whole graph* and finds shortest paths to all nodes taking into account different move costs. It uses a *priority queue*.

• **Best-first** search finds a path to *a target node* by exploring the frontier nodes that are estimated to be closer to the target (*h(t)*)

• **A*** search finds a path to *a target node* by exploring the frontier nodes that have the maximal sum of distance from the source (*f(t)*) and estimated distance to the target (*h(t)*)

http://www.redblobgames.com/pathfinding/a-star/introduction.html

see *Pearl, J. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, 1984*

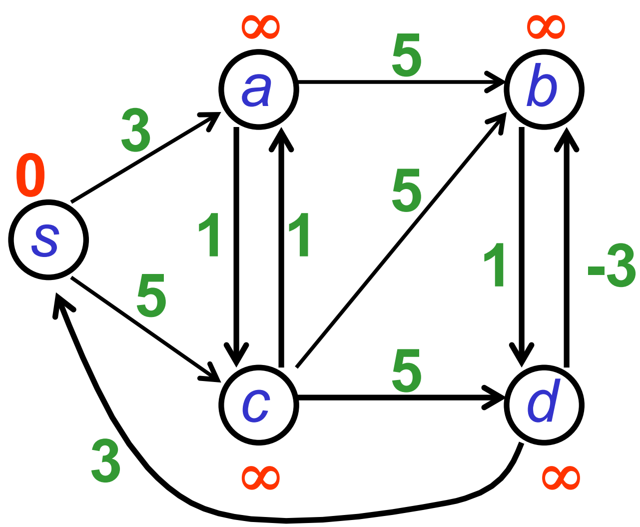
Bellman-Ford Algorithm

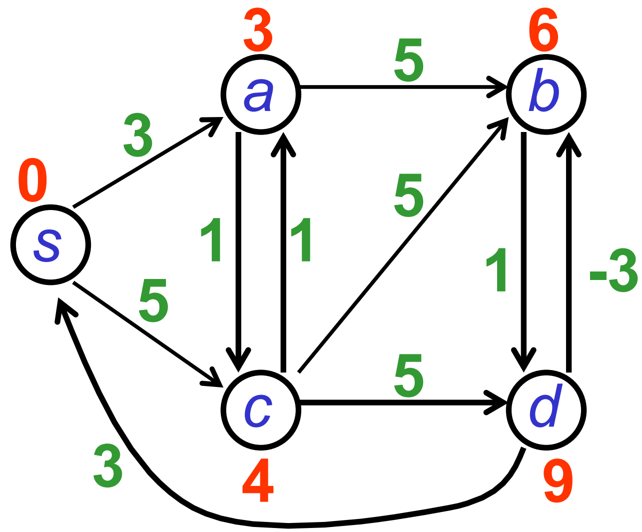
No condition on weights : for all edges (p, q), $v(p, q) \in \mathbf{R}$

```
begin
    INIT;
    Q ← S ;
    for i←1 to |S|-1 do
        for each (q, r) ∈ A do
            RELAX(q, r) ;
    for each (q, r) ∈ A do
        if d(q) + v(q, r) < d(r) then
            return « negative cost cycle detected »
        else
            retour « minimal costs computed »
end
```

Time complexity : $O(|S| \cdot |A|)$

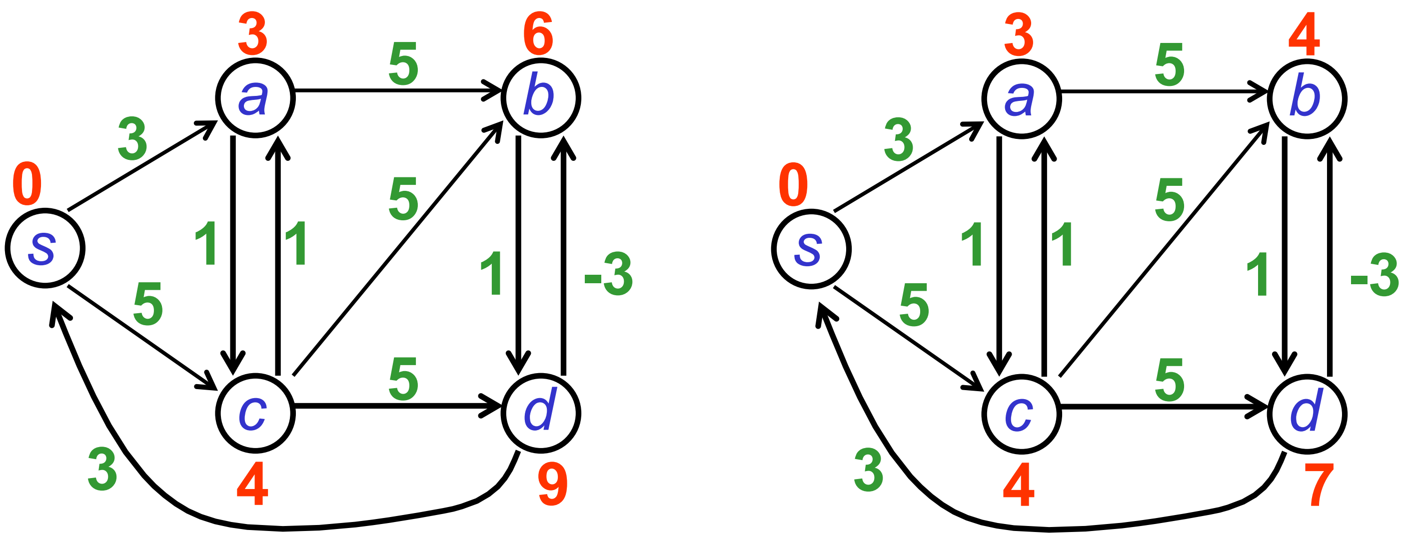
Example 1





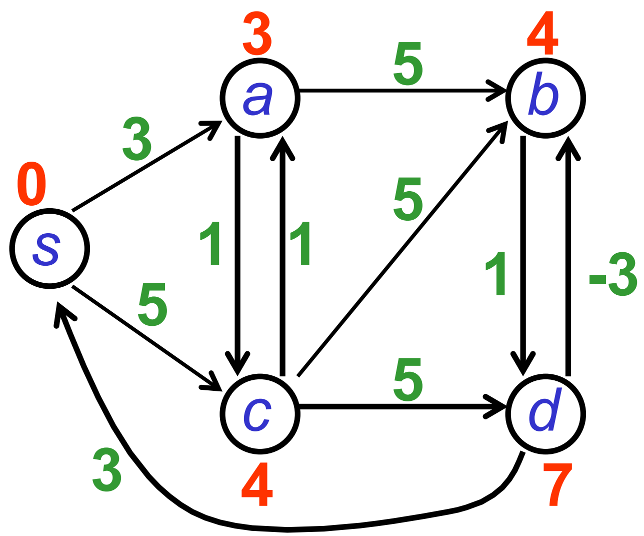
Step 1: relaxing all edges in the following order:
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

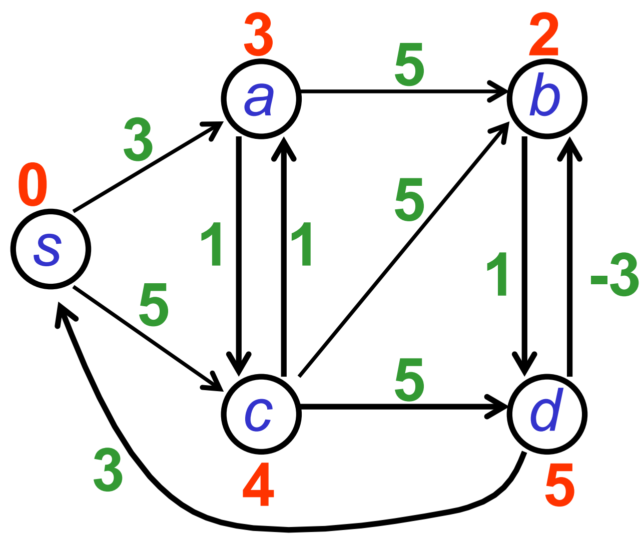
Example 1 (cont)



Step 2: relaxing all edges in the following order:
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

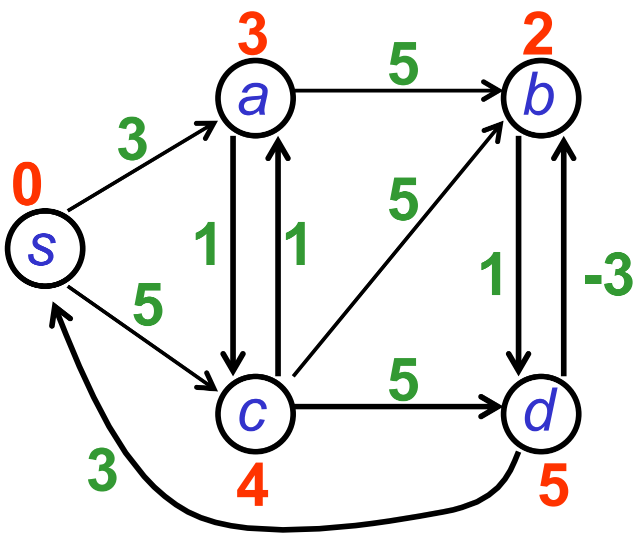
Example 1 (cont)

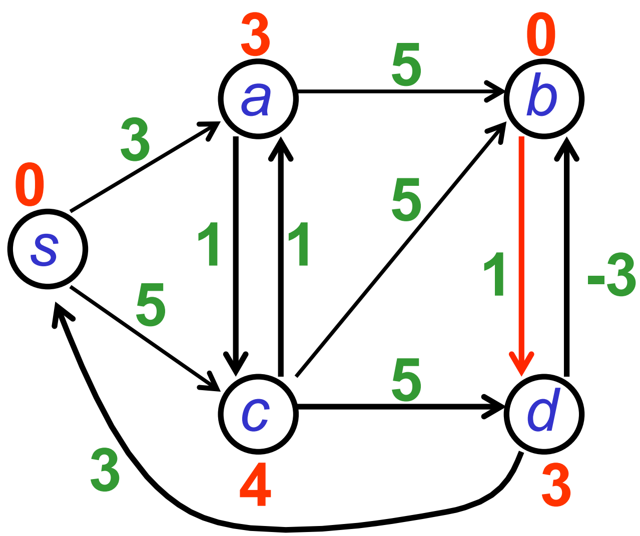




Step 3: relaxing all edges in the following order:
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Example 1 (cont)

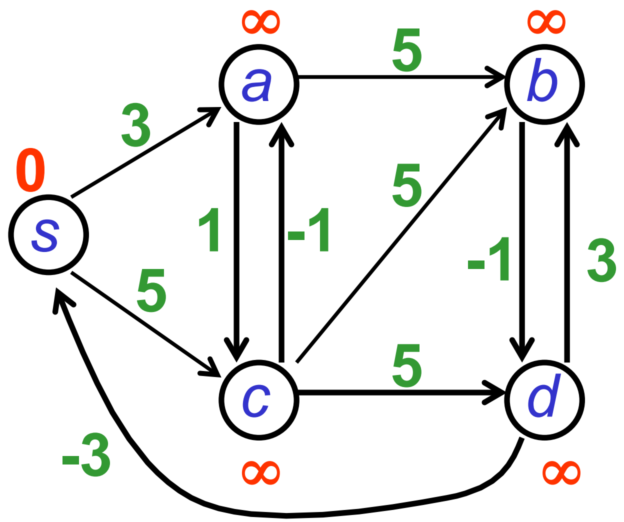


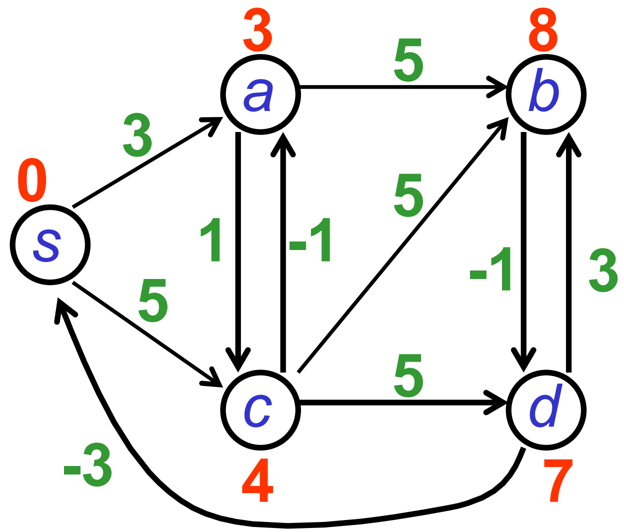


Step 4: relaxing all edges in the following order:
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

relaxation still possible ⇒ cycle of negative cost

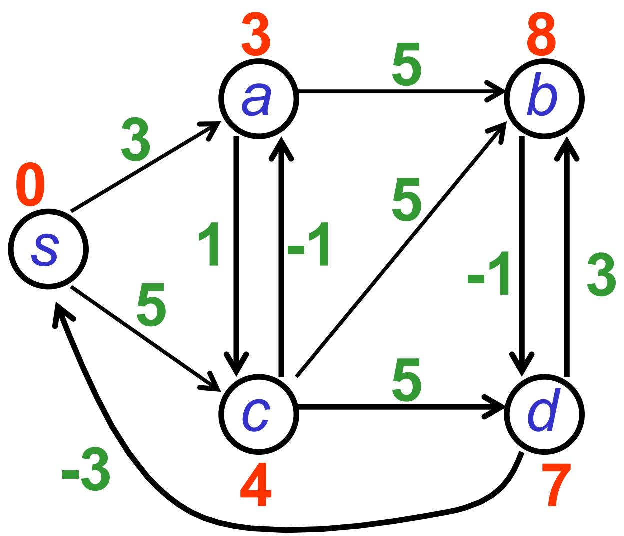
Example 2

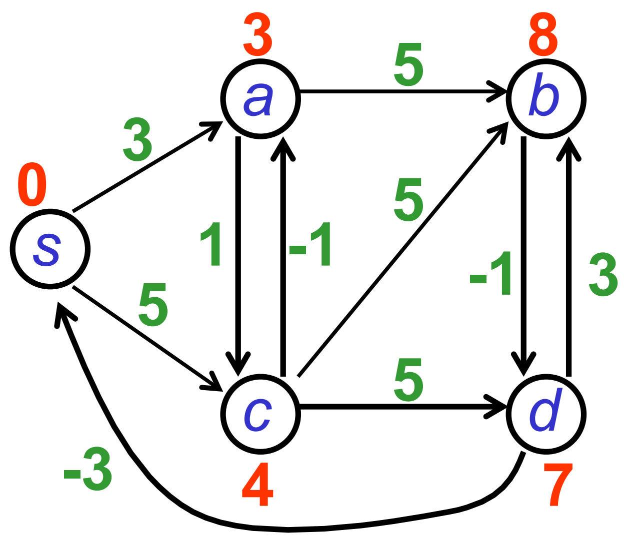




Step 1: relaxing all edges in the following order:
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Example 2 (cont)





Step 2: relaxing all edges in the following order:
(s,a) (s,c) (a,b) (a,c) (b,d) (c,a) (c,b) (c,d) (d,b) (d,s)

no more possible relaxation ⇒ costs correctly computed

Why Bellman-Ford algorithm is correct?

because, if there is no negative-cost cycle, every node has a cycle-free shortest path with at most $|S|$-1 edges

((s_0,s_1), (s_1,s_2), …, (s_{k-1},s_k)) with $s_0=s$ and s_k=t, $k\leq|S|-1$

At iteration i, we will relax (among other edges) (s_{i-1},s_i). This guarantees the shortest path value for all nodes. No relaxation will be possible anymore.

If there exists a negative-cost cycle, one of the edge along the cycle must be possible to relax (prove).

Shortest paths in Directed Acyclic Graphs

No condition on edge cost (as DAG has no cycle)

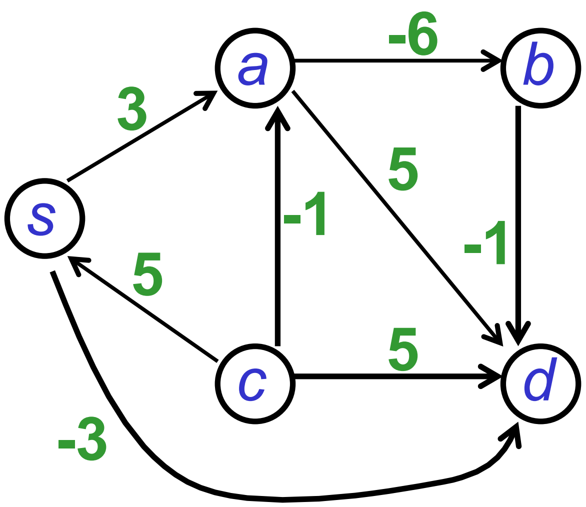
Idea: perform topological sort and process nodes in order

```
begin
      INIT;
      for all q ∈ S in topological order do
            for all r successor of q do
                  RELAX(q, r) ;
end
```

Time: $O(|S| + |A|)$

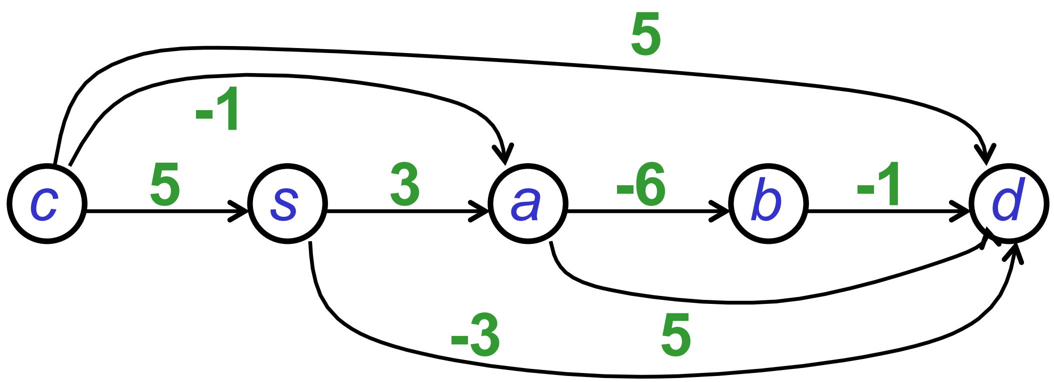
Why the algorithm is correct?

Example

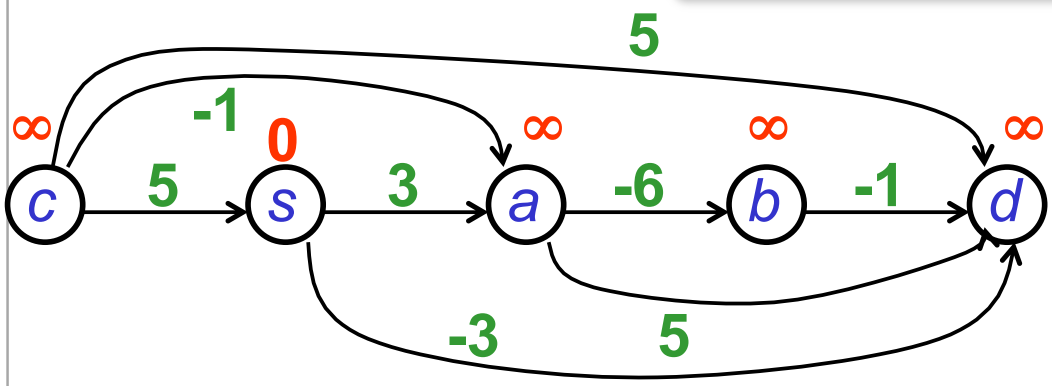


Topological order

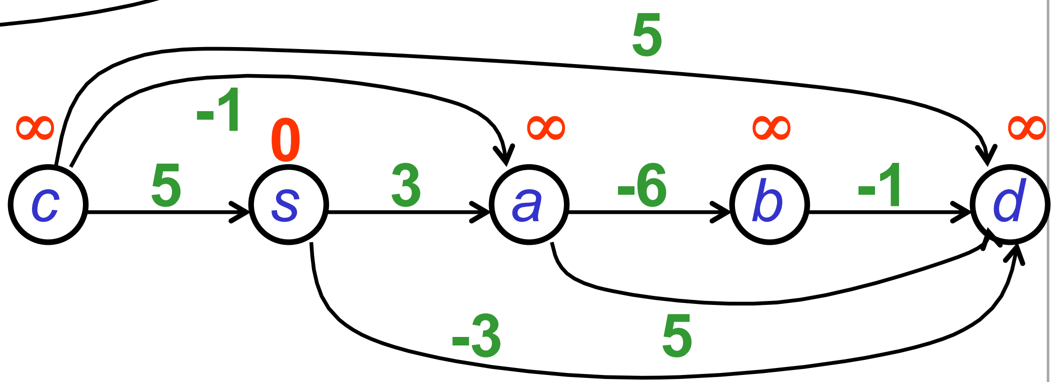
c, *s*, *a*, *b*, *d*



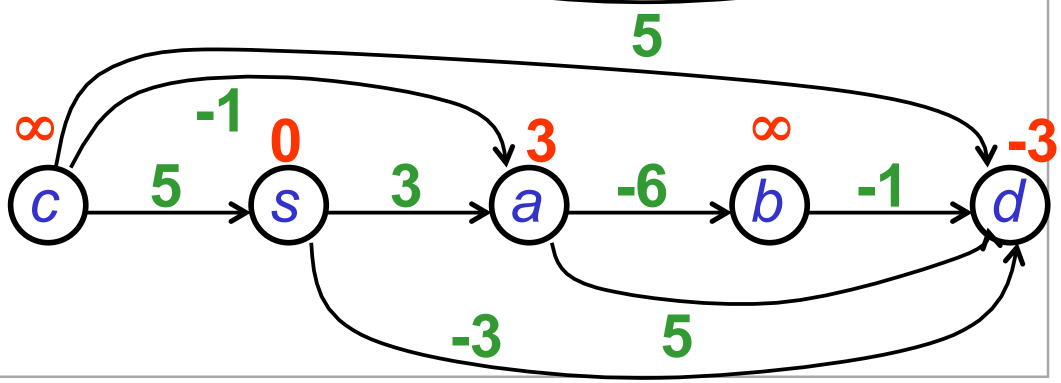
Computing costs



Processing *c*

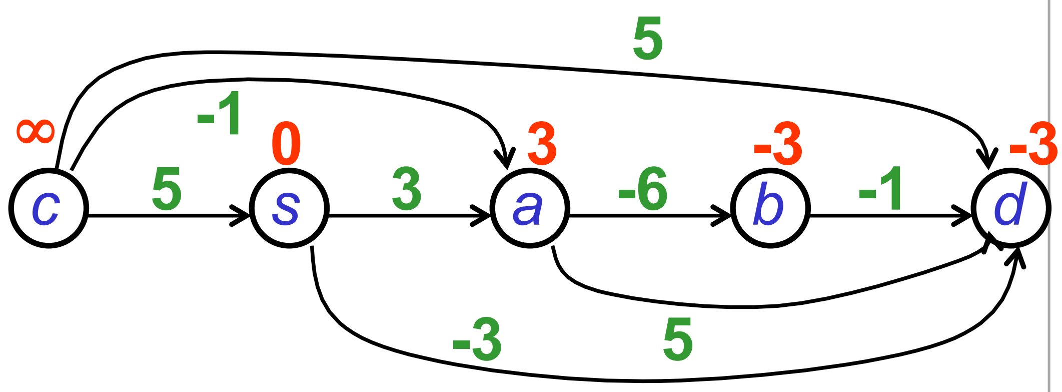


Processing *s*



Computing costs

Processing *a*



Processing *b*