Design and Analysis of Algorithms

Instructor: Sharma Thankachan

Lecture 16:

Single-Source Shortest-Path

Slides modified from Dr. Hon, with permission

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

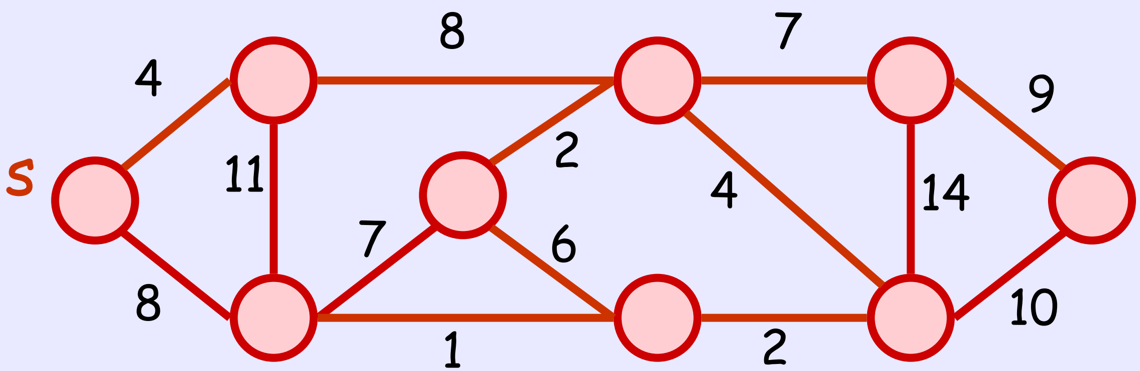
Single-Source Shortest Path

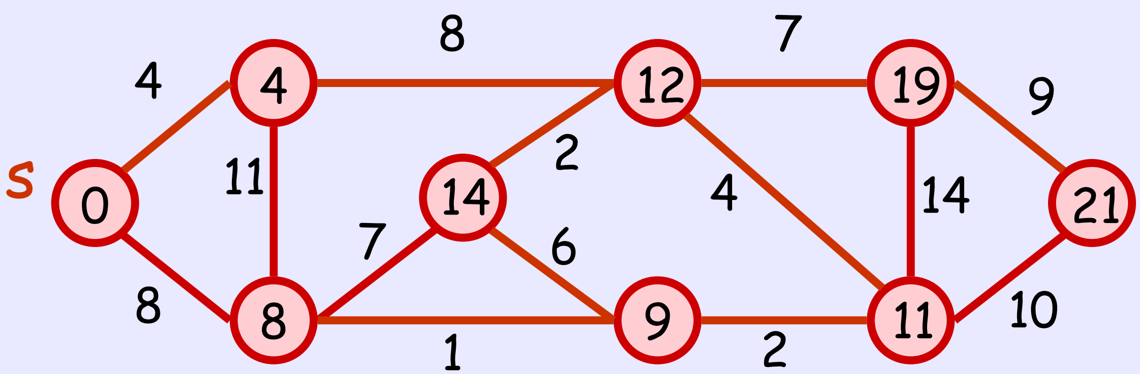
- Let $G = (V,E)$ be a weighted graph
  - the edges in G have weights
  - can be directed/undirected
  - can be connected/disconnected
- Let s be a special vertex, called source

Target: For each vertex v, compute the length of shortest path from s to v

Single-Source Shortest Path

- E.g.,

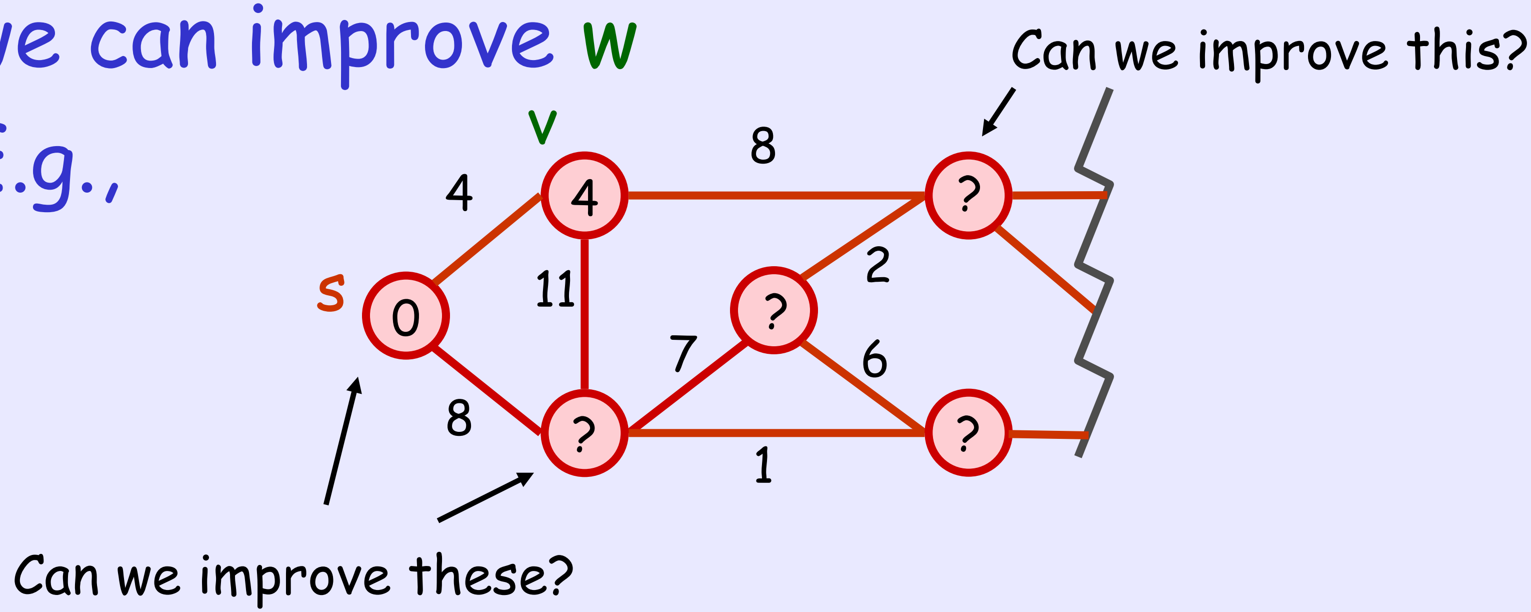




Relax

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

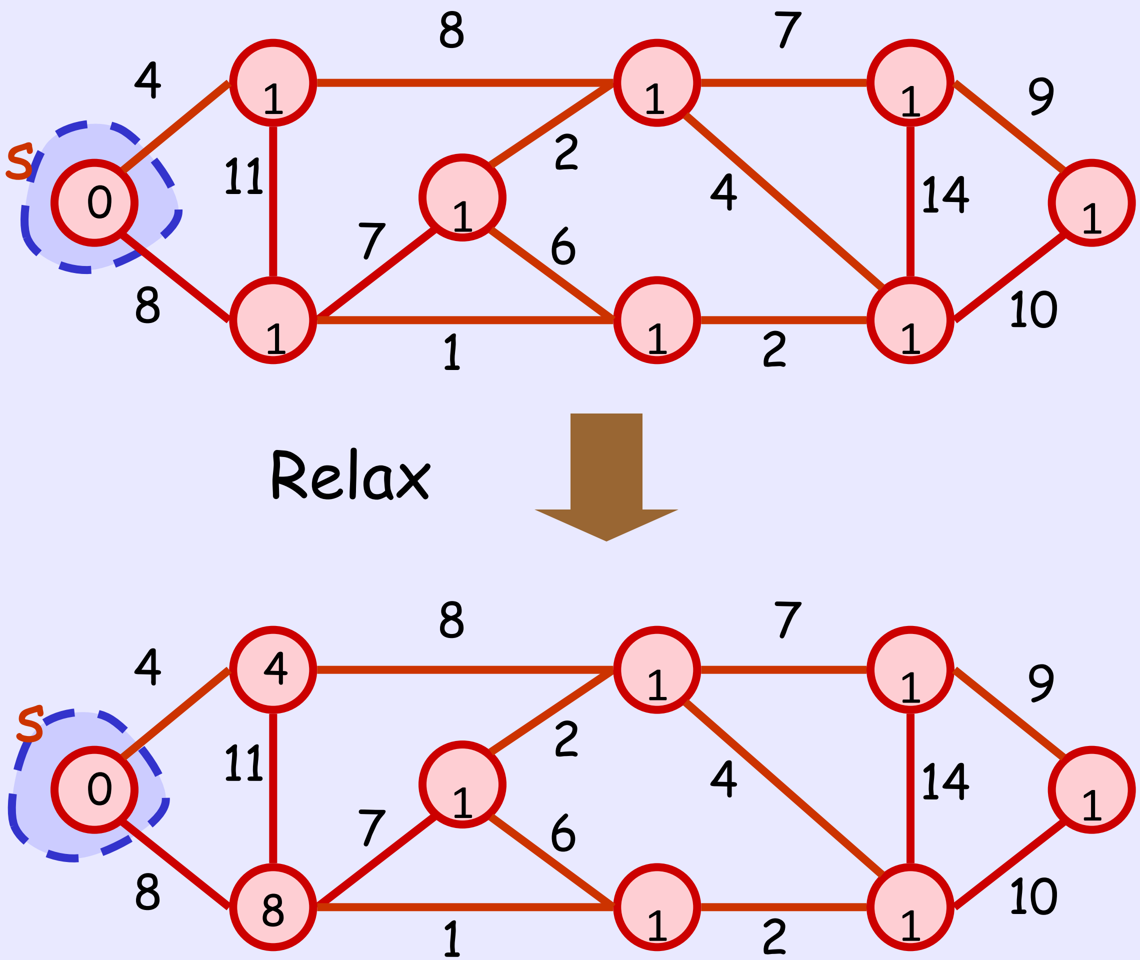
  when a vertex v can be reached from the source with a certain distance, we examine an outgoing edge, say (v,w), and check if we can improve w
- E.g.,



Dijkstra's Algorithm

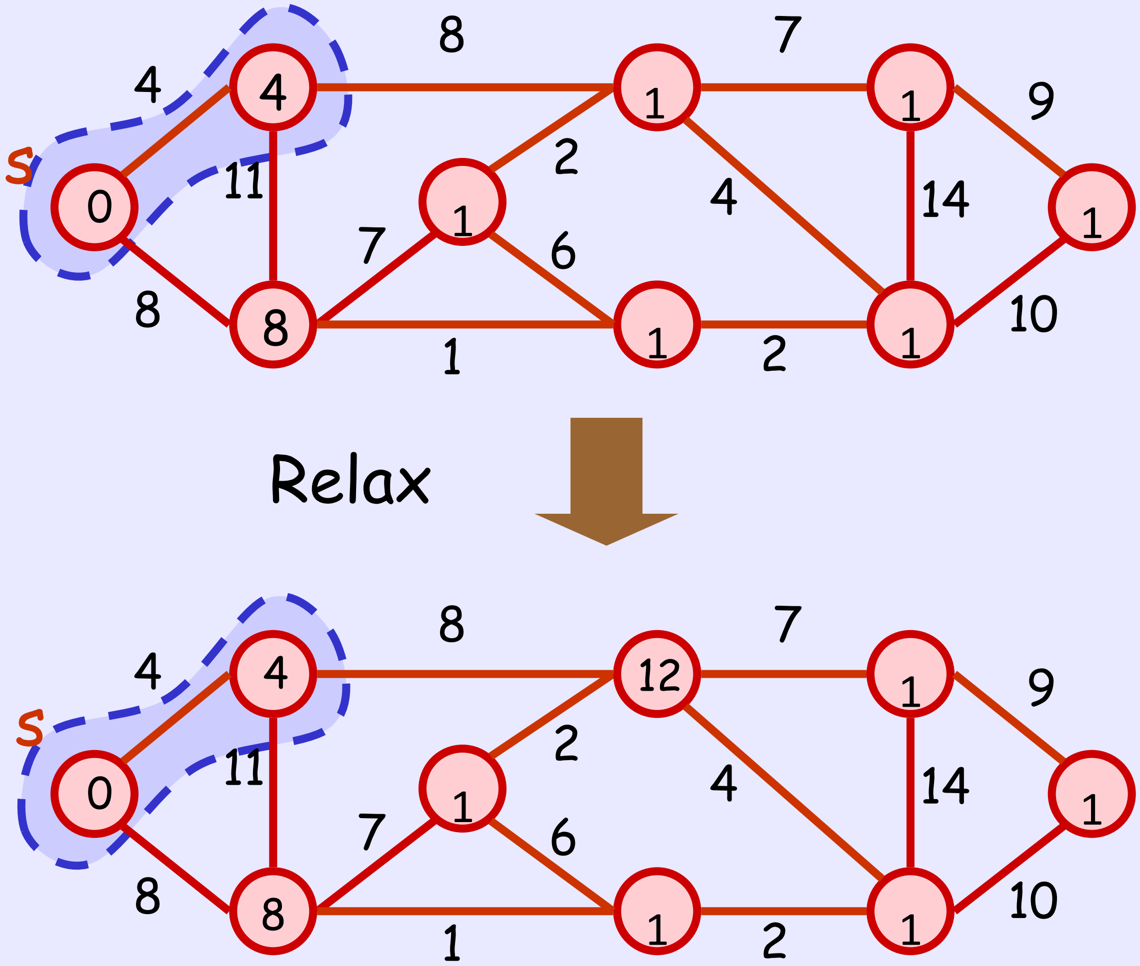
```
Dijkstra(G, s)
  For each vertex v,
     Mark v as unvisited, and set d(v) = 1 ;
  Set d(s) = 0 ;
  while (there is unvisited vertex) {
    v = unvisited vertex with smallest d ;
    Visit v, and Relax all its outgoing edges;
  }
  return d ;
```

Example

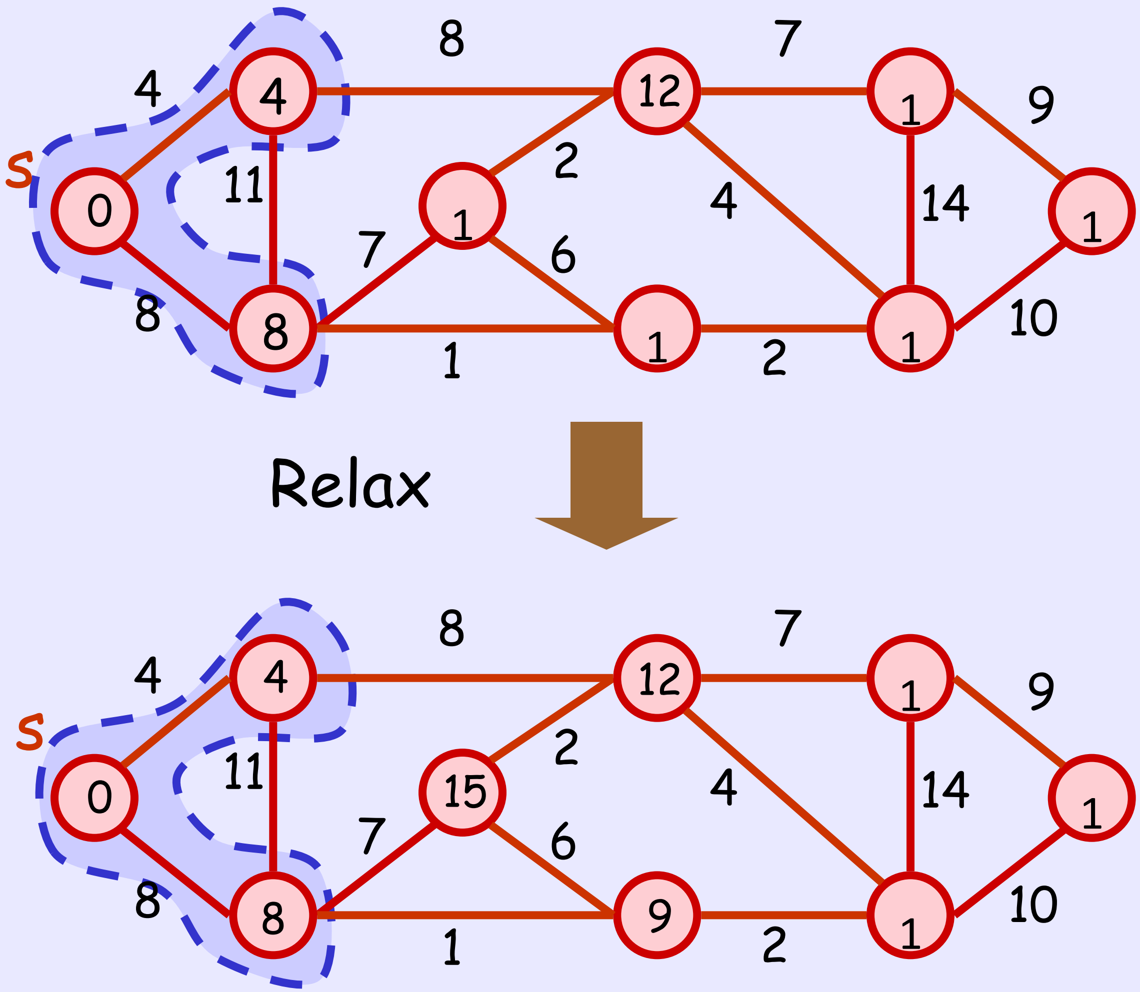


s
0
1
1
1
1
1
1
1
1
4
8
11
7
2
6
1
7
4
2
14
9
10
Relax
s
0
4
8
1
1
1
1
1
1
4
8
11
7
2
6
1
7
4
2
14
9
10

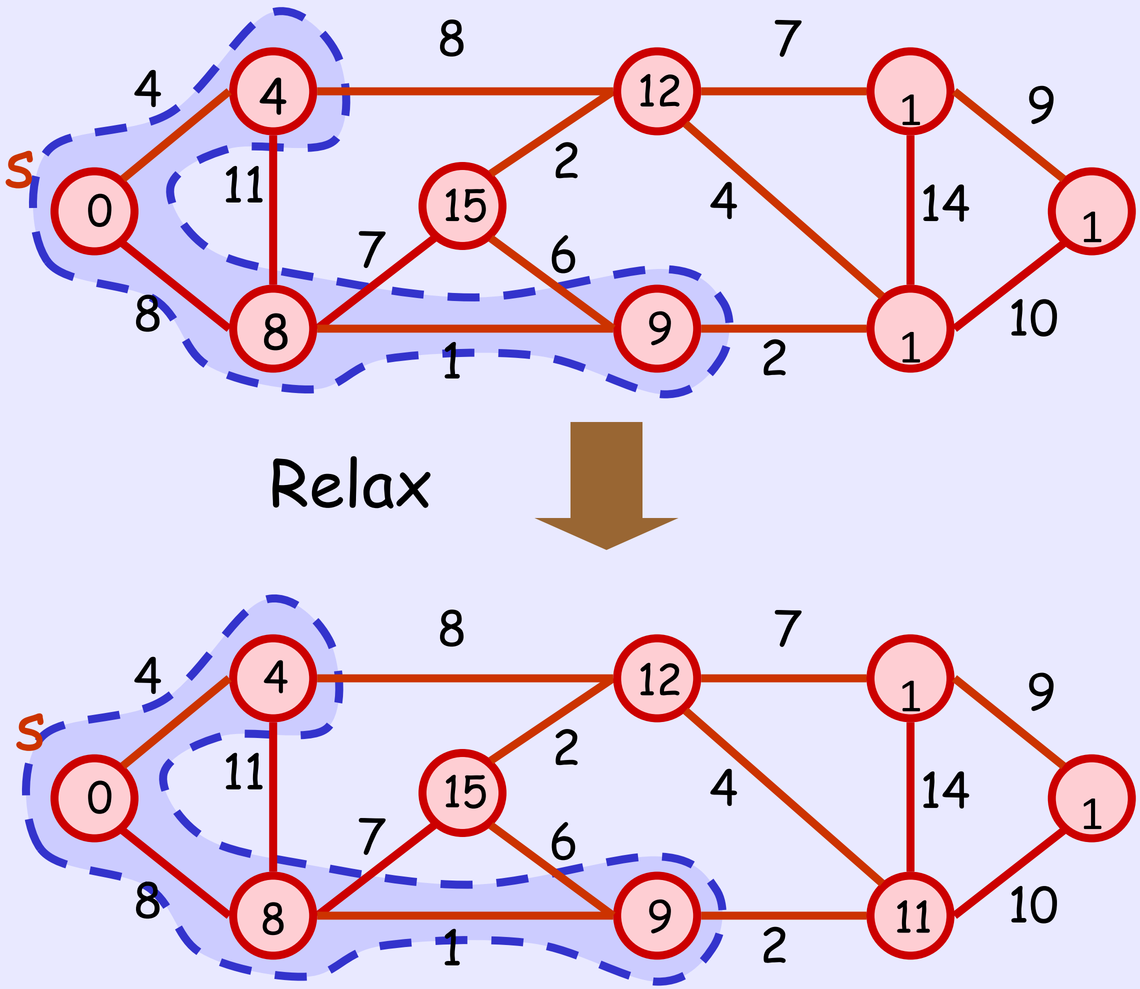
Example


s
0
4
8
1
8
7
9
4
11
7
2
6
4
14
8
1
2
10
Relax
s
0
4
8
12
1
8
7
9
4
11
7
2
6
4
14
8
1
2
10

Example

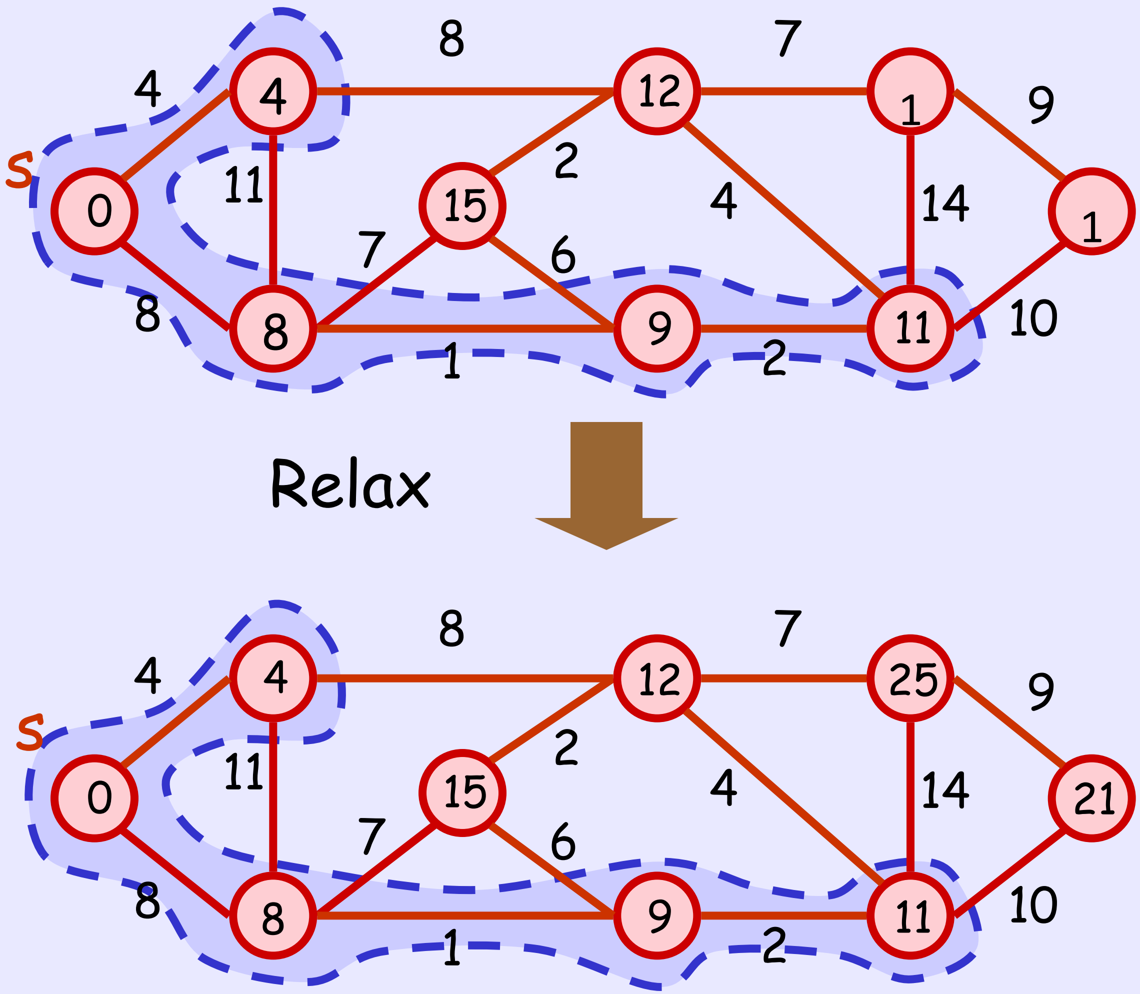

s
0
4
8
12
1
1
1
1
1
4
8
11
7
6
2
4
7
9
14
10
1
2
Relax
s
0
4
8
12
15
9
1
1
1
4
8
11
7
6
2
4
7
9
14
10
1
2

Example

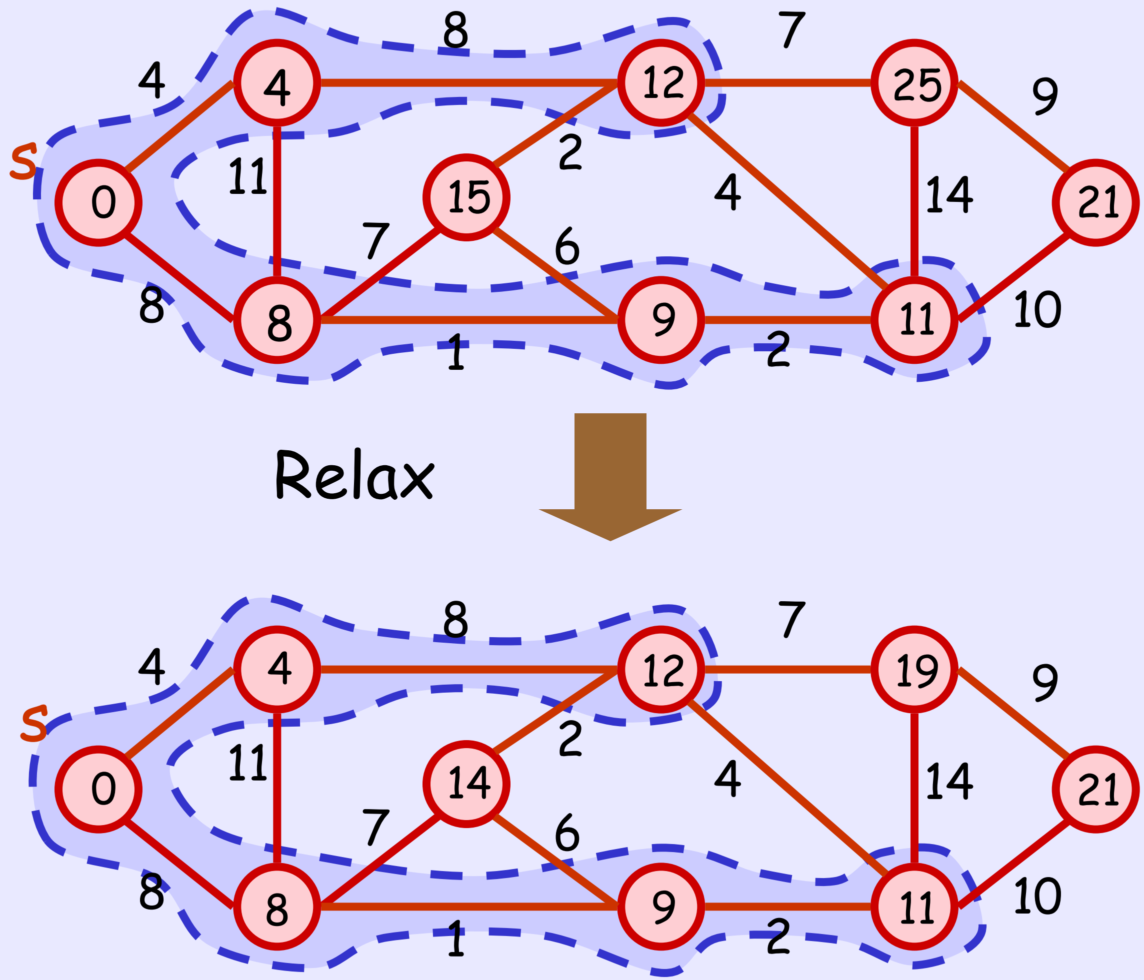


s
0
4
8
12
15
9
1
1
1
8
7
4
11
2
7
6
4
14
9
8
1
2
10
Relax
s
0
4
8
12
15
9
1
1
11
8
7
4
11
2
7
6
4
14
9
8
1
2
10

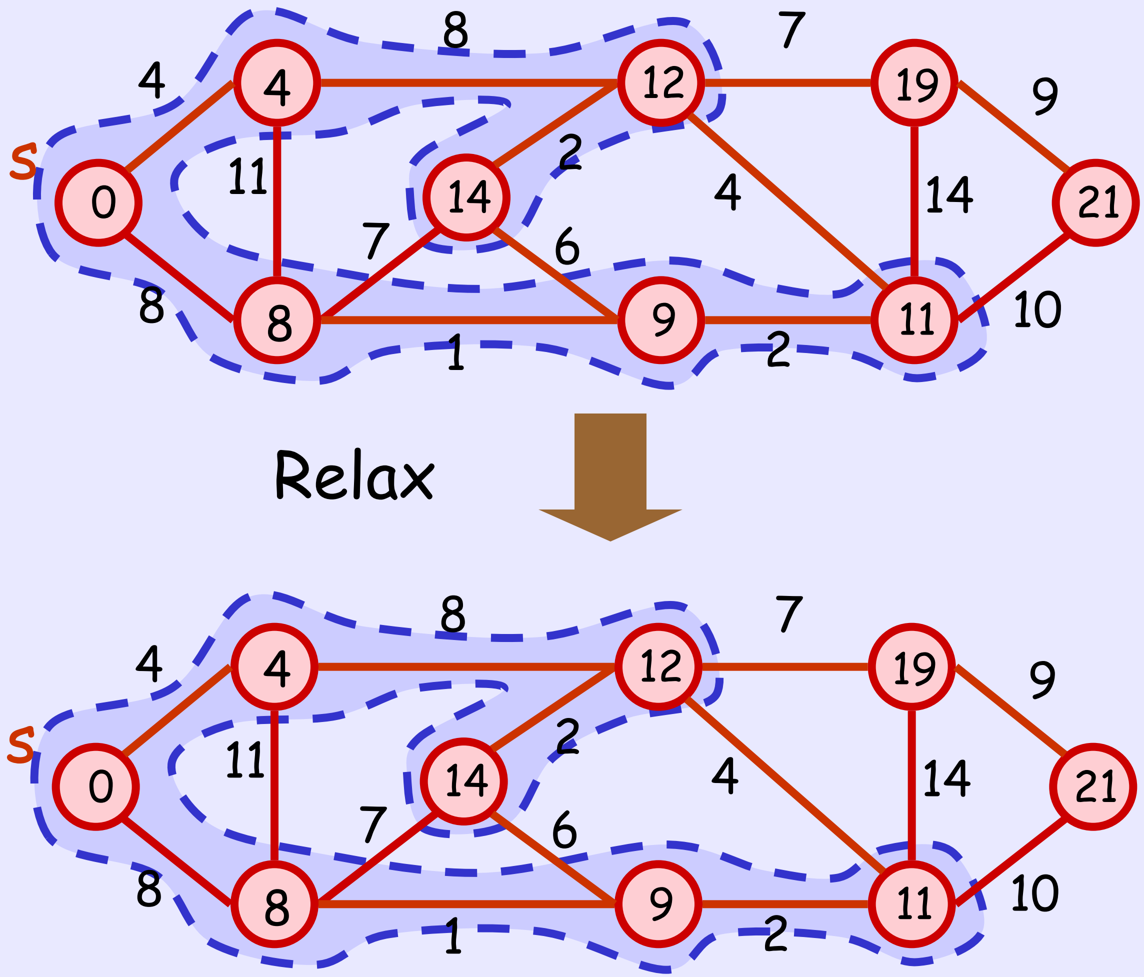
Example


s
0
4
8
12
15
9
11
1
1
8
7
4
11
2
4
14
9
7
6
8
1
2
10
Relax
s
0
4
8
12
15
9
11
25
21
8
7
4
11
2
4
14
9
7
6
8
1
2
10

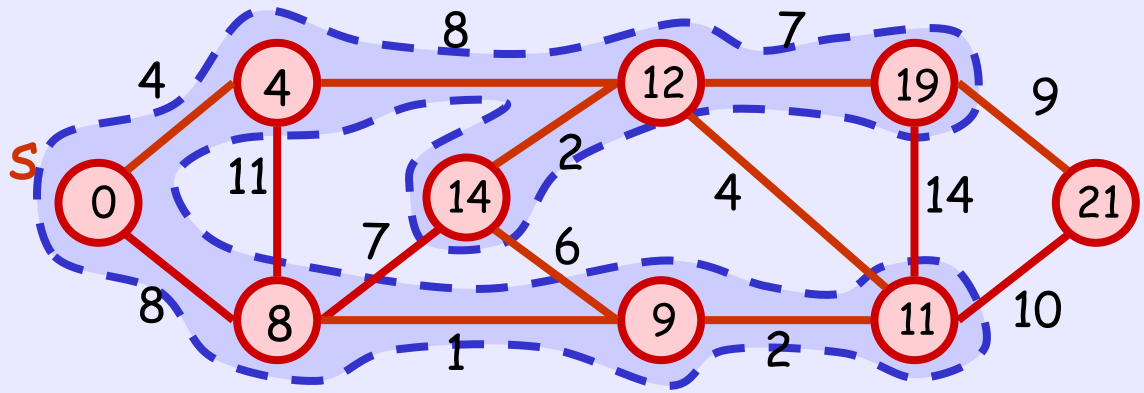
Example



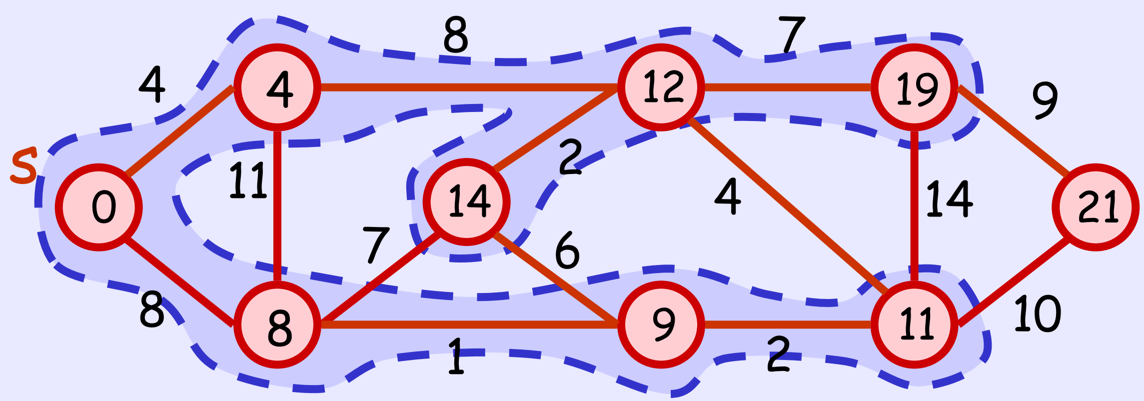
Example


s
0
4
8
12
14
9
19
11
21
4
8
11
8
7
2
1
7
6
4
2
9
14
10
Relax
s
0
4
8
12
14
9
19
11
21
4
8
11
8
7
2
1
7
6
4
2
9
14
10

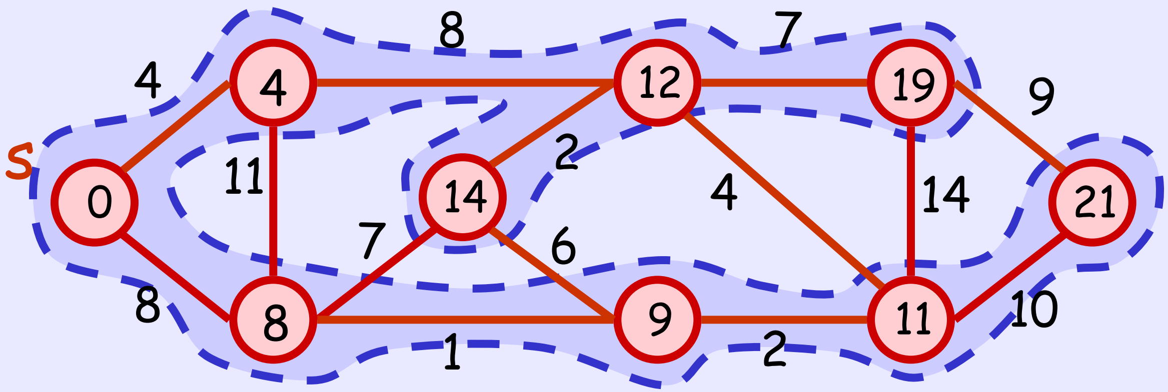
Example



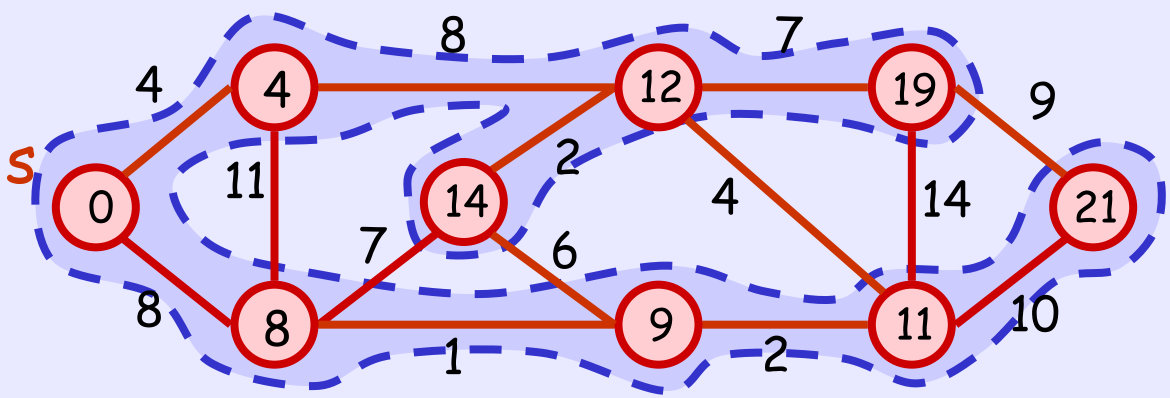
Relax



Example



Relax



Correctness

Theorem:

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

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

    ➔ $d(v_r) = 1$ 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 $t \le r-1$ (why??)

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

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

➔ $d(v_r)$ 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 using Fibonacci Heap,
  - Relax ⇔ Decrease-Key
  - Pick vertex ⇔ Extract-Min
- Running Time:
  - O(V) Insert/Extract-Min
  - At most O(E) Decrease-Key

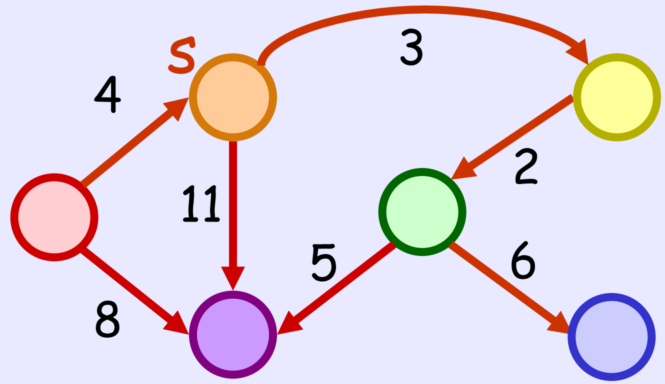
➔ Total Time: O(E + V log V)

Finding Shortest Path in DAG

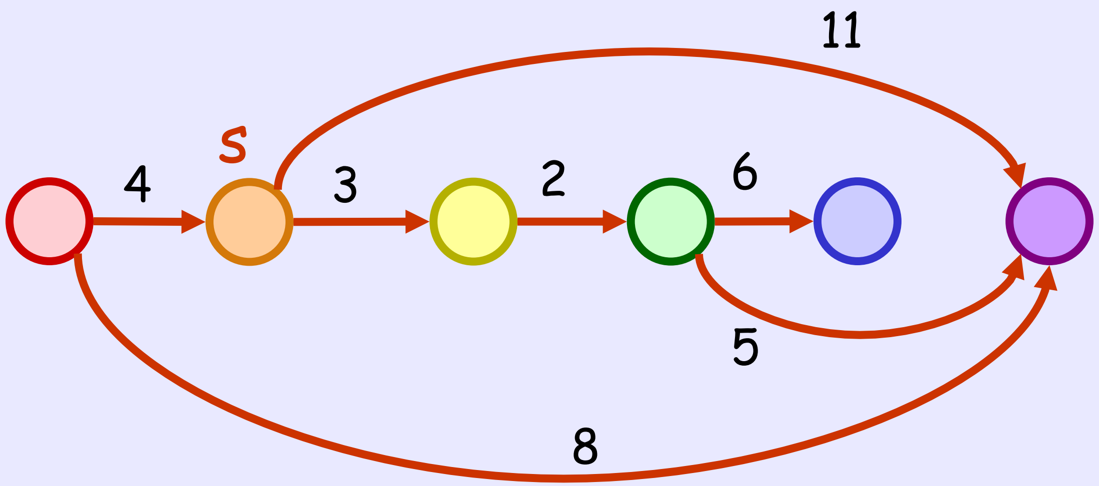
We have a faster algorithm for DAG :

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

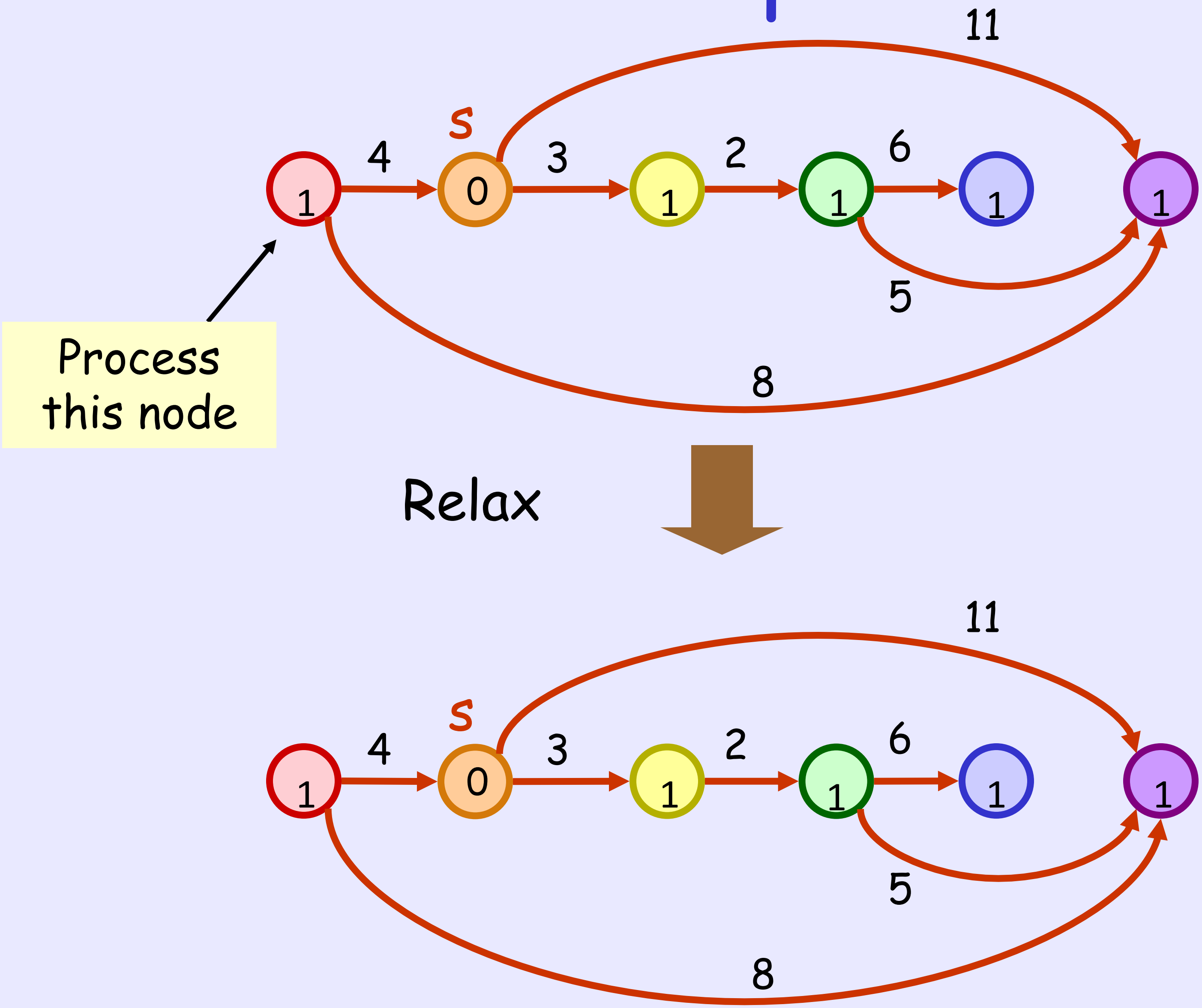
Example



Topological Sort

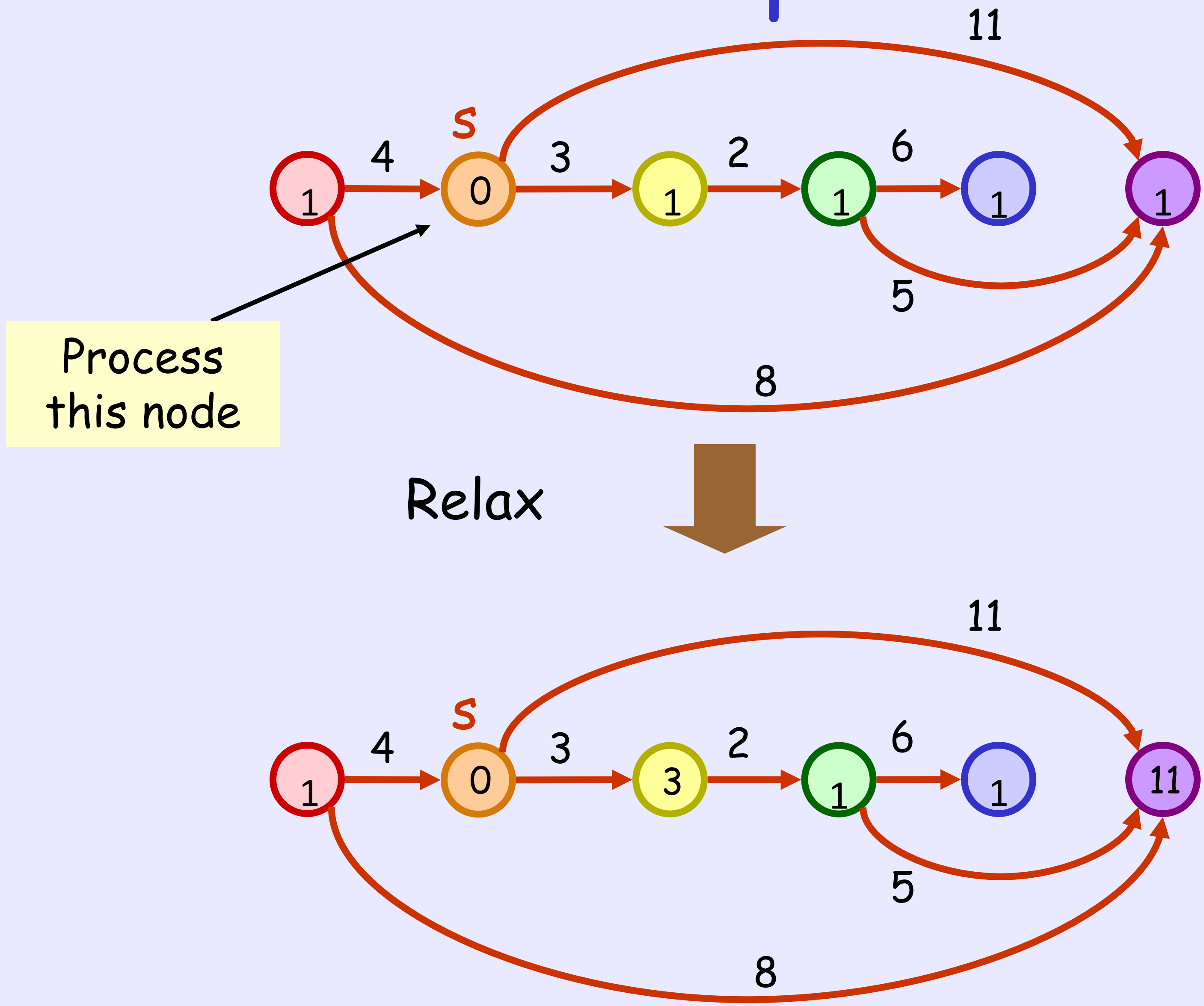


Example



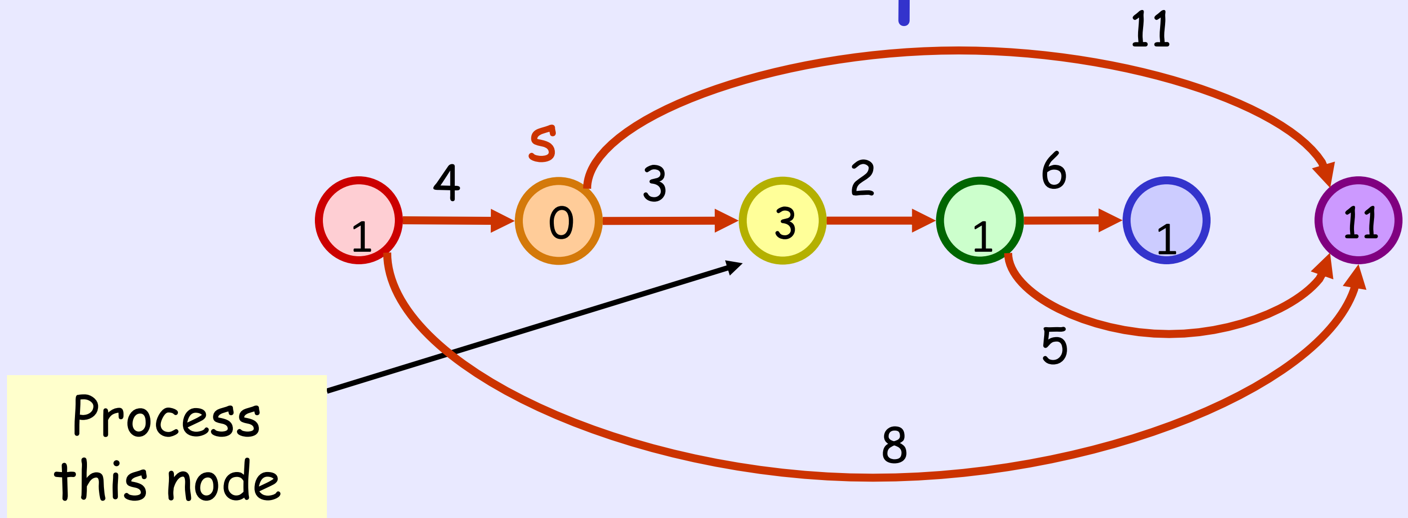
11
s
4
3
2
6
1
0
1
1
1
1
5
8
Process this node
Relax
11
s
4
3
2
6
1
0
1
1
1
1
5
8

Example

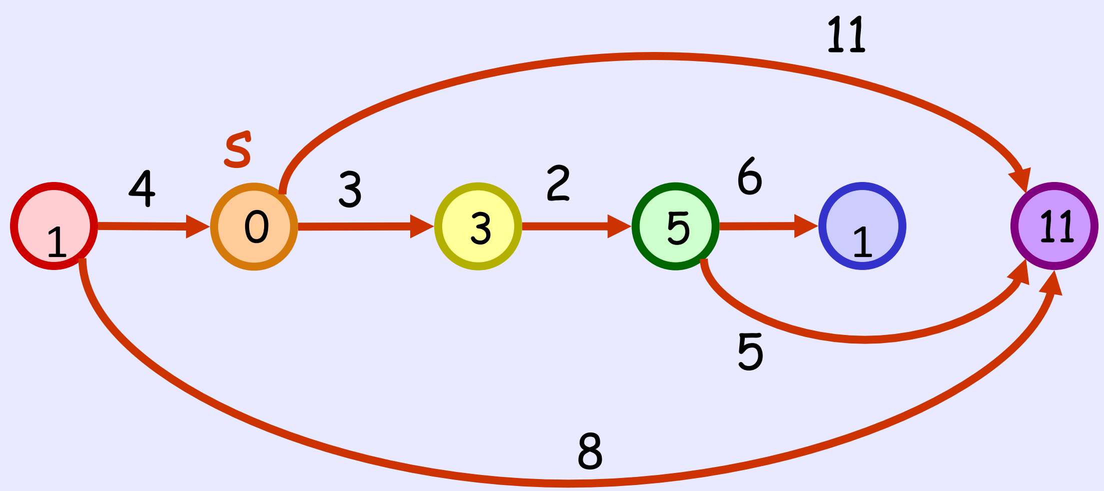


s
4
3
2
6
11
5
8
1
0
1
1
1
1
Process this node
Relax
s
4
3
2
6
11
5
8
1
0
3
1
1
11

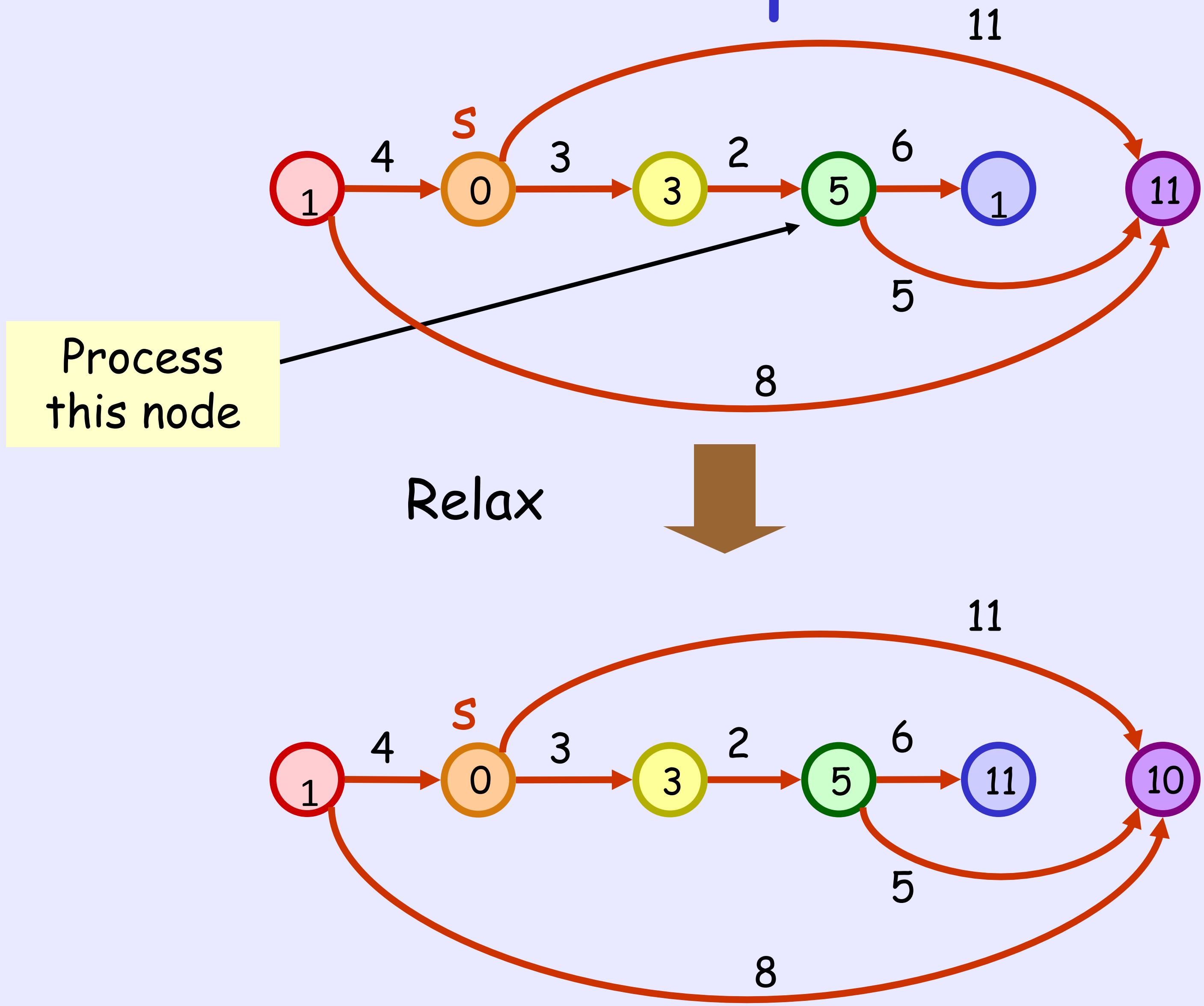
Example



Relax

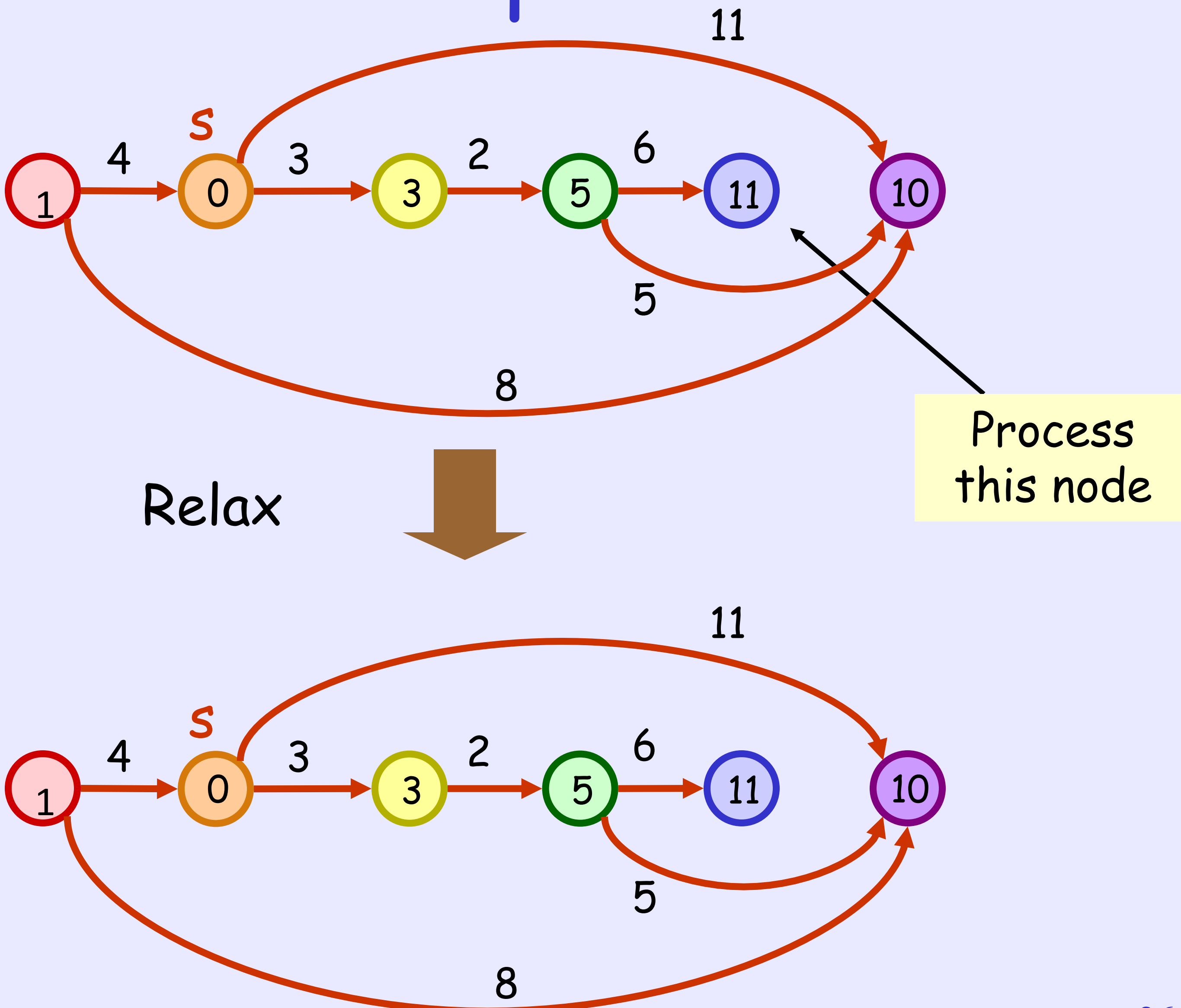


Example

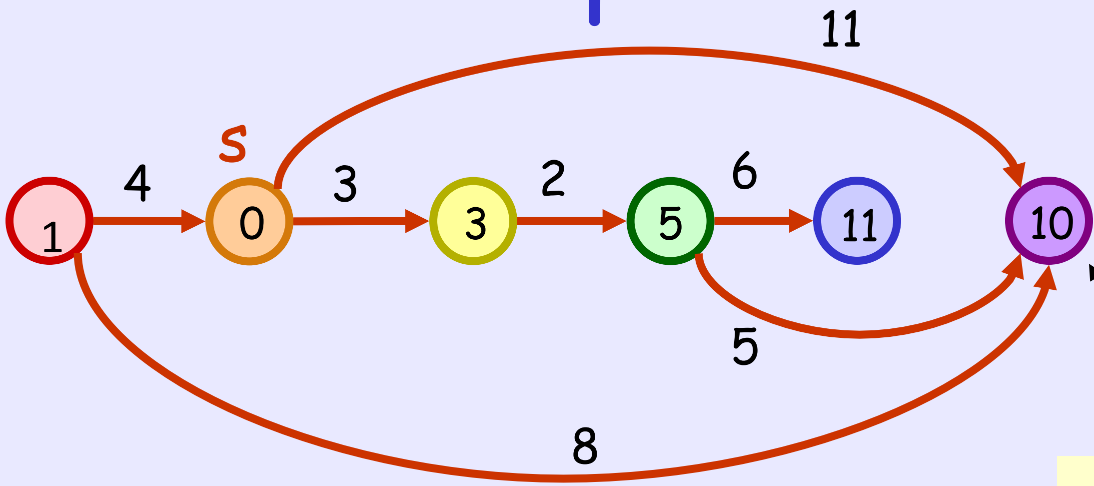


11
s
4
3
2
6
1
0
3
5
1
11
5
8
Process this node
Relax
11
s
4
3
2
6
1
0
3
5
11
10
5
8

Example

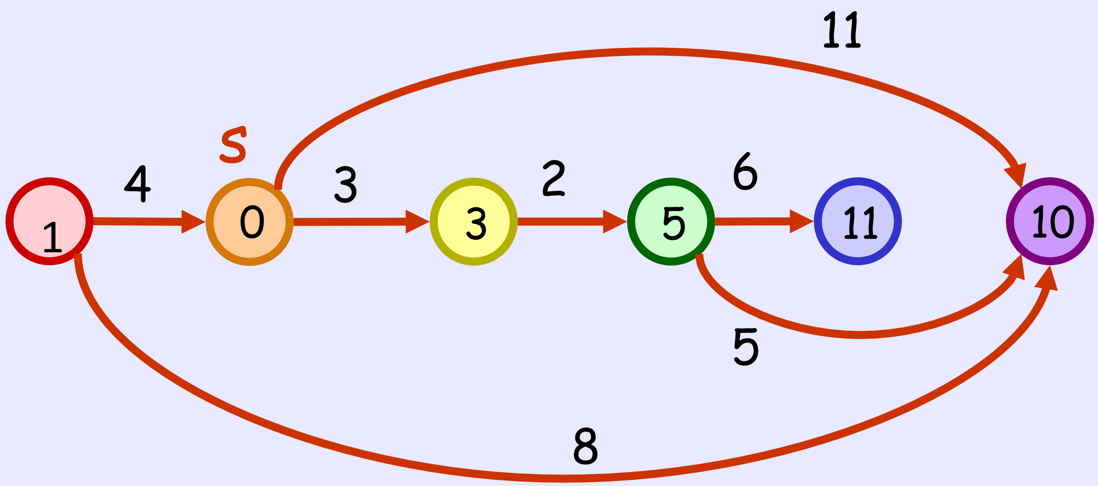


Example



Process this node

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, $d(v_k) = 1$

  → 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

    → $d(v_r) = 1$ 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

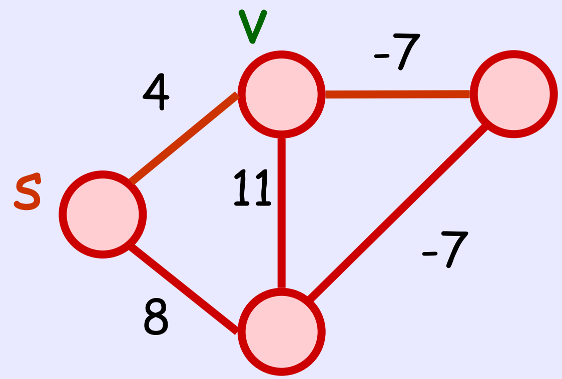
- DAG-Shortest-Path selects vertex sequentially according to topological order
  - no need to perform Extract-Min
- We can store the d values of the vertices in a single array ➔ Relax takes O(1) time
- Running Time:
  - Topological sort : O(V + E) time
  - O(V) select, O(E) Relax : O(V + E) time

➔ 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 ??

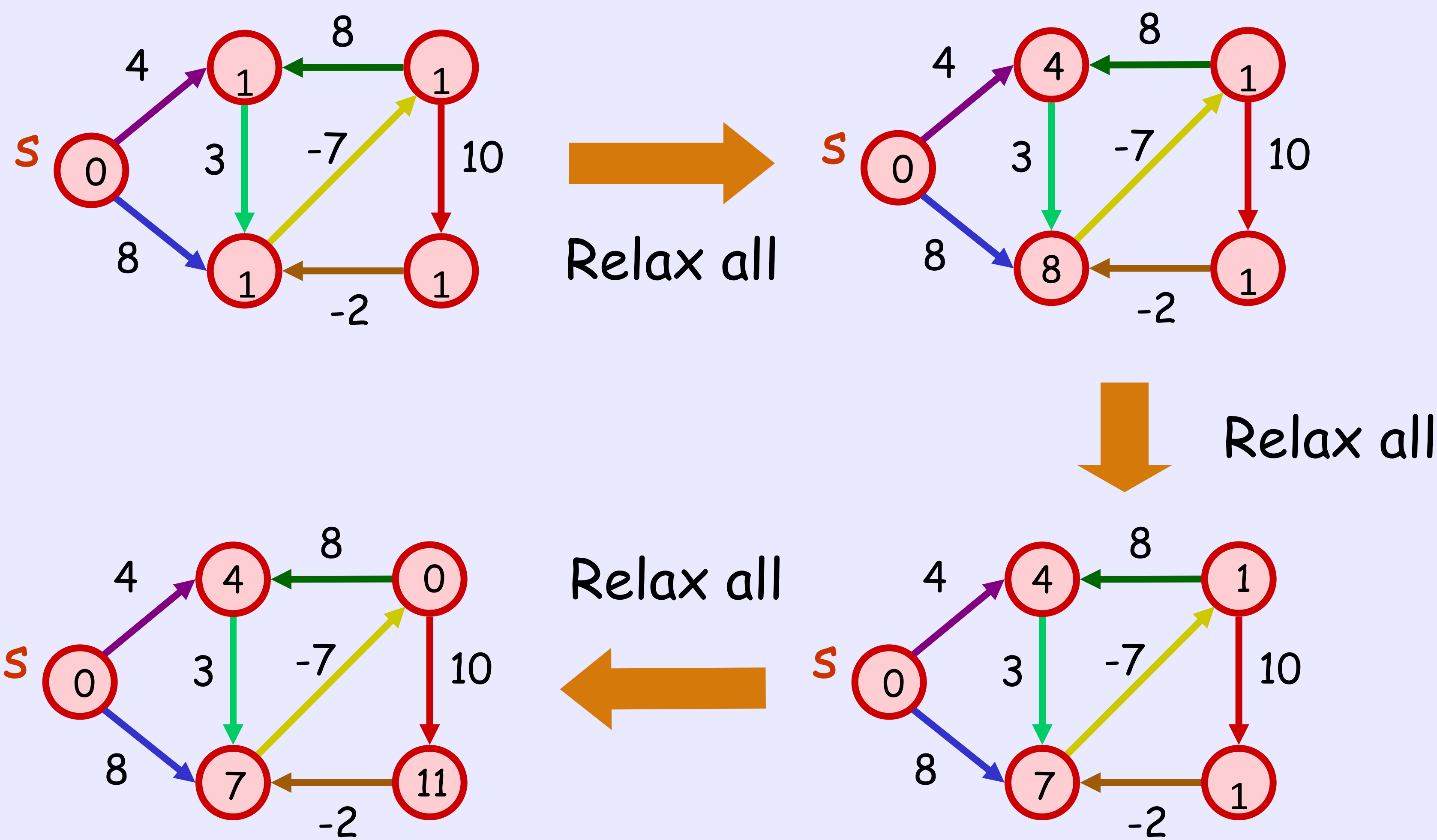
  ➔ if input graph is known to be a DAG, DAG-Shortest-Path is still correct

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

Bellman-Ford Algorithm

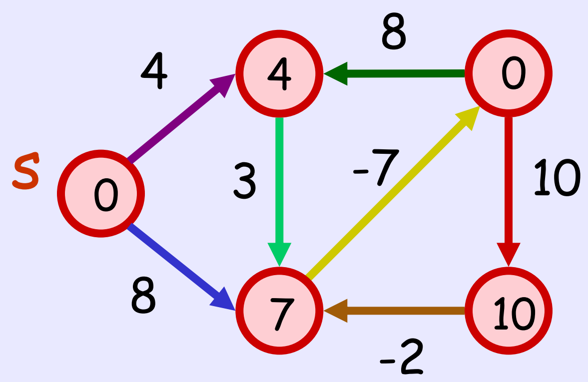
```
Bellman-Ford(G, s)     // runs in O(VE) time

  For each v, set d(v) = 1 ; 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


s
0
1
1
1
1
4
8
3
-7
10
8
-2
Relax all
s
0
4
1
8
1
4
8
3
-7
10
8
-2
Relax all
s
0
4
1
7
1
4
8
3
-7
10
8
-2
Relax all
s
0
4
0
7
11
4
8
3
-7
10
8
-2

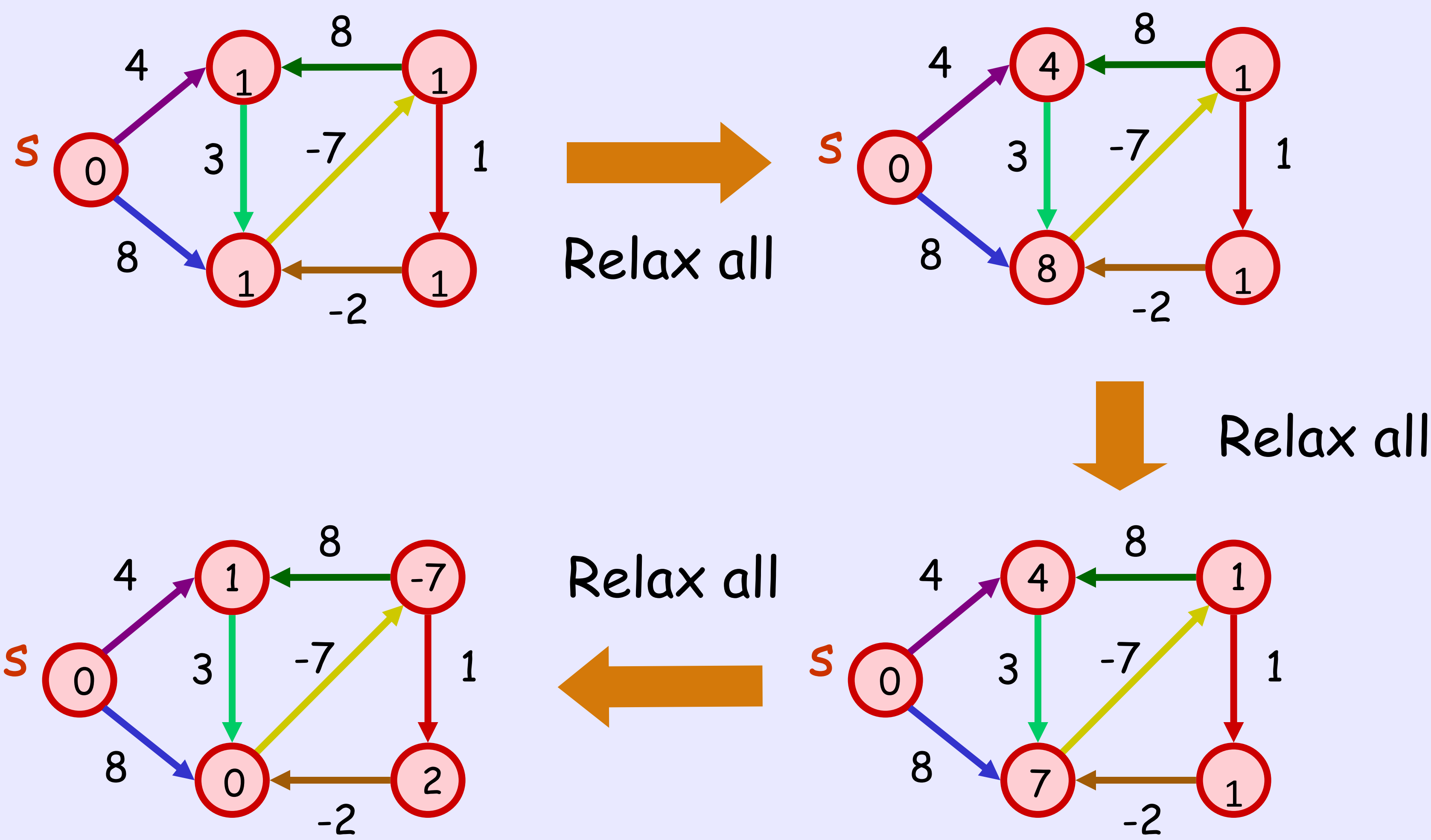
Example 1

After the 4th Relax all



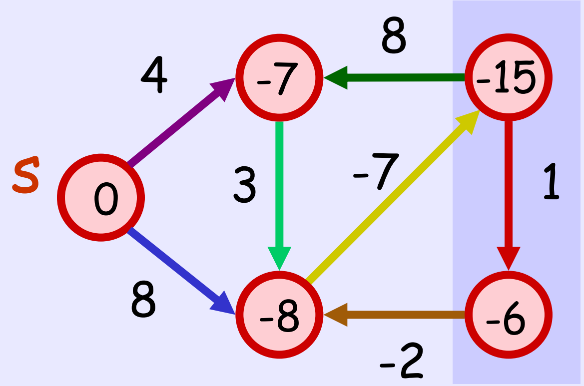
After checking, we found that there is nothing wrong ➔ distances are correct

Example 2


s
0
1
1
1
1
4
8
3
-7
1
8
-2
Relax all
s
0
4
8
1
1
4
8
3
-7
1
8
-2
Relax all
s
0
4
7
1
1
4
8
3
-7
1
8
-2
Relax all
s
0
1
0
-7
2
4
8
3
-7
1
8
-2

Example 2

After the 4th 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:

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 (for any ordering of edges in each Relax all)

How to prove ? (By induction)

Corollary

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

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

for all edge (u,v)

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

$$d(v) \leq \text{length of any path from } s \text{ to } v$$

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

"Something Wrong" Lemma

Lemma: If there is a negative-weight cycle, then when Bellman-Ford terminates,

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

for some edge (u,v)

How to prove ? (By contradiction)

Proof

- Firstly, we know that there is a cycle

$$C = (v_1, v_2, \ldots, v_k, v_1)$$

whose total weight is negative

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

- 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

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

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

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

➔ $\sum_{i = 1 \text{ to } k} \text{weight}(v_i, v_{i+1})$

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

➔ Contradiction occurs !!

Correctness (Part 2)

- Combining the previous corollary and lemma, we have:

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)