Chapter 24: Single-Source Shortest-Path

About this lecture

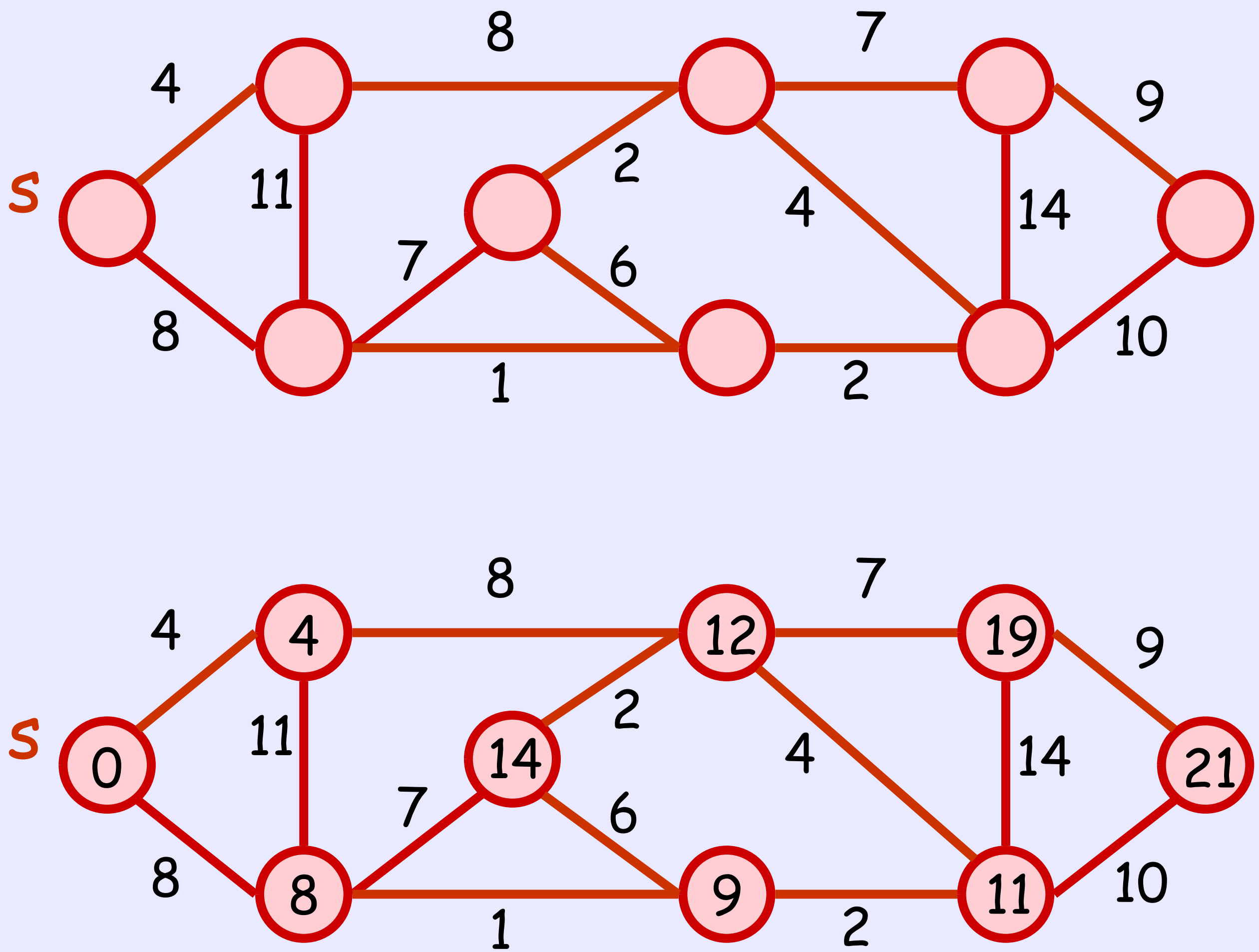
- What is the problem about ?
- Dijkstra's Algorithm [1959]
  - ~ Prim's Algorithm [1957]
- Folklore Algorithm for DAG
- Bellman-Ford Algorithm
  - Discovered by Bellman [1958], Ford [1962]
  - Allowing negative edge weights

Single-Source Shortest Path

- Let $G = (V,E)$ be a weighted graph
  - ✓ the edges in G have positive weights
  - ✓ can be directed/undirected
  - ✓ can be connected/disconnected
- Let s be a special vertex, called source
- Target: For each vertex v, compute the length of the shortest path from s to v

Single-Source Shortest Path

- E.g.,



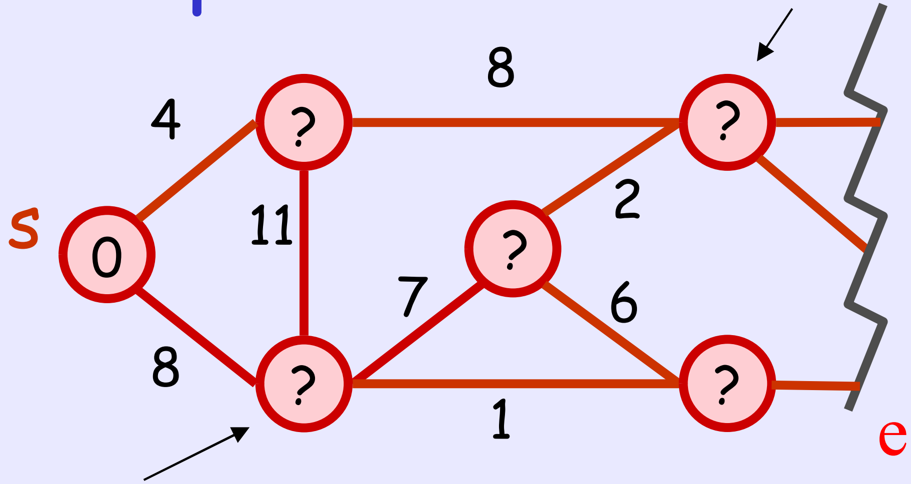
Relax

- A common operation that is used in the algorithms is called Relax :

  when a vertex v can be reached from the source with a certain distance, we examine an outgoing edge, say (u, v), and check if we can improve v

- E.g.,

Can we improve this?



Can we improve this?

Relax (u, v, w)

If $d(v) > d(u) + w(u, v)$

$d(v) = d(u) + w(u, v)$

$d(v)$ is a shortest-path estimate from source s to v.

Dijkstra's Algorithm

Dijkstra(G, s)

For each vertex v,

Mark v as unvisited, and set $d(v) = \infty$;

Set $d(s) = 0$;

while (there is unvisited vertex) {

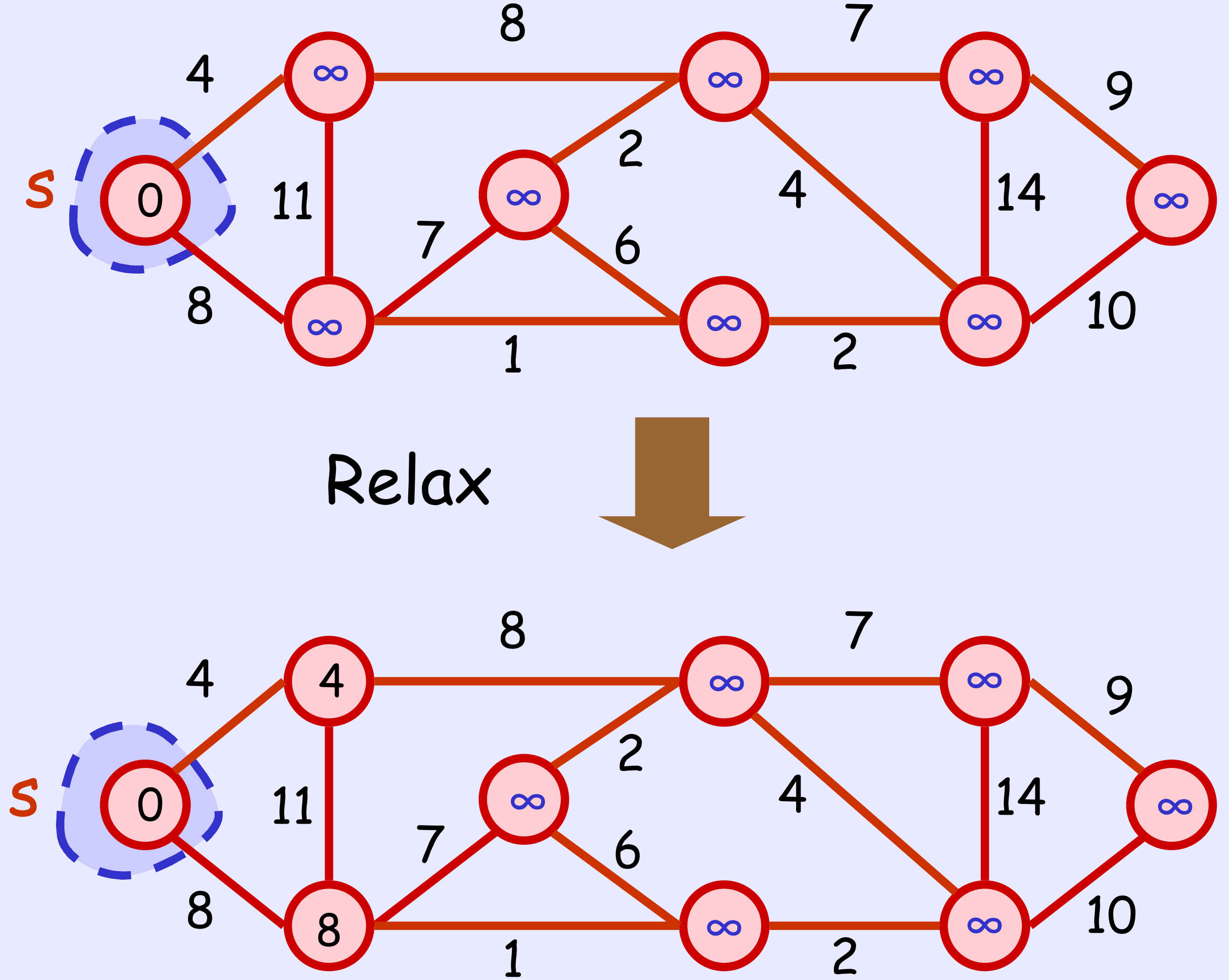
v = unvisited vertex with smallest d(v) ;

Visit v, and Relax all its outgoing edges;

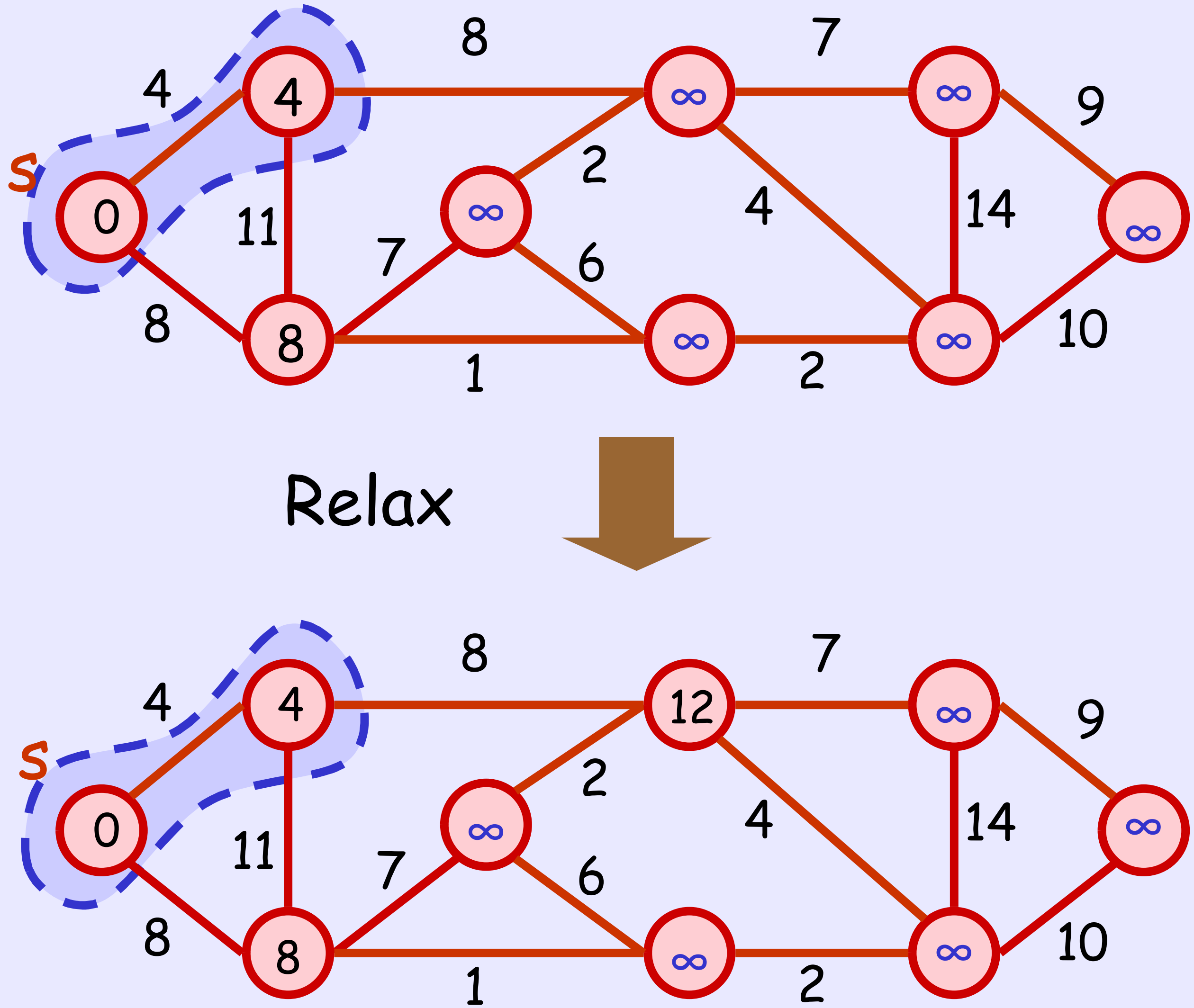
}

Return d;

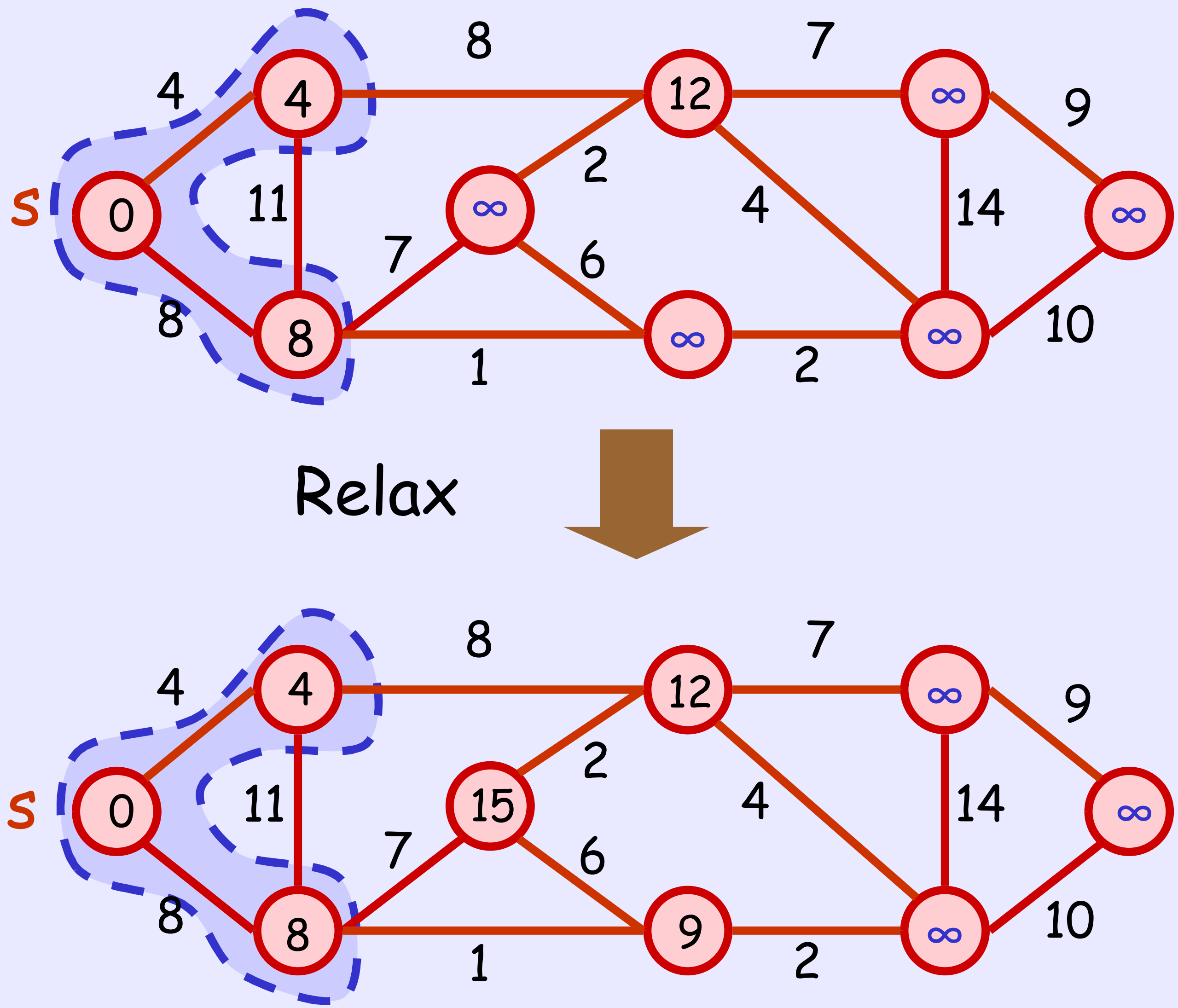
Example



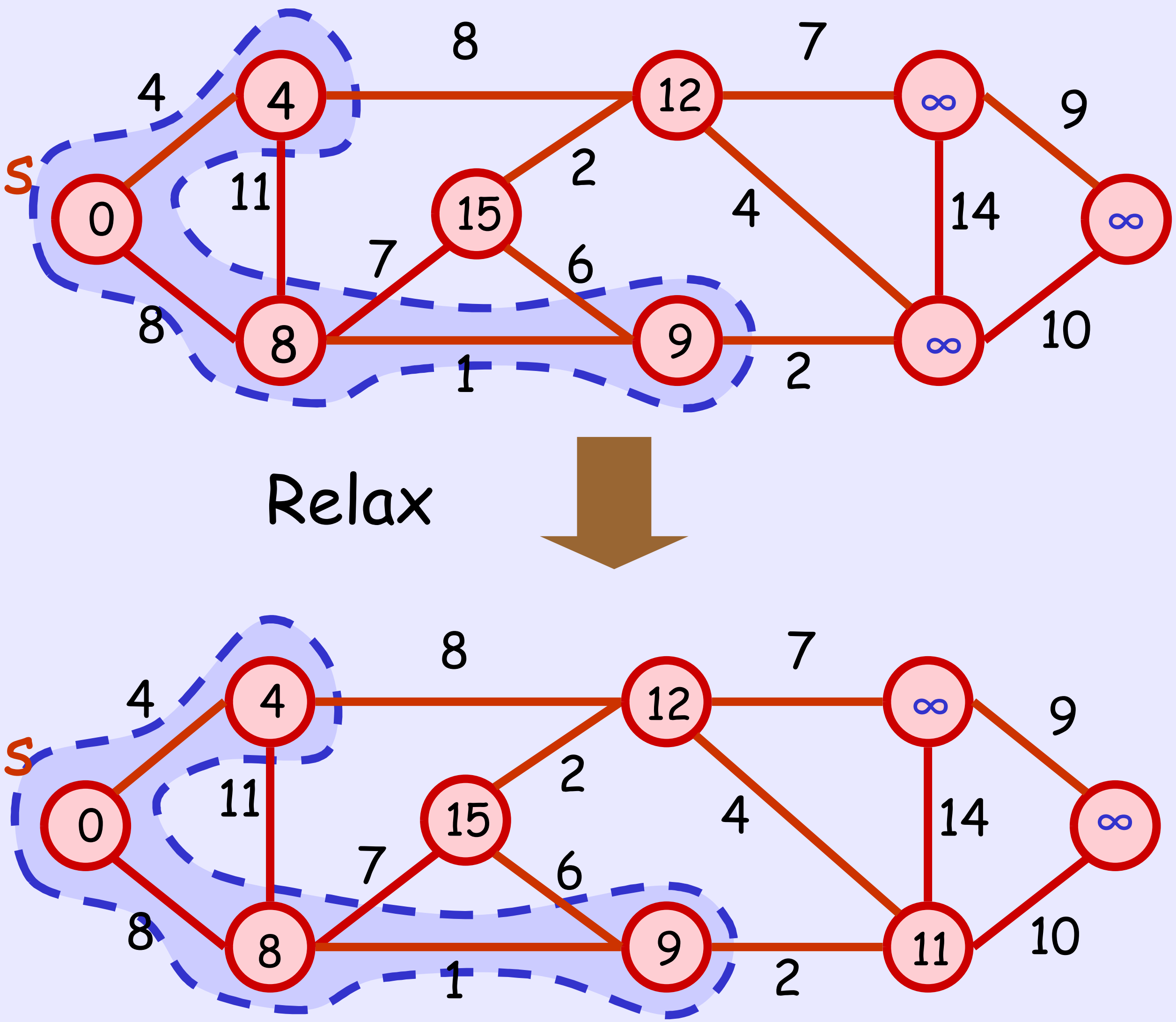
Example



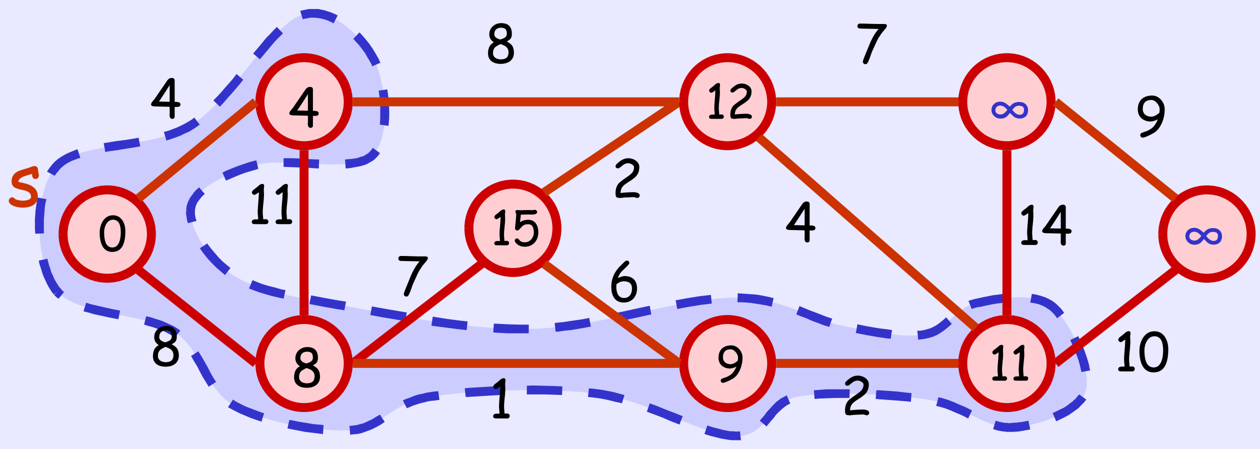
Example



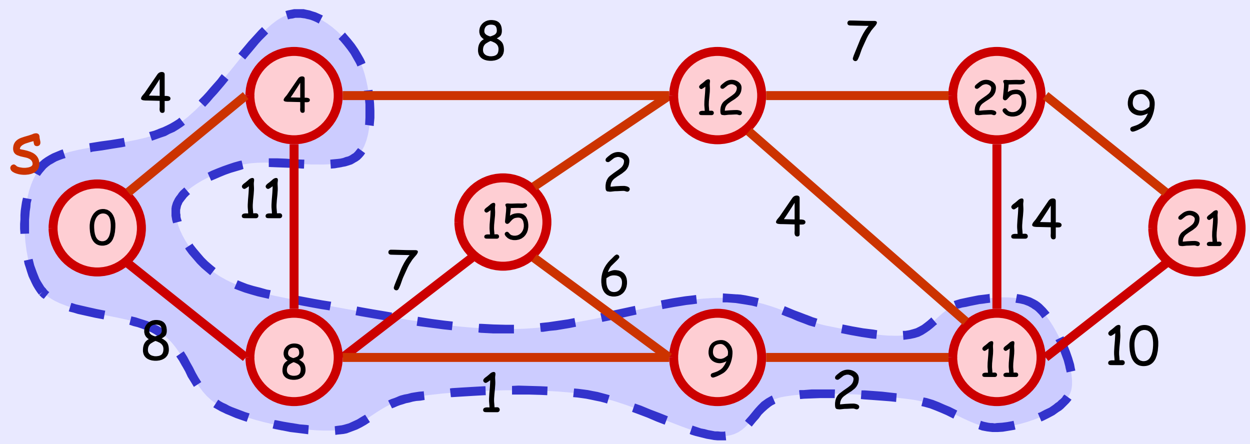
Example



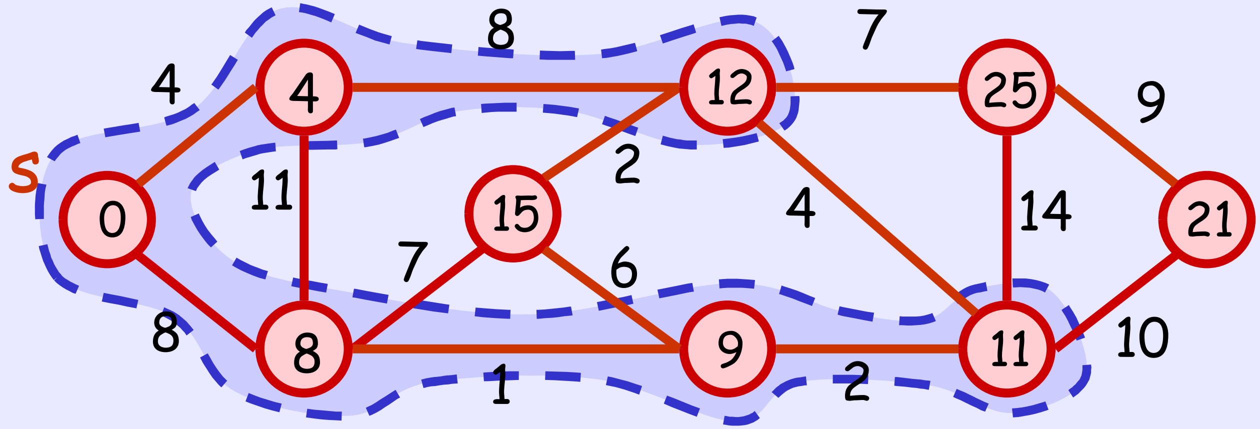
Example



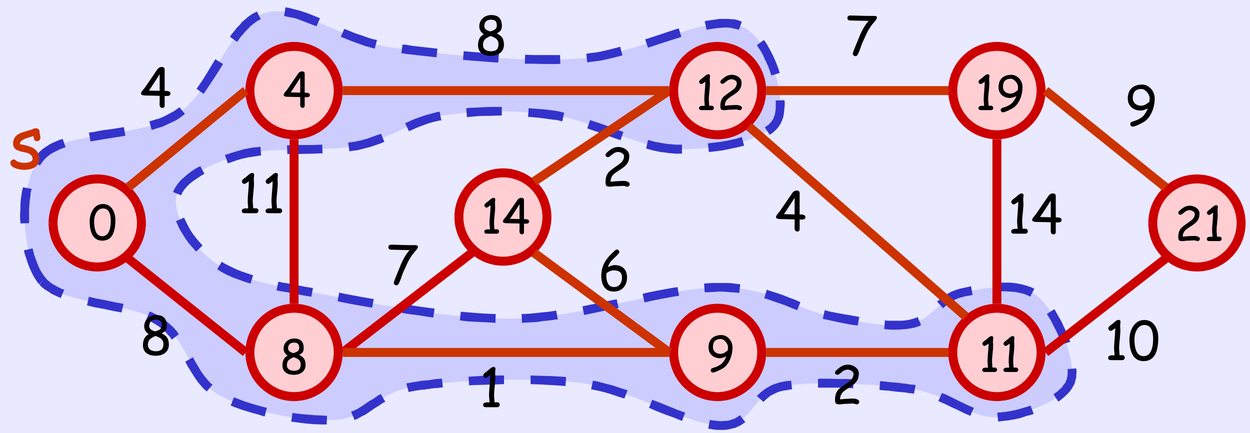
Relax



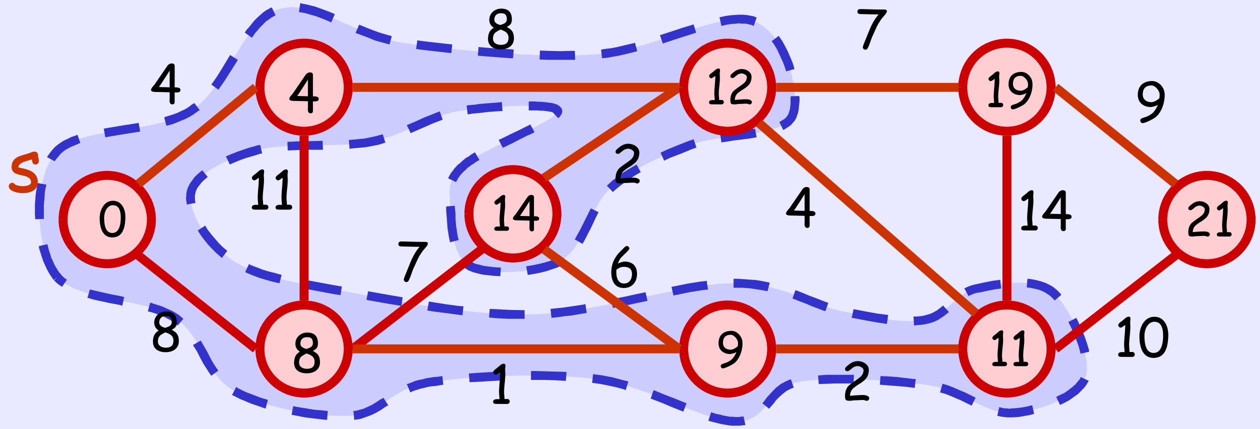
Example



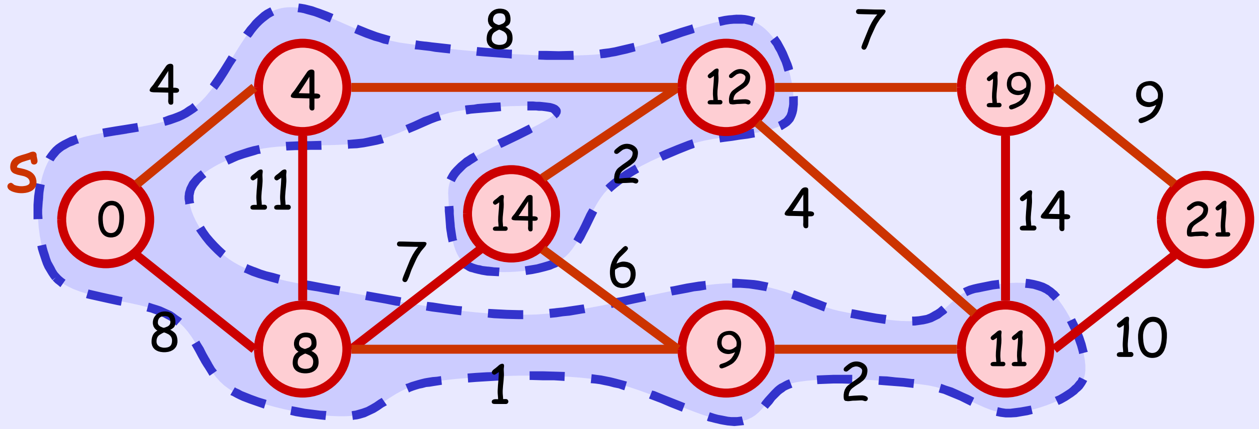
Relax



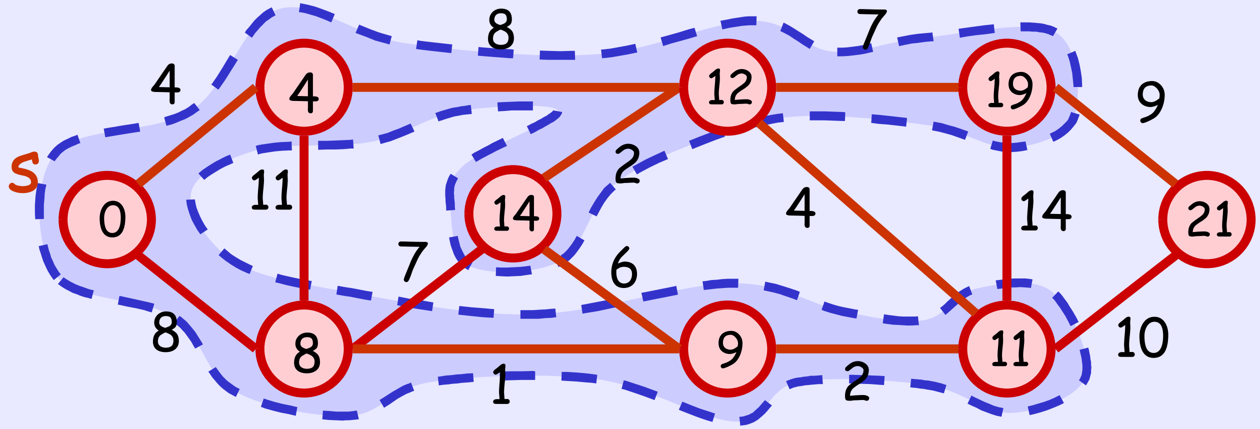
Example



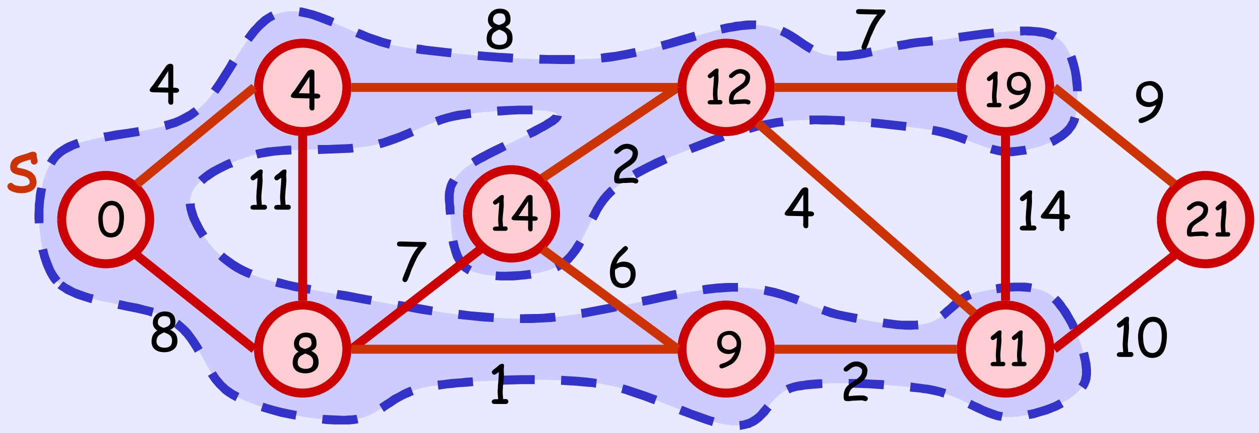
Relax



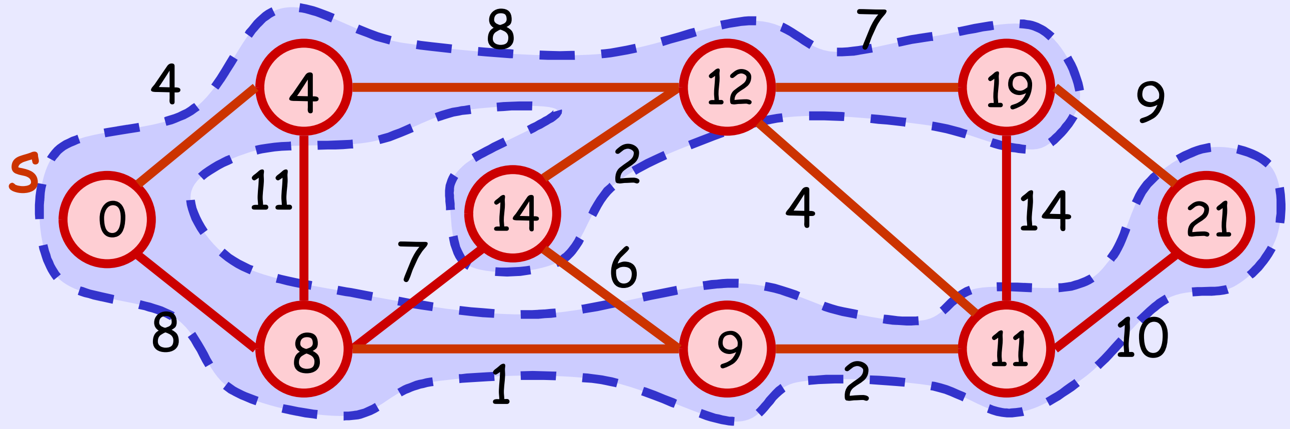
Example



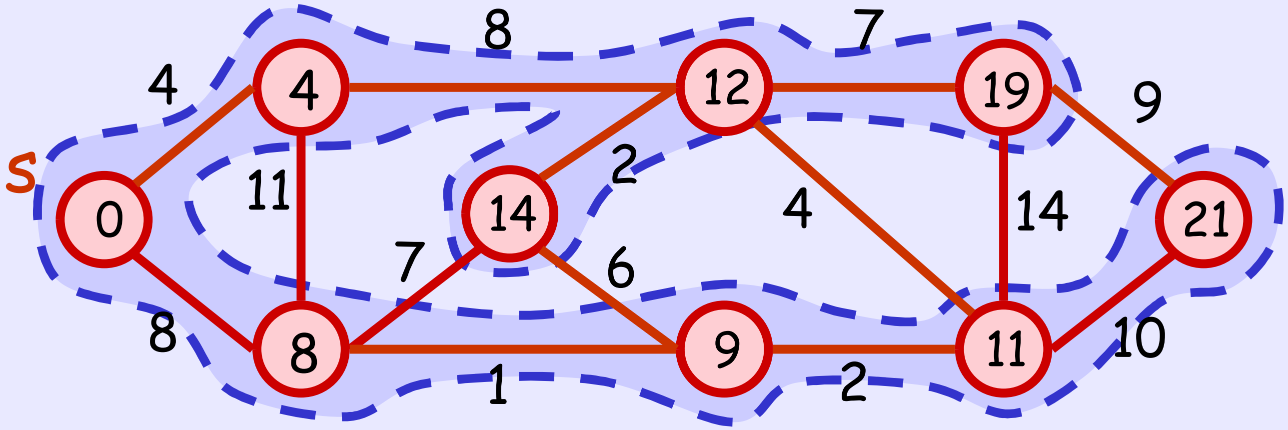
Relax



Example



Relax



Correctness

- Theorem:

  (i) The k^{th} vertex closest to the source s is selected at the k^{th} step inside the while loop of Dijkstra's algorithm

  (ii) Also, by the time a vertex v is selected, d(v) will store the length of the shortest path from s to v

- How to prove ? (By induction)

Proof

- Both statements are true for $k = 1$;
- Let v_j = j^{th} closest vertex from s
- Now, suppose both statements are true for $k = 1, 2, \ldots, r-1$
- Consider the r^{th} closest vertex v_r
  - If there is no path from s to v_r

    → $d(v_r) = \infty$ is never changed
  - Else, there must be a shortest path from s to v_r ; Let v_t be the vertex immediately before v_r in this path

Proof (cont)

- Then, we have $t \leq r-1$ (why??)

→ $d(v_r)$ is set correctly once v_t is selected, and the edge (v_t, v_r) is relaxed (why??)

(ii) → After that, $d(v_r)$ is fixed (why??)

(i) → $d(v_r)$ is correct when v_r is selected ; also, v_r must be selected at the r^{th} step, because no unvisited nodes can have a smaller d value at that time

Thus, the proof of inductive case completes

Performance

- Dijkstra's algorithm is similar to Prim's
- By simply store d(v) in the vth array.
  - Relax (Decrease-Key): $O(1)$
  - Pick vertex (Extract-Min): $O(V)$
- Running Time:
  - the cost of $|V|$ operation Extract-Min is $O(V^2)$
  - At most $O(E)$ Decrease-Key

  ➔ Total Time: $O(E + V^2) = O(V^2)$

Performance

- By using binary Heap (Chapter 6),
  - Relax ⇔ Decrease-Key: $O(\log V)$
  - Pick vertex ⇔ Extract-Min: $O(\log V)$
- Running Time:
  - the cost of each |V| operation Extract-Min is $O(V \log V)$
  - At most $O(E)$ Decrease-Key

  ➔ Total Time: $O((E + V) \log V)$

  $= O(E \log V)$

Performance

- By using Fibonacci Heap (Chapter 19),
  - Relax ⇔ Decrease-Key
  - Pick vertex ⇔ Extract-Min
- Running Time:
  - At most $O(E)$ Decrease-Key
  - the amortized cost of each $|V|$ operation Extract-Min is $O(\log V)$

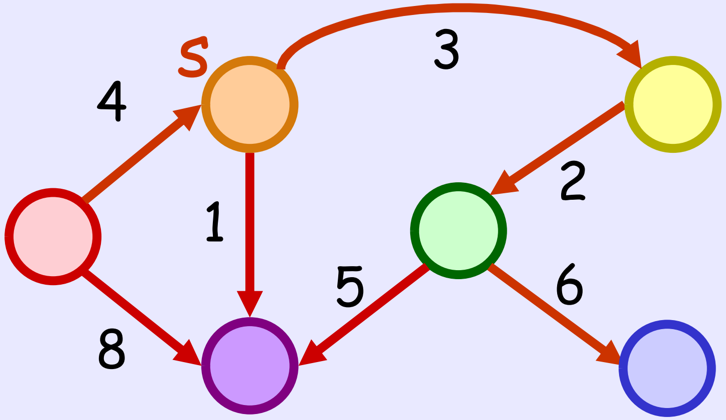
    ➔ Total Time: $O(E + V \log V)$

Finding Shortest Path in DAG

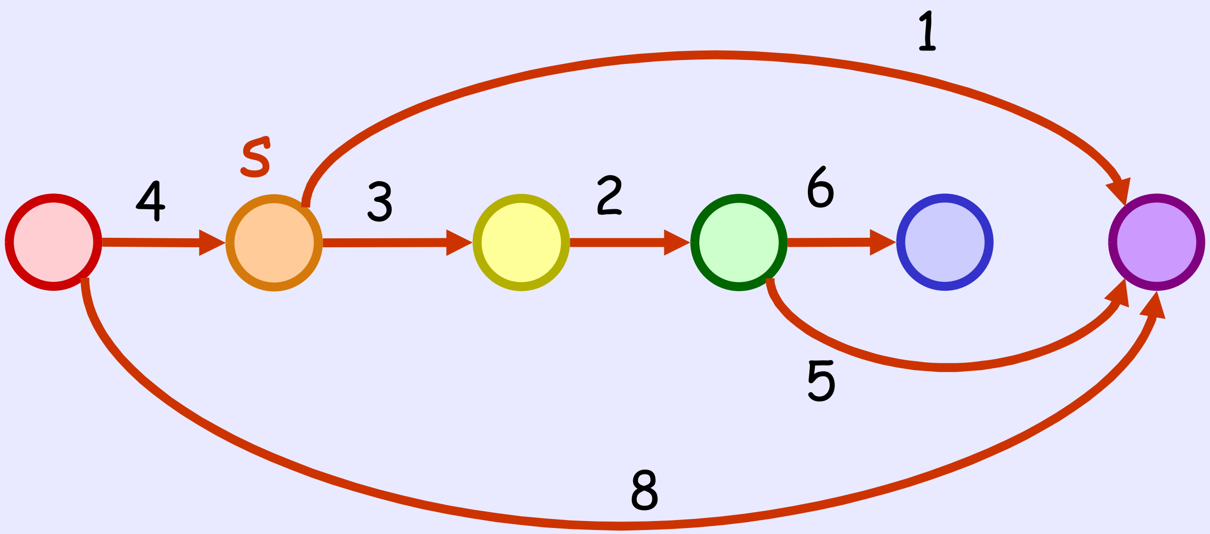
We have a faster algorithm for DAG :

```
DAG-Shortest-Path(G, s)
    Topological Sort G ;
    For each v, set d(v) = ∞ ; Set d(s) = 0 ;
    for (k = 1 to |V|) {
        v = kth vertex in topological order ;
        Relax all outgoing edges of v ;
    }
    return d ;
```

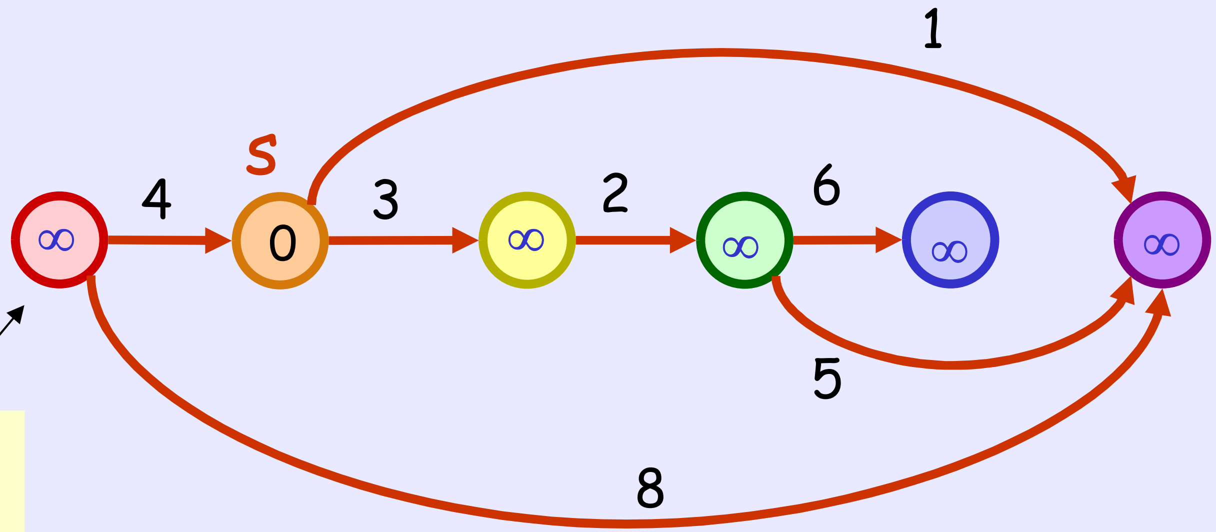
Example



Topological Sort

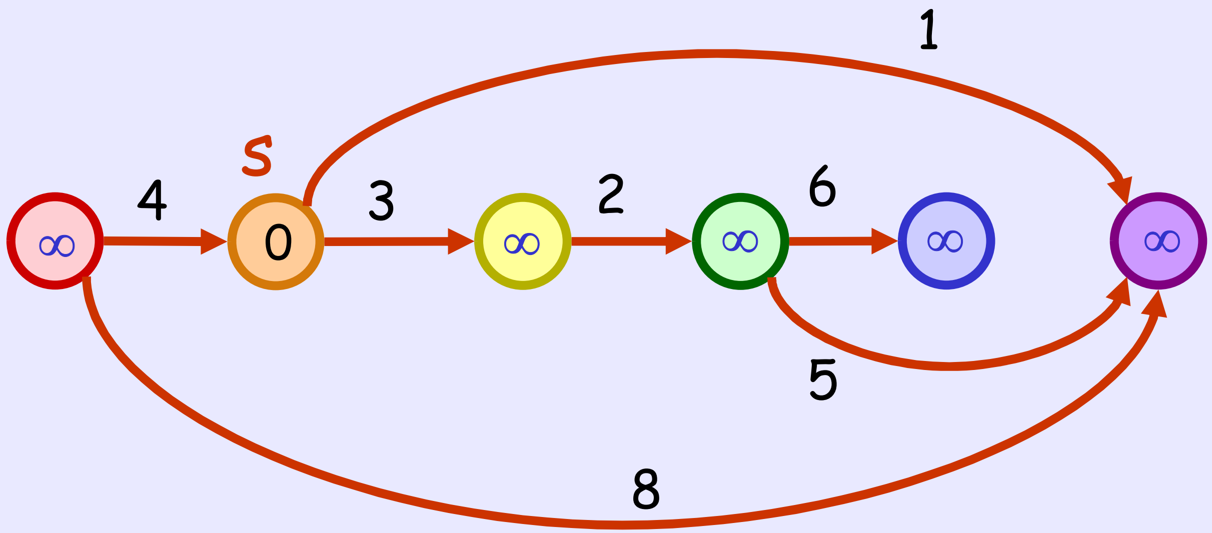


Example

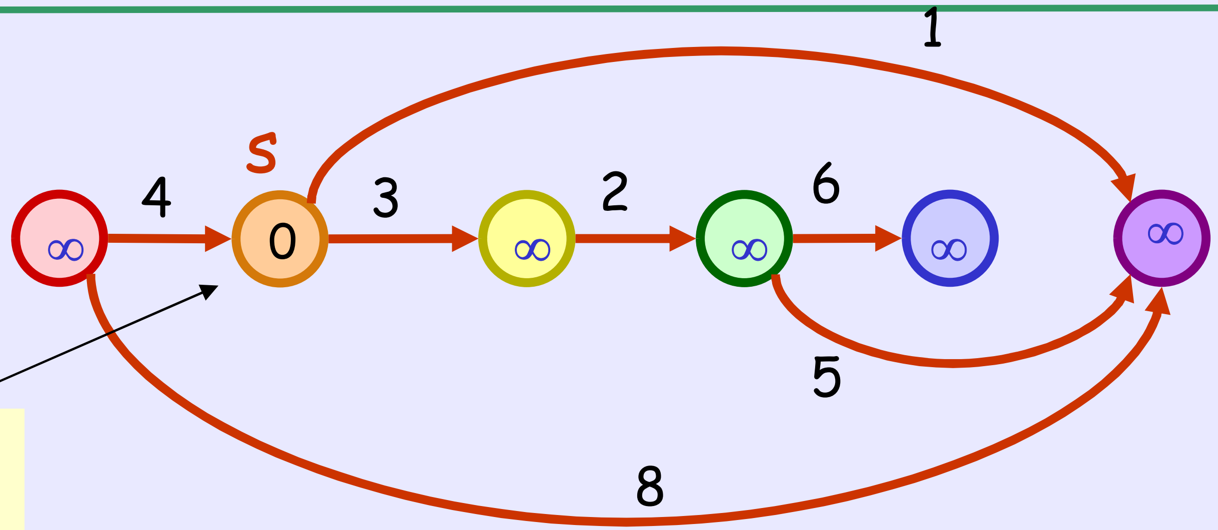


Process this node

Relax

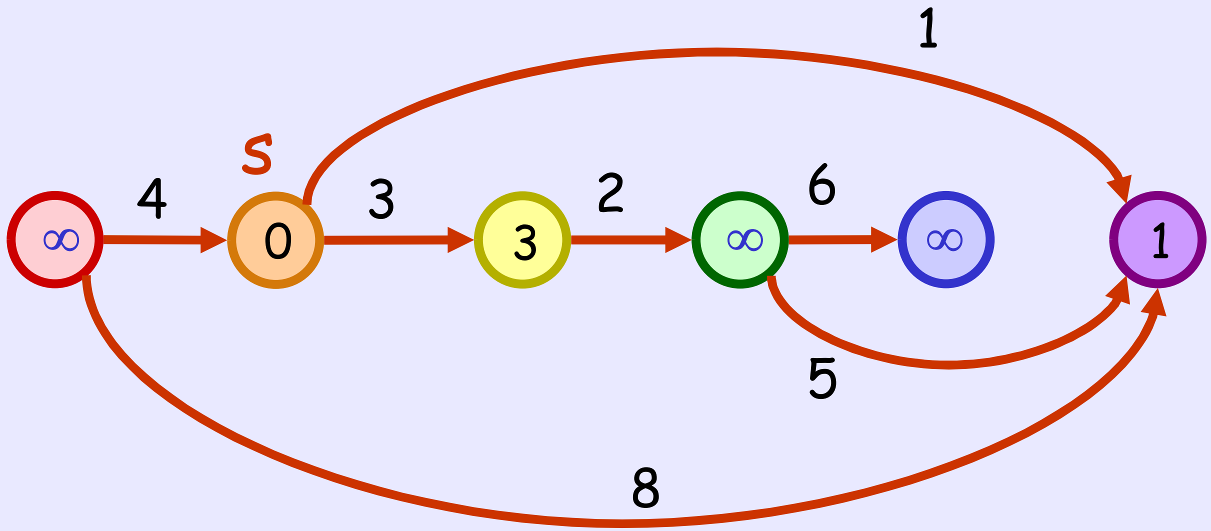


Example

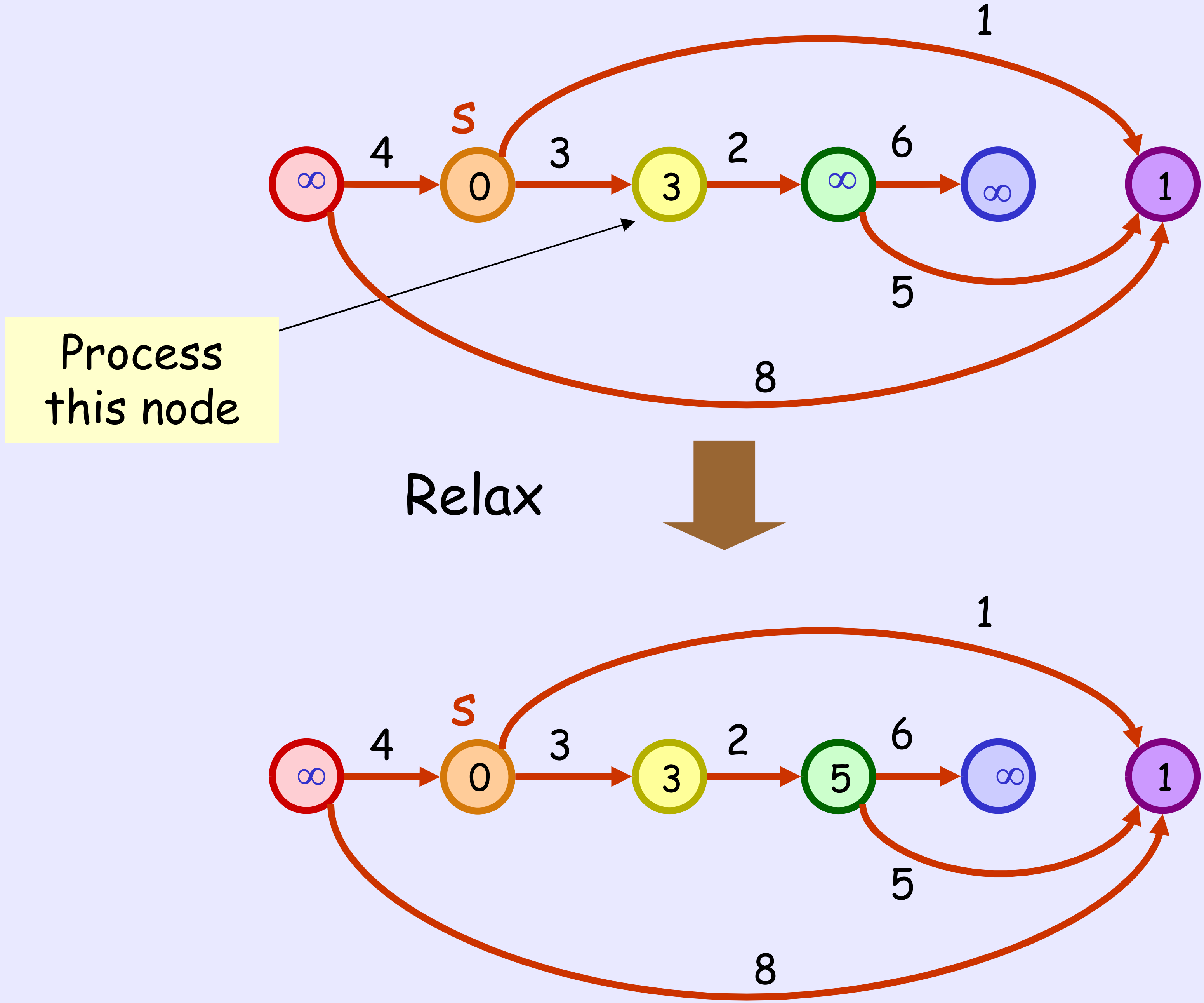


Process this node

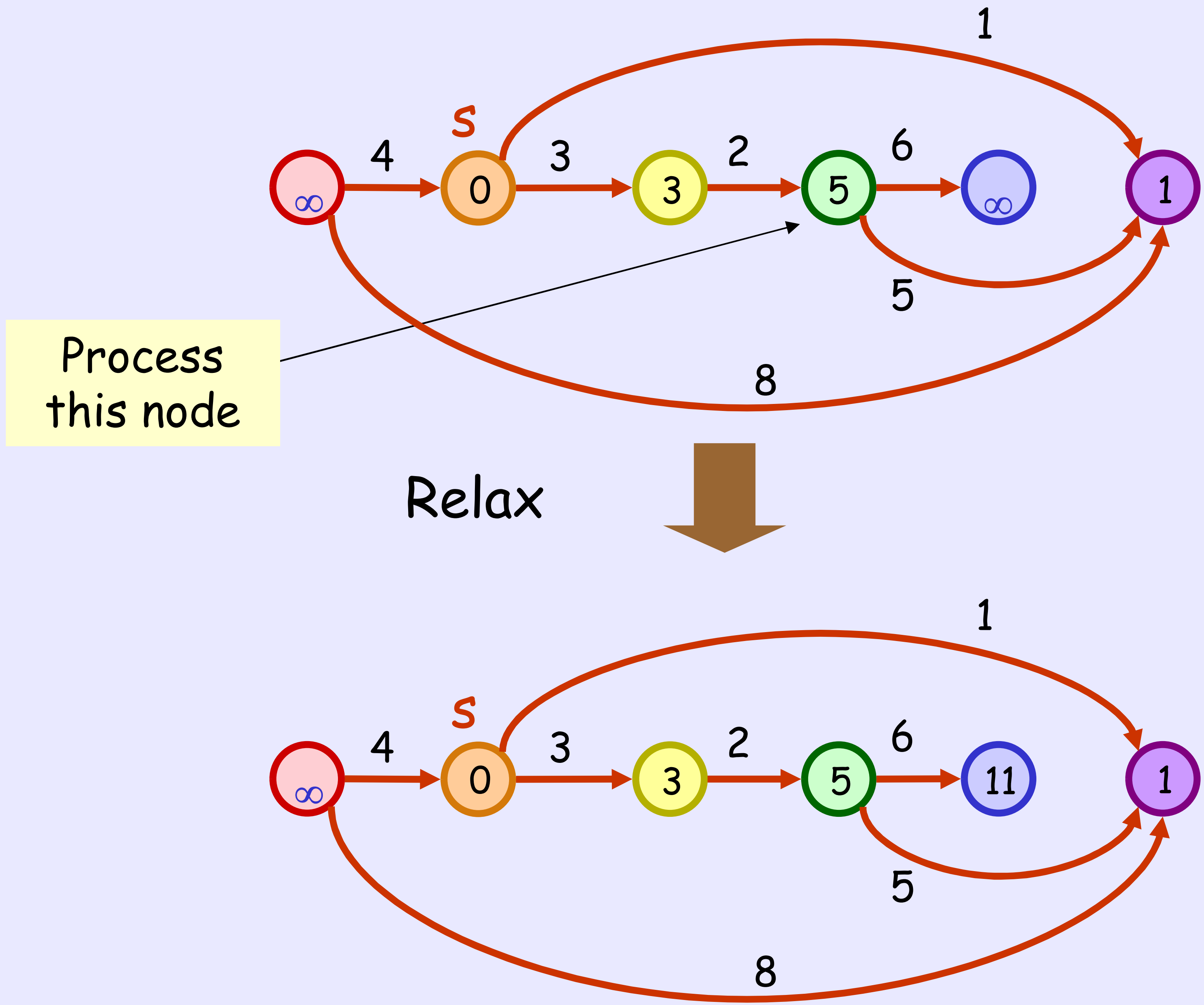
Relax



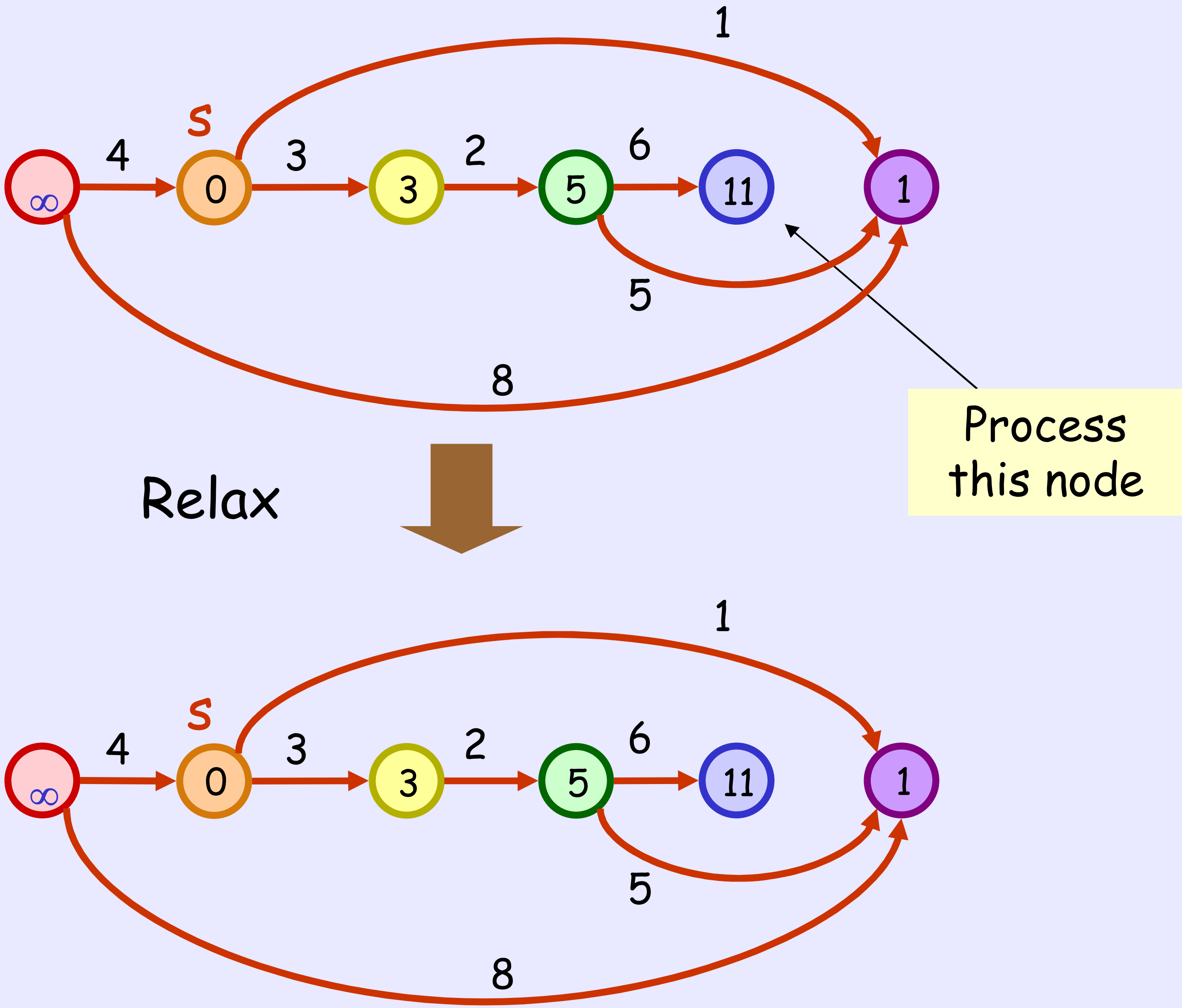
Example



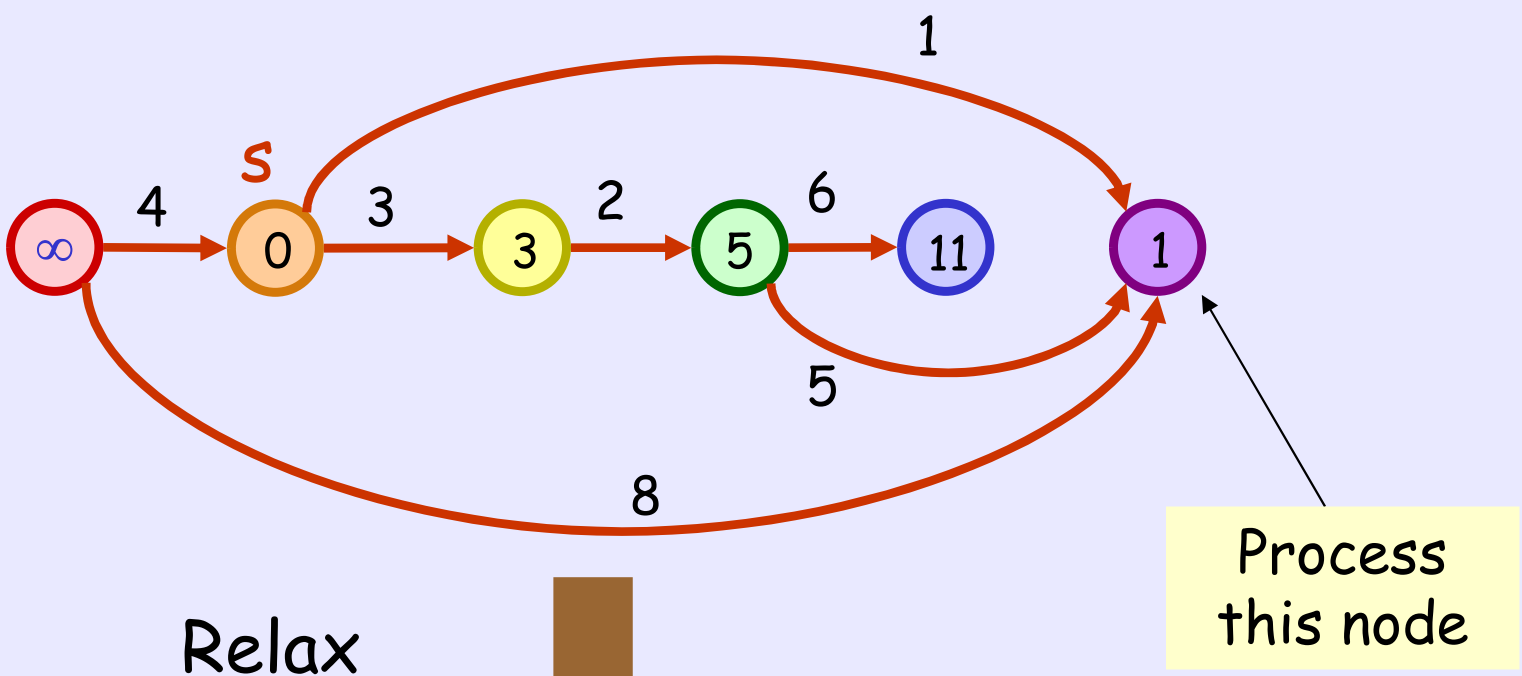
Example



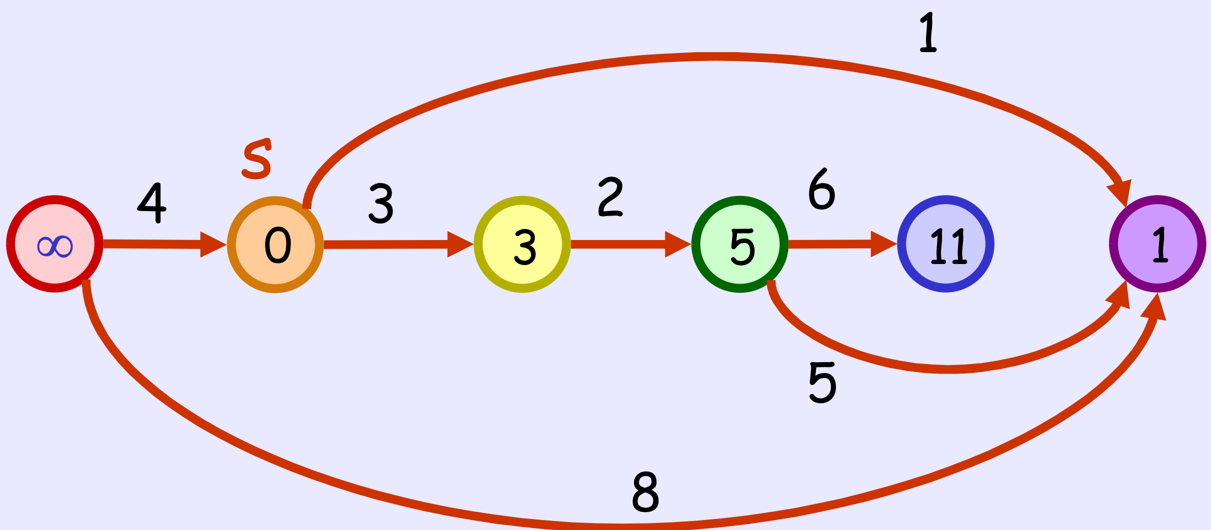
Example



Example



Relax



Correctness

- Theorem:

  By the time a vertex v is selected, d(v) will store the length of the shortest path from s to v

- How to prove ? (By induction)

Proof

- Let v_j = j^{th} vertex in the topological order
- We will show that $d(v_k)$ is set correctly when v_k is selected, for $k = 1,2, \ldots, |V|$
- When $k = 1$,

  $v_k = v_1$ = leftmost vertex

  If it is the source, $d(v_k) = 0$

  If it is not the source, $d(v_k) = \infty$

  → In both cases, $d(v_k)$ is correct (why?)

  → Base case is correct

Proof (cont)

- Now, suppose the statement is true for $k = 1, 2, \ldots, r-1$
- Consider the vertex v_r
  - If there is no path from s to v_r

    ➔ $d(v_r) = \infty$ is never changed
  - Else, we shall use similar arguments as proving the correctness of Dijkstra's algorithm …

Proof (cont)

- First, let v_t be the vertex immediately before v_r in the shortest path from **s** to v_r

  → $t \leq r-1$

  → $d(v_r)$ is set correctly once v_t is selected, and the edge (v_t, v_r) is relaxed

  → After that, $d(v_r)$ is fixed

  → $d(v_r)$ is correct when v_r is selected

- Thus, the proof of inductive case completes

Performance

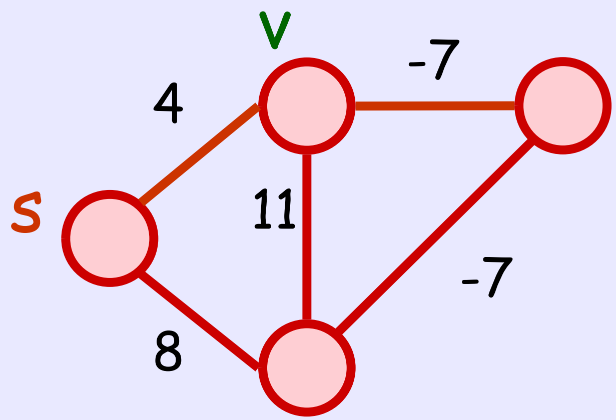
- DAG-Shortest-Path selects vertex sequentially according to topological order
  - no need to perform Extract-Min
- We can store the d values of the vertices in a single array ➔ Relax takes O(1) time
- Running Time:
  - Topological sort : O(V + E) time
  - O(V) select, O(E) Relax : O(V + E) time

➔ Total Time: O(V + E)

Handling Negative Weight Edges

- When a graph has negative weight edges, shortest path may not be well-defined

E.g.,



What is the shortest path from s to v?

Handling Negative Weight Edges

- The problem is due to the presence of a cycle C, reachable by the source, whose total weight is negative

  → C is called a negative-weight cycle

- How to handle negative-weight edges ??

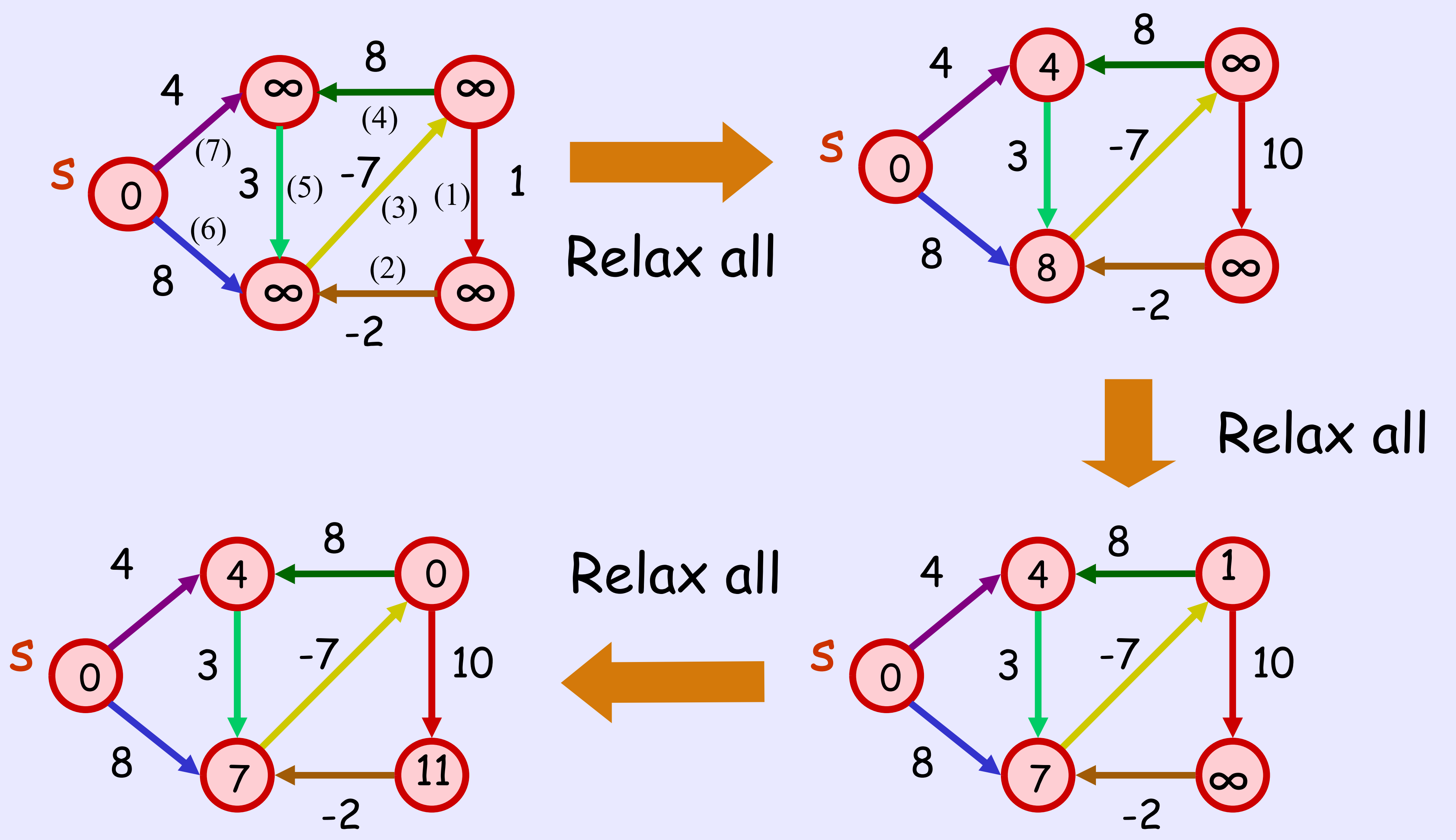
  → if input graph is known to be a DAG, DAG-Shortest-Path is still correct

  → For the general case, we can use Bellman-Ford algorithm

Bellman-Ford Algorithm

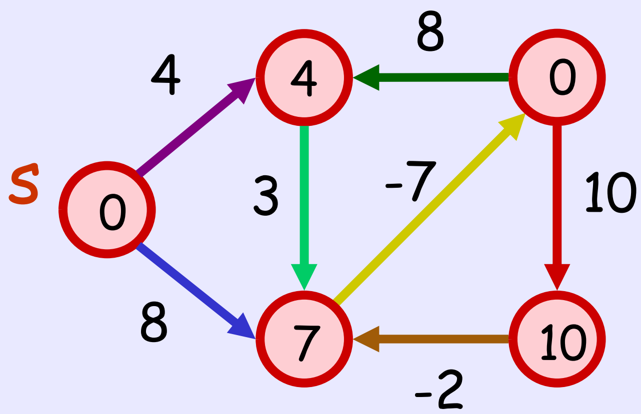
```
Bellman-Ford(G, s)      // runs in O(VE) time

  For each v, set d(v) = ∞ ; Set d(s) = 0 ;
  for (k = 1 to |V|-1)
     Relax all edges in G in any order ;
  /* check if s reaches a neg-weight cycle */
  for each edge (u,v),
     if (d(v) > d(u) + weight(u,v))
          return "something wrong !!" ;
  return d ;
```

Example 1



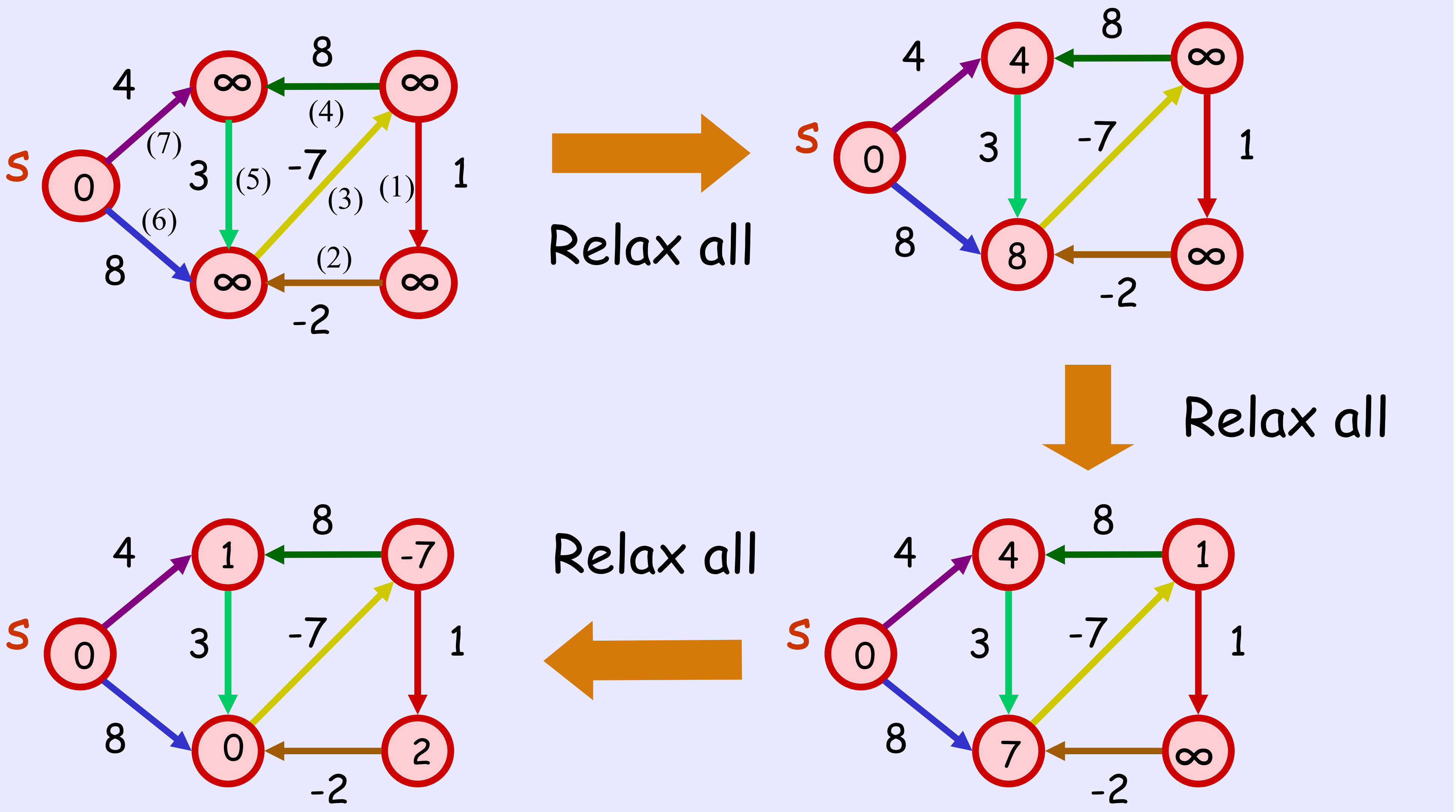
Example 1

After the 4^{th} Relax all



After checking, we found that there is nothing wrong ➔ distances are correct

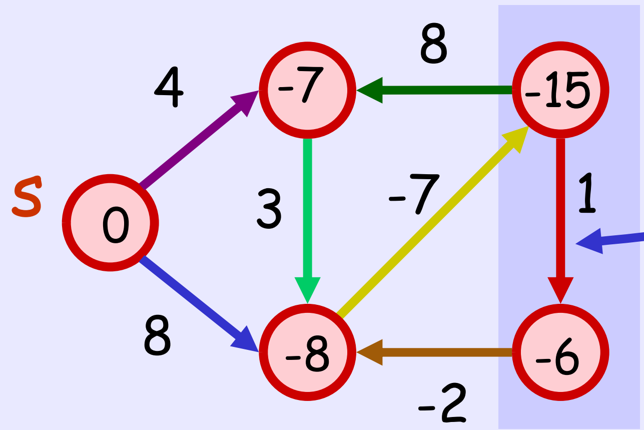
Example 2



s
0
4
8
∞
∞
∞
∞
8
3
-7
1
-2
(1)
(2)
(3)
(4)
(5)
(6)
(7)
Relax all
s
0
4
8
4
8
∞
∞
3
-7
1
-2
Relax all
s
0
4
8
4
7
1
∞
3
-7
1
-2
Relax all
s
0
4
8
1
0
-7
2
3
-7
1
-2

Example 2

After the 4^{th} Relax all



This edge shows something must be wrong …

After checking, we found that something must be wrong ➔ distances are incorrect

Correctness (Part 1)

- Theorem:

  There is a negative-weight cycle in the input graph if and only if when Bellman-Ford terminates,

  $$d(v) > d(u) + weight(u,v)$$

  for some edge (u,v)

- How to prove ? (By contradiction)

Proof

- (=>) Firstly, if there is a cycle

$$C = (v_0, v_1, \dots , v_{k-1}, v_0)$$

then total weight is negative (trivial!)

- That is, $\sum_{i = 0 \text{ to } k-1} \text{weight}(v_i, v_{(i+1) \bmod k}) < 0$

- Now, suppose on the contrary that

$$d(v) \leq d(u) + \text{weight}(u,v)$$

for all edge (u, v) at termination

Proof (cont)

- Can we obtain another bound for

$$\sum_{i = 0 \text{ to } k-1} \text{weight}(v_i, v_{(i+1) \bmod k}) \ ?$$

- By rearranging, for all edge (u,v)

$$\text{weight}(u,v) \geq d(v) - d(u)$$

$$\Rightarrow \sum_{i = 0 \text{ to } k-1} \text{weight}(v_i, v_{(i+1) \bmod k})$$

$$\geq \sum_{i = 0 \text{ to } k-1} (d(v_{(i+1) \bmod k}) - d(v_i)) = 0 \quad \text{(why?)}$$

➔ Contradiction occurs !!

(<=) by next corollary

Corollary

- Corollary: If there is no negative-weight cycle, then when Bellman-Ford terminates,

  $$d(v) \le d(u) + weight(u,v),$$

  for all edge (u,v)

Proof: By the next theorem, $d(u)$ and $d(v)$ are the cost of shortest path from s to u and v, respectively. Thus, we must have

$$d(v) \le \text{cost of any path from } s \text{ to } v$$

$$\Rightarrow d(v) \le d(u) + weight(u,v)$$

Correctness (Part 2)

- Theorem:

  If the graph has no negative-weight cycle, then for any vertex v with shortest path from s consists of k edges, Bellman-Ford sets d(v) to the correct value after the k^{th} Relax all edges (for any ordering of edges in each Relax all)

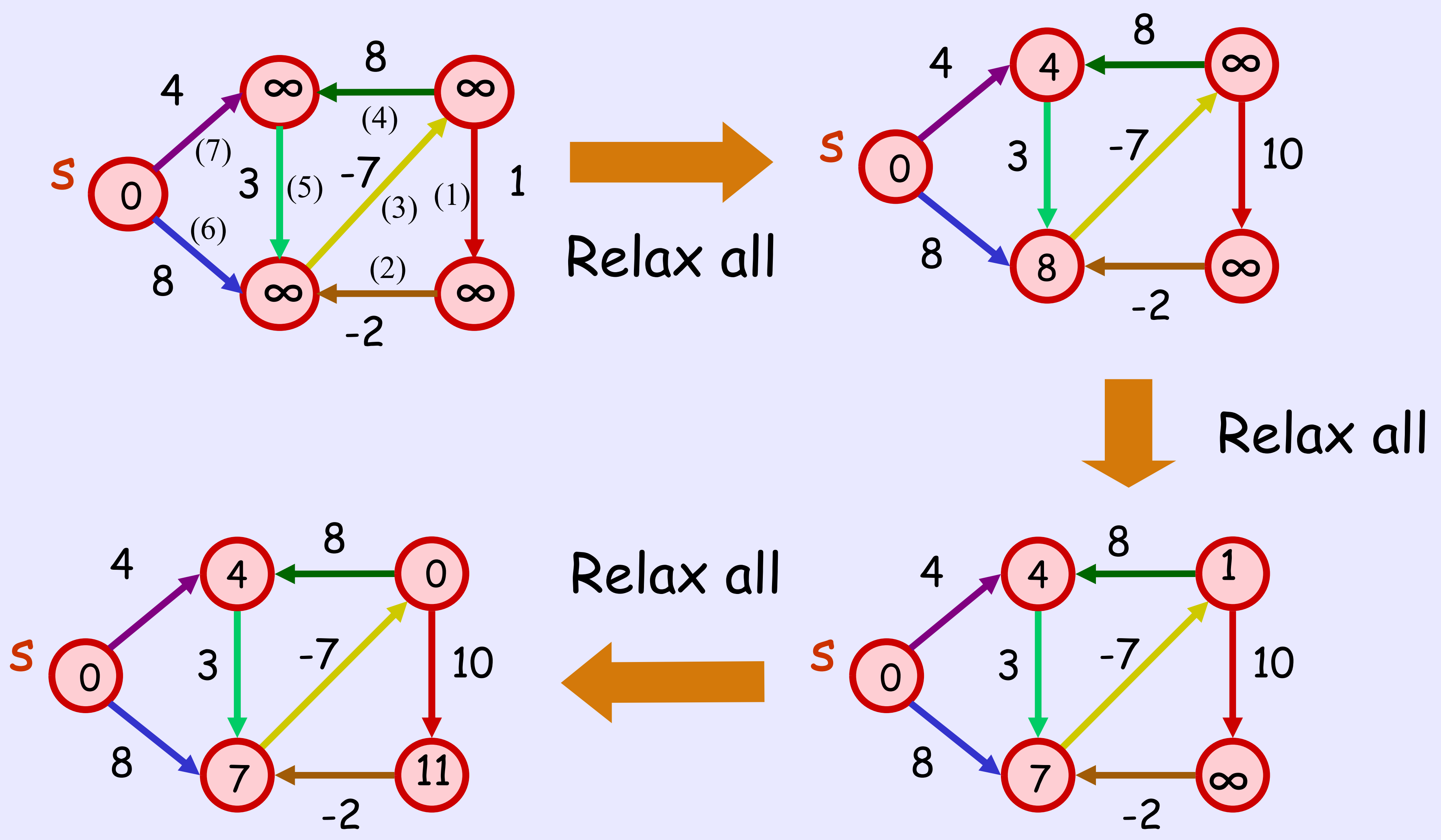
- How to prove ?

Path-Relaxation Property

- Consider any shortest path p from $s = v_0$ to v_k, and let $p = (v_0, v_1, \ldots, v_k)$. If we relax the edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ in order, then $d(v_k)$ is the shortest path from s to v_k. Proof by induction (omit)

Example:

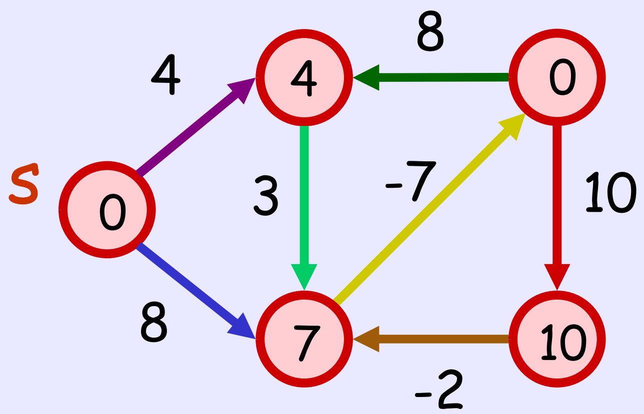
Example 1



s
0
4
8
∞
∞
∞
∞
(7)
(6)
3
(5)
-7
(4)
(3)
(1)
1
(2)
-2
Relax all
s
0
4
8
4
8
∞
∞
3
-7
10
-2
Relax all
s
0
4
8
4
7
1
∞
3
-7
10
-2
Relax all
s
0
4
8
4
7
0
11
3
-7
10
-2

Example 1

After the 4th Relax all



After checking, we found that there is nothing wrong ➔ distances are correct

Proof

- Consider any vertex v that is reachable from s, and let $p = (v_0, v_1, \ldots, v_k)$, where $v_0 = s$ and $v_k = v$ be any shortest path from s to v.
- p has at most $|V| - 1$ edges, and so $k \leq |V| - 1$. Each of the $|V| - 1$ iterations relaxes all $|E|$ edges.
- Among the edges relaxed in the ith iteration, (for $i = 1, 2, \ldots k$) is (v_{i-1}, v_i).
- By the path-relaxation property, $d(v) = d(v_k)$ = the shortest path from s to v.

Performance

- When no negative edges
  - Dijkstra's algorithm
    - Using array $O(V^2)$
    - Using Binary heap implementation: $O(E \lg V)$
    - Using Fibonacci heap: $O(E + V\log V)$
- When DAG
  - DAG-Shortest-Paths: $O(E + V)$ time
- When negative cycles
  - Using Bellman-Ford algorithm: $O(V E) = (V^3)$

Homework

- Exercises: 24.1-3, 24.2-3, 24.2-4, 24.3-5, 24.3-8, 24.3-10

Quiz

- Which of the following statements are true for the Minimum Spanning Tree (MST) of a graph G =(V, E)?

a. MST is the spanning tree that have the minimum weight

b. MST of a graph is not unique

c. MST has exactly |V| -1 edges