CS2x1:Data Structures and Algorithms

Koteswararao Kondepu

k.kondepu@iitdh.ac.in

Recap: SSSP

SSSP

Positive Weights | **Negative Weights**

Dijkstra

```
DIJKSTRA (G, w, s) {
  INITIALIZE-SINGLE-SOURCE (G, s)
  S = ∅
  Q = G. V
  while Q ≠ ∅ ;
     u = EXTRACT-MIN(Q)
     S = S ∪ {u}
     for each vertex v ϵ Q. Adj[u]
       RELEAX (u, v, w)
}
```

```
INITIALIZE-SINGLE-SOURCE (G, s)
{
 for each v ϵ G.V
    v. d = ∞
    v. π = NIL
 s. d = 0
```

Bellman-Ford

```
RELAX (u, v, w) {
  if v.d > u.d + w(u, v)
     v. d =  u.d + w(u, v)
     v. π = u
```

Total time complexity: <u>O(V) + O (V log V) + O(V) + O(E logV)</u>

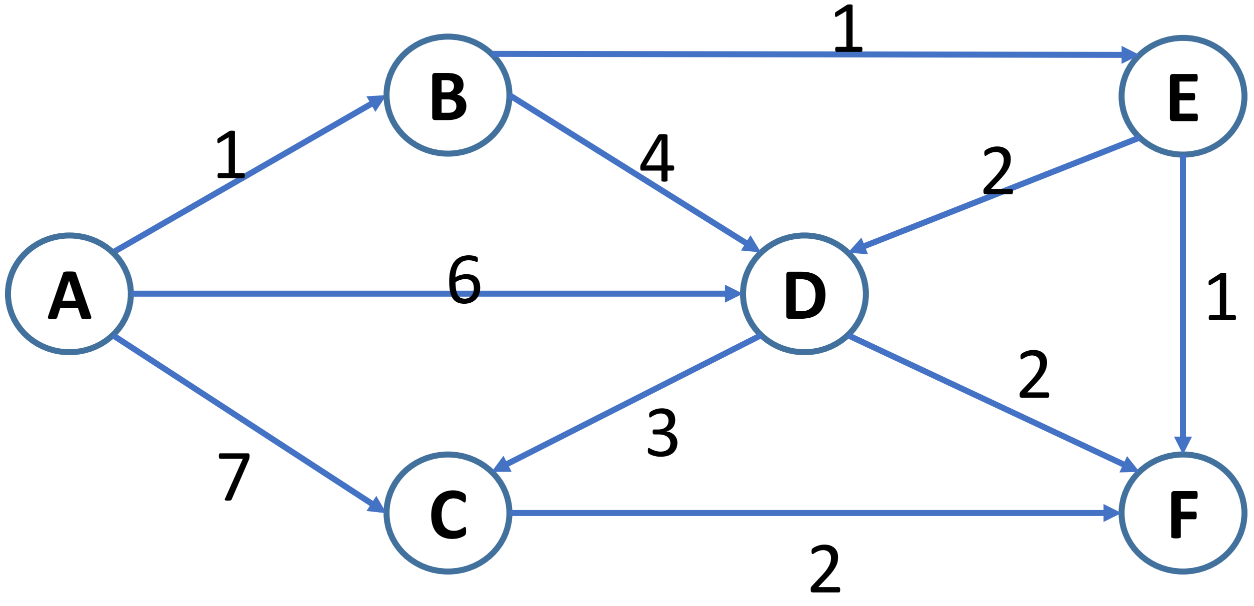
: O(ElogV)

Exercise: Dijkstra

Consider the following digraph starting at ***vertex A*** and apply Dijkstra's single source shortest path algorithm on it.

Select the correct order from the following in which vertices are removed from the Priority Queue?

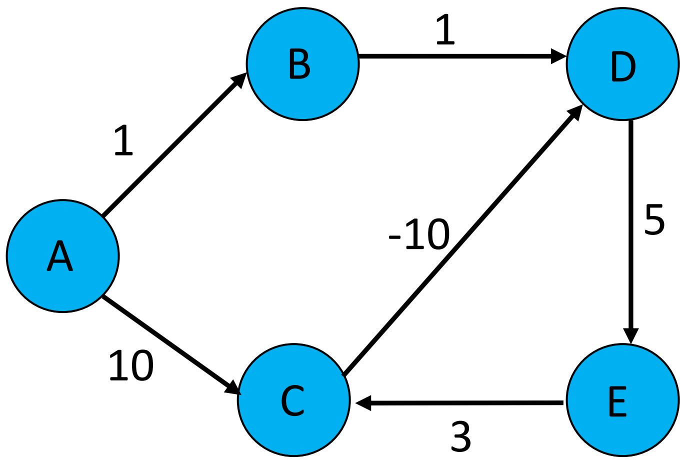
a. A, B, C, D, E, F
b. A, B, F, E, C, D
c. A, B, F, E, D, C
d. A, B, E, F, D, C



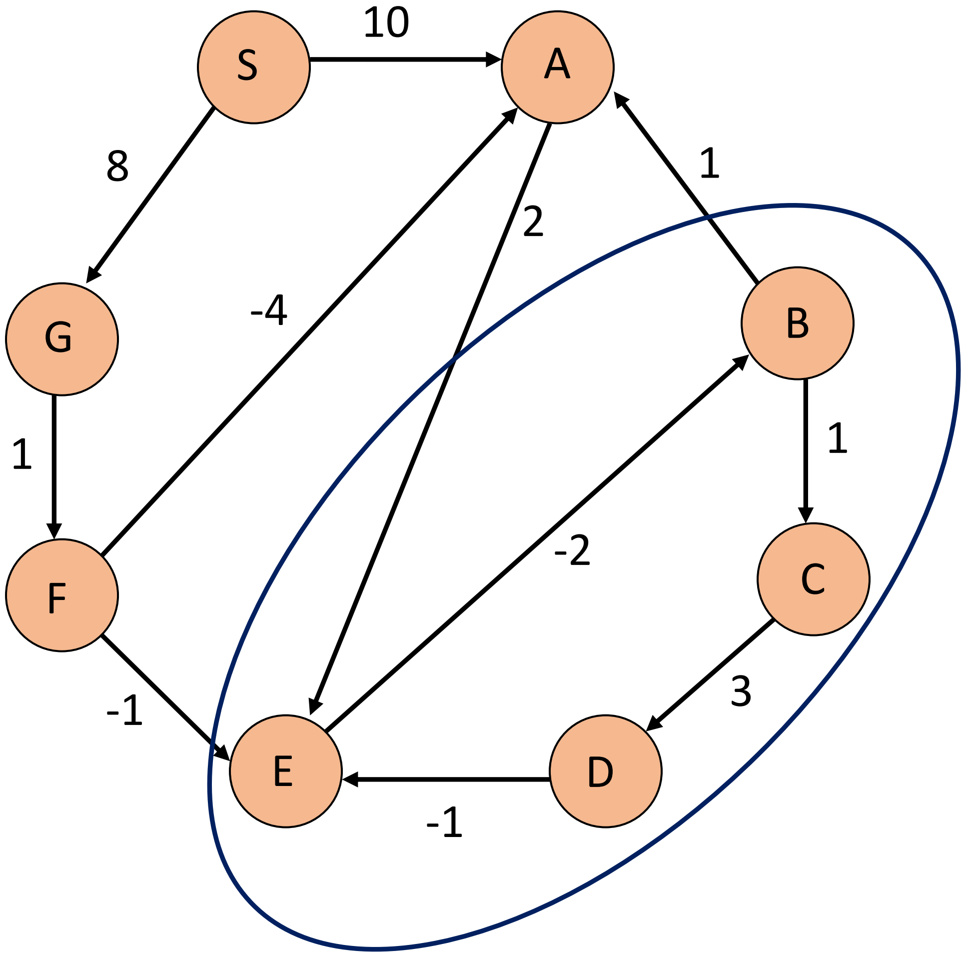
Exercise: Dijkstra (1)

Negative cycles

What is the shortest path from A to E?

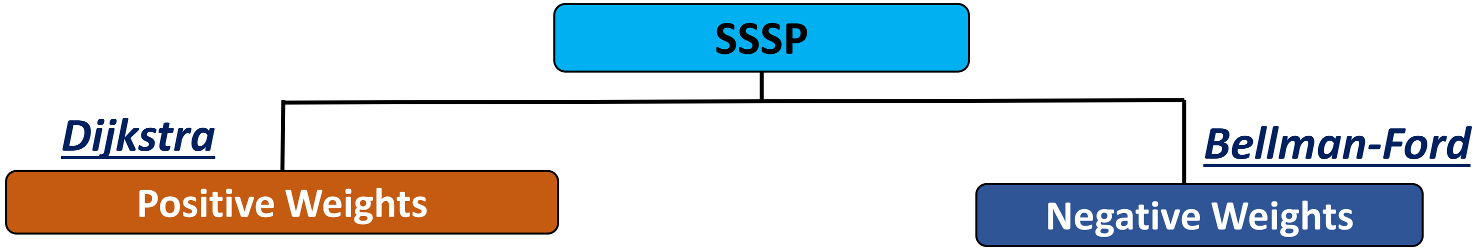


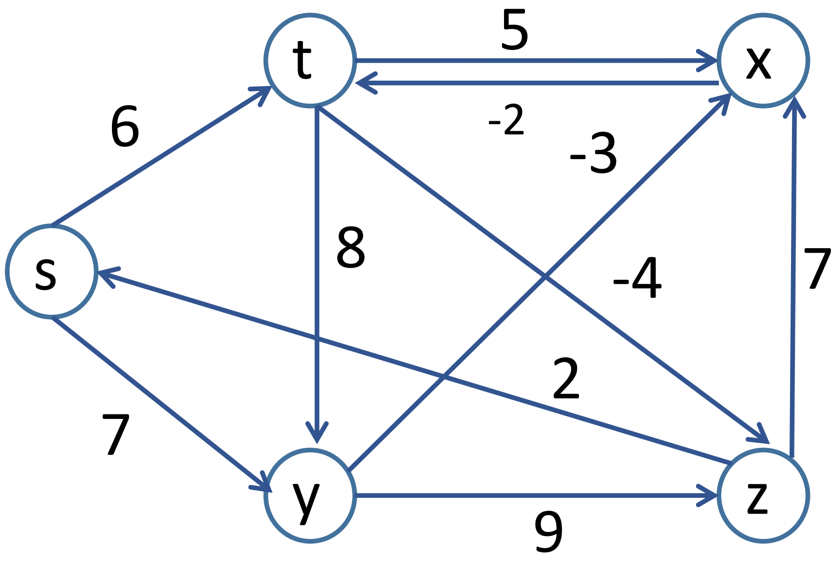
What is the shortest path from S to D?



- Dijkstra doesn't work for **<u>Graphs with negative weight cycle</u>**, Bellman-Ford works for such graphs.
- Bellman-Ford is also simpler than Dijkstra and suites well for distributed systems.
- It does not use Priority Queue

SSSP → Dijkstra





SSSP: Bellman-Ford

1

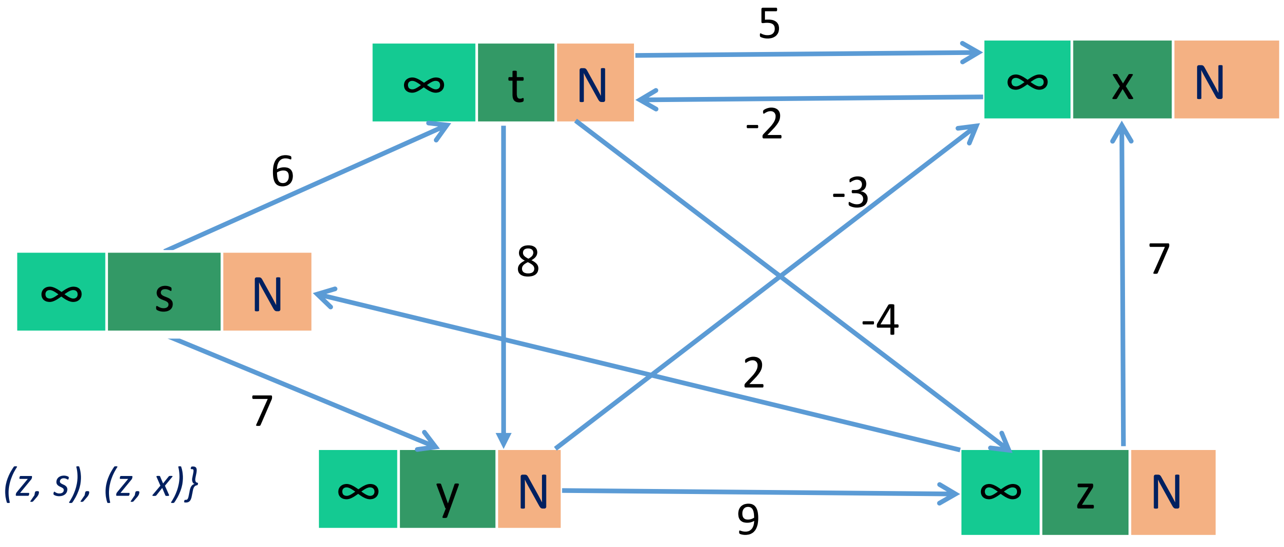
BELLMAN-FORD (G, w, s) {
1 INITIALIZE-SINGLE-SOURCE (G, s)
2 for i = 1 to |G.V| - 1
3 for each edge (u, v) $\in$ G.E
4 RELAX (*u, v, w*)
5 for each edge (u, v) $\in$ G.E
6 if v. d > u. d + w(u, v)
7 return False
8 return True

2

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

3

RELAX (*u, v, w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u



Step 1 : INITIALIZE-SINGLE-SOURCE
*Step 2 - 4: for i = **1** to 5 – 1*
Step 3 : # of edges =
{ (s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)}
Step 4 : RELAX (s, t, w); RELAX (s, y, w)

SSSP: Bellman-Ford (1)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
      for each edge (u, v) ∈ G.E
           RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
           if v. d > u. d + w(u, v)
             return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
    if v. d > u. d + w(u, v)
       v. d =  u. d + w(u, v)
       v. π = u
```

Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***1*** *to 5 – 1*

Step 3 :   # of edges =

{ ***(s, t), (s, y)****, (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)}*

Step 4 :   RELAX (s, t, w);

∞ > 0 + 6;

t. d = 6;

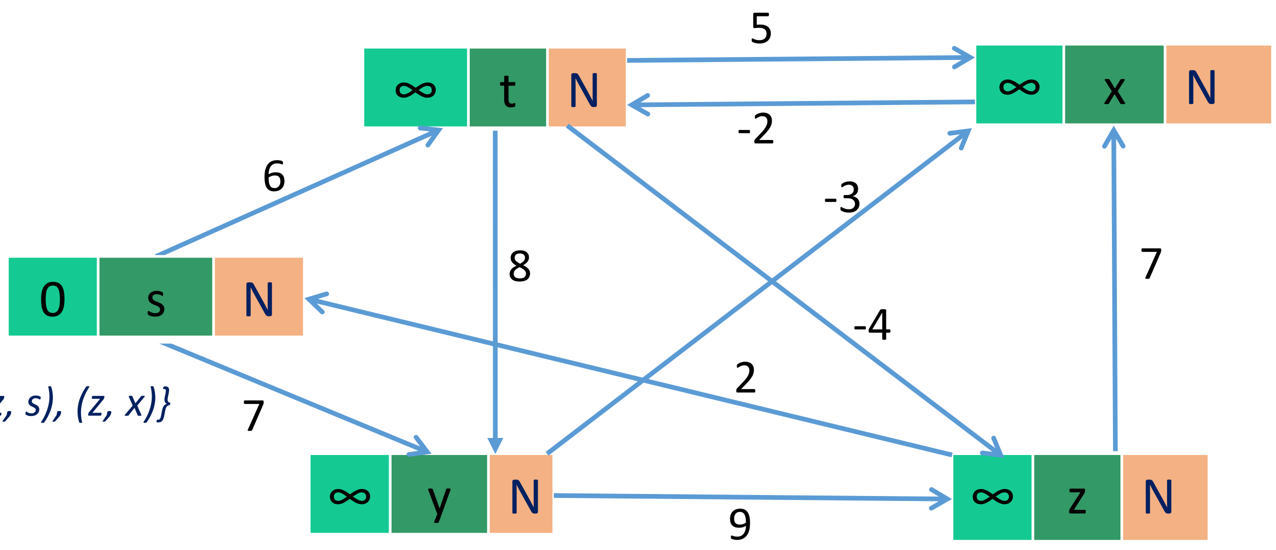
t. π = s;

RELAX (s, y, w)

∞ > 0 + 7;

y. d = 7;

y. π = s;

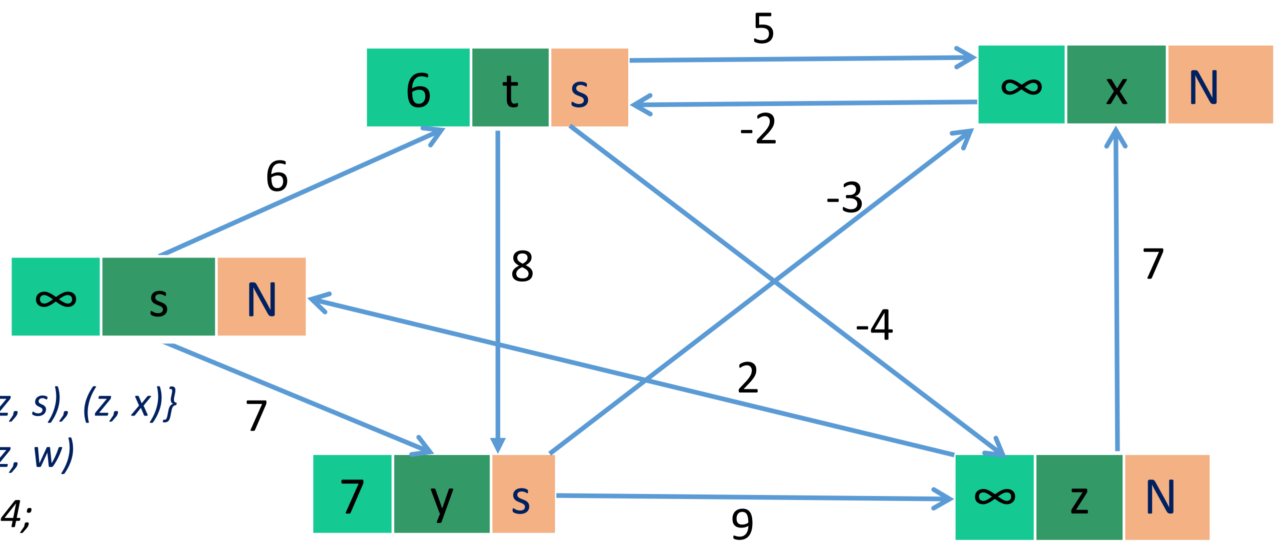


SSSP: Bellman-Ford (2)

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
          return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
   v. d = ∞
   v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
   v. d =  u. d + w(u, v)
   v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***1*** *to 5 – 1*

Step 3 : # of edges =

{ (s, t), (s, y), ***(t, x), (t, y), (t, z)****, (y, x), (y, z), (x, t), (z, s), (z, x)}*

Step 4 : RELAX (t, x, w); — $\infty > 6 + 5$; $x.\,d = 11$; $x.\,\pi = t$;

RELAX (t, y, w); — $7 > 6 + 8$;

RELAX (t, z, w) — $\infty > 6 + -4$; $z.\,d = 2$; $z.\,\pi = t$;

SSSP: Bellman-Ford (3)

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
            return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ∈ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
    v. d =  u. d + w(u, v)
    v. π = u
```

Step 1 : INITIALIZE-SINGLE-SOURCE

*Step 2 : for i = **1** to 5 – 1*

Step 3 : # of edges =

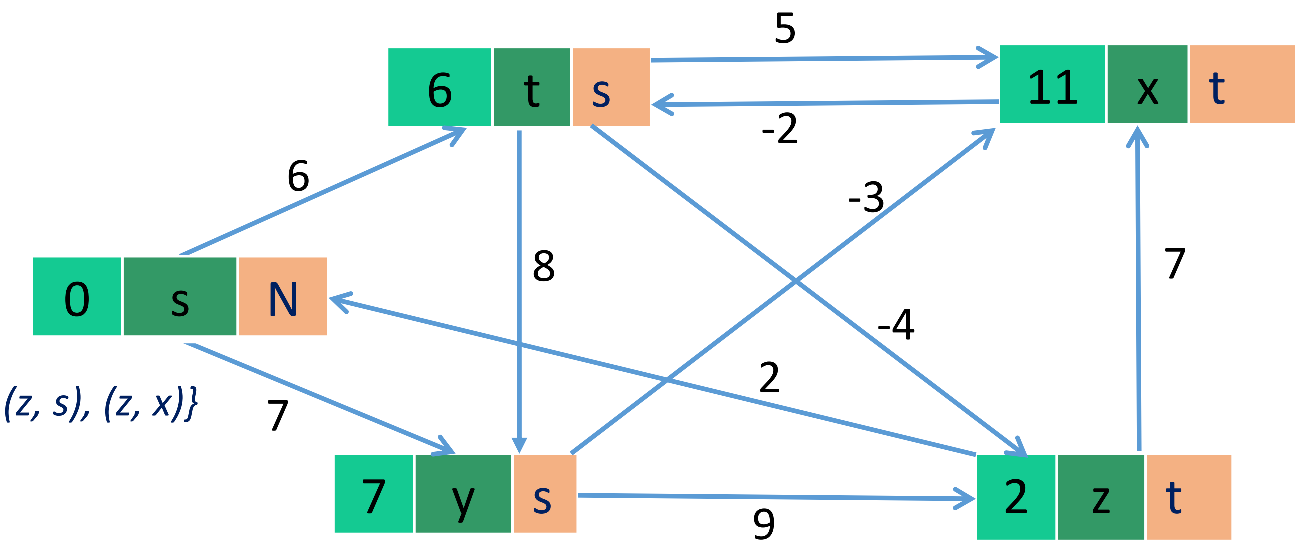
*{ (s, t), (s, y), (t, x), (t, y), (t, z), **(y, x), (y, z)**, (x, t), (z, s), (z, x)}*

Step 4 : RELAX (y, x, w); *RELAX (y, z, w);*

11 > 7 + -3; *2 > 7 + 9;*

x. d = 4;

x. π = y;

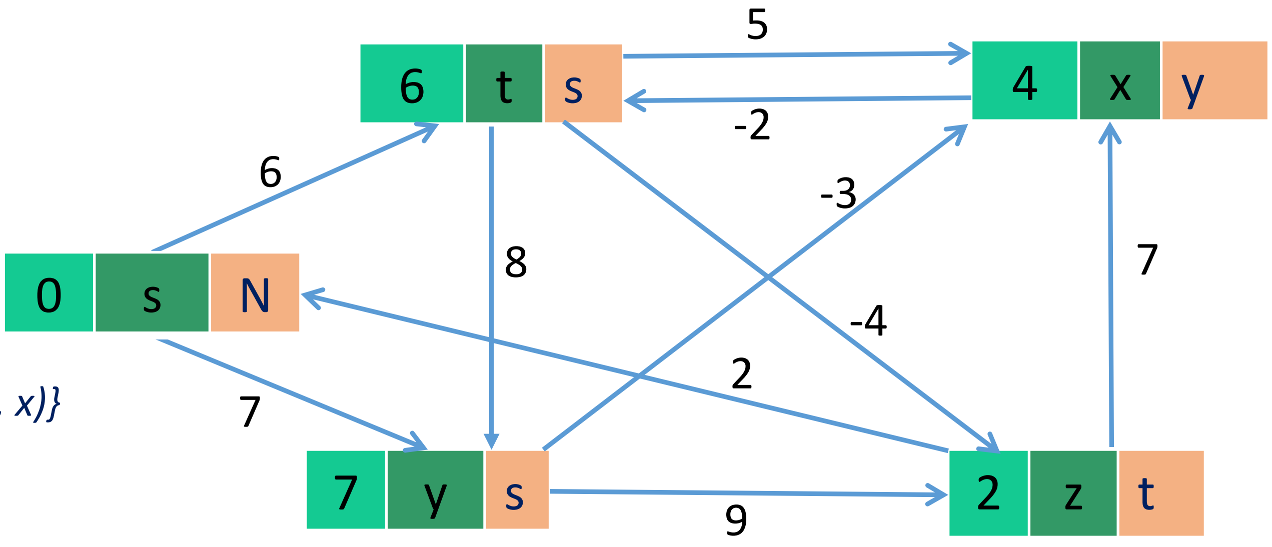


SSSP: Bellman-Ford (4)

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
          return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
   v. d = ∞
   v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
   v. d = u. d + w(u, v)
   v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***1*** *to 5 – 1*

Step 3 : # of edges =

{ (s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), ***(x, t)****, (z, s), (z, x)}*

Step 4 : RELAX (x, t, w);

6 > 4 + -2;

t. d = 2;

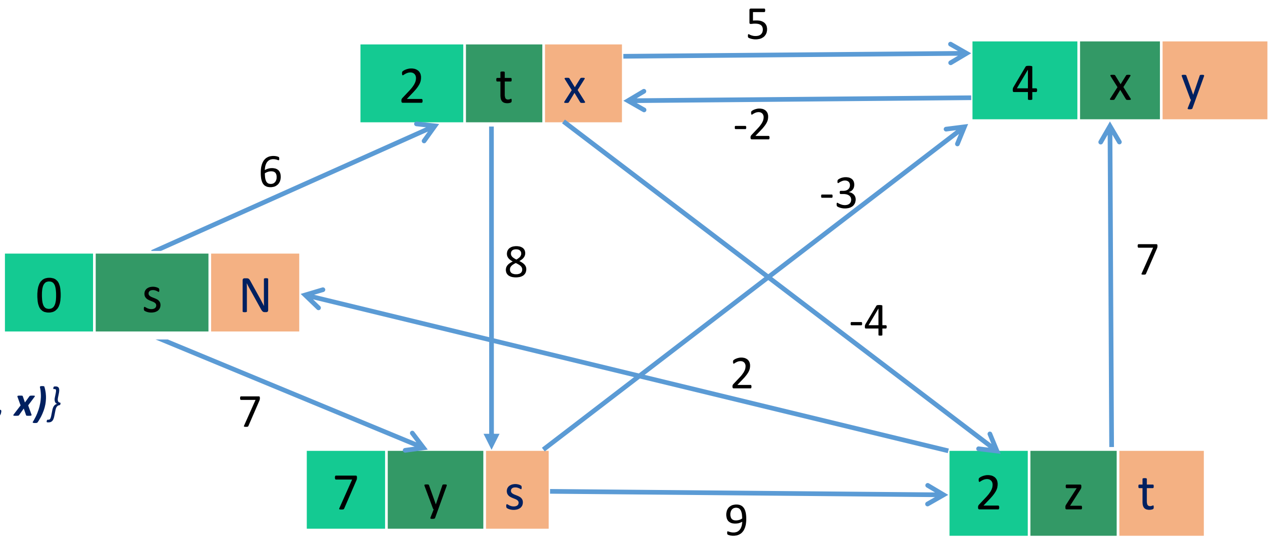
t. π = x;

SSSP: Bellman-Ford (5)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
     for each edge (u, v) ∈ G.E
         RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
         if v. d > u. d + w(u, v)
           return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
   v. d = ∞
   v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
   if v. d > u. d + w(u, v)
      v. d =  u. d + w(u, v)
      v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***1*** *to 5 – 1*

Step 3 : # of edges =

{ (s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), ***(z, s), (z, x)****}*

Step 4 : RELAX (z, s, w); *RELAX (z, x, w)*

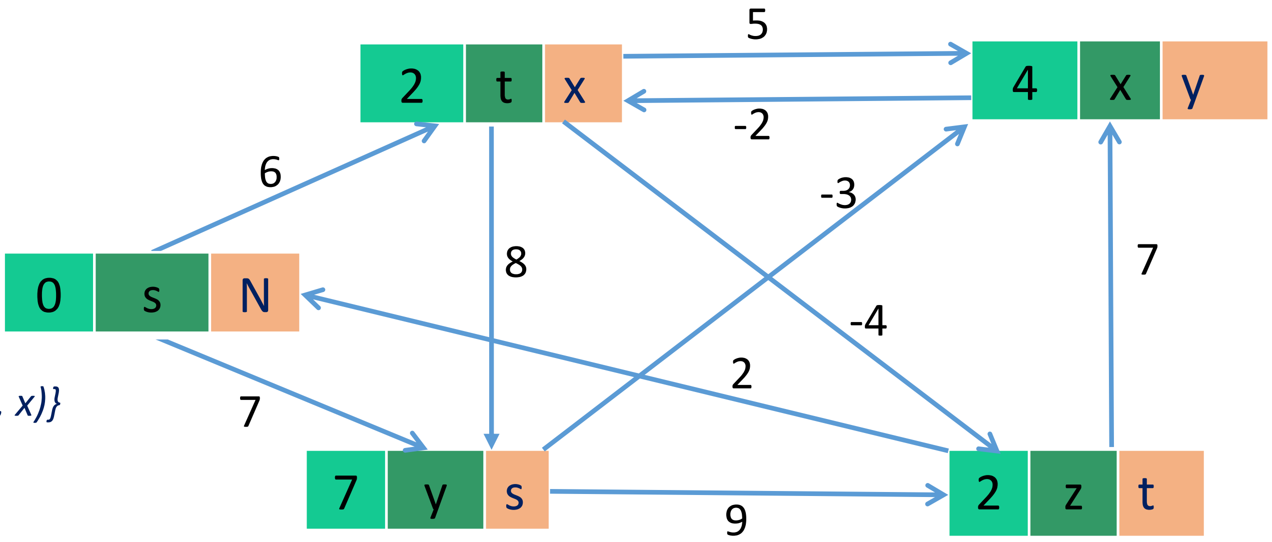
0 > 2 + 2; *4 > 2 + 7;*

SSSP: Bellman-Ford (6)

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) $\in$ G.E
4   RELAX (*u, v, w*)
5 for each edge (u, v) $\in$ G.E
6   if v. d > u. d + w(u, v)
7   return False
8 return True

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2   *v. d* = ∞
3   *v. π* = NIL
4 *s. d* = 0

RELAX (*u, v, w*) {
1   if v. d > u. d + w(u, v)
2   v. *d* = u. d + w(u, v)
3   *v. π* = u



Step 1 : INITIALIZE-SINGLE-SOURCE
Step 2 : for i = ***2***
Step 3 :   # of edges =
{ ***(s, t), (s, y)****, (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)}*
Step 4 :   RELAX (s, t, w);   RELAX (s, y, w)
2 > 0 + 6;   7 > 0 + 7;

SSSP: Bellman-Ford (7)

BELLMAN-FORD (G, w, s) {

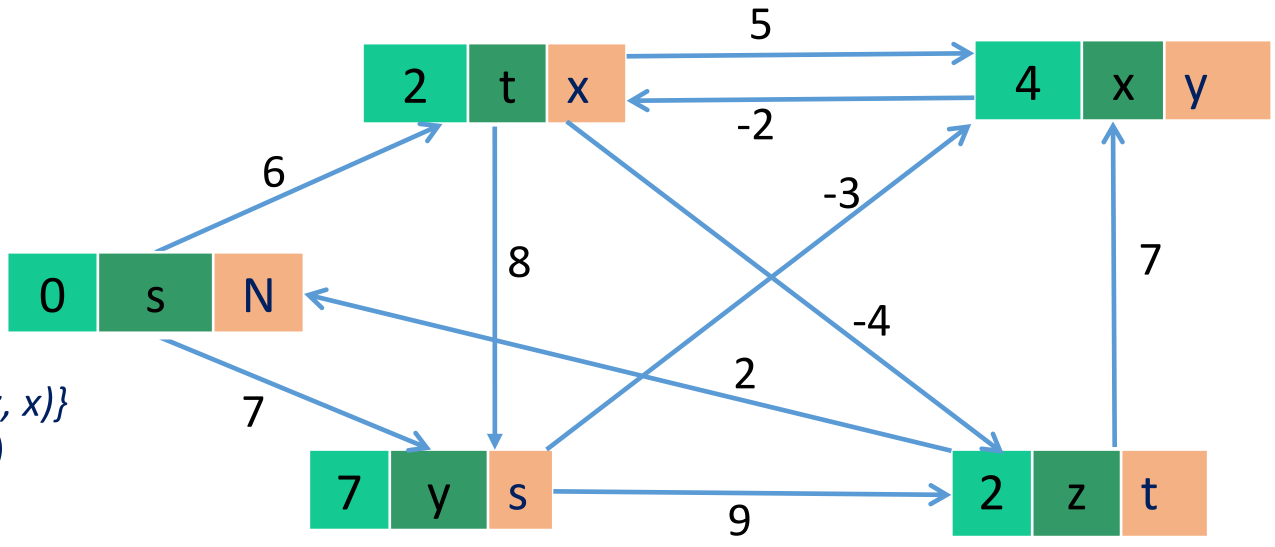
```
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
            return False
return True
```

INITIALIZE-SINGLE-SOURCE (G, s) {

```
for each v ∈ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

RELAX (*u, v, w*) {

```
if v. d > u. d + w(u, v)
    v. d = u. d + w(u, v)
    v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***2***

Step 3 : # of edges =

{ (s, t), (s, y), ***(t, x), (t, y), (t, z)****, (y, x), (y, z), (x, t), (z, s), (z, x)}*

Step 4 : RELAX (t, x, w); RELAX (t, y, w); RELAX (t, z, w)

4 > 2 +5;

7 > 2 + 8;

2 > 2 + -4;

z. d = -2

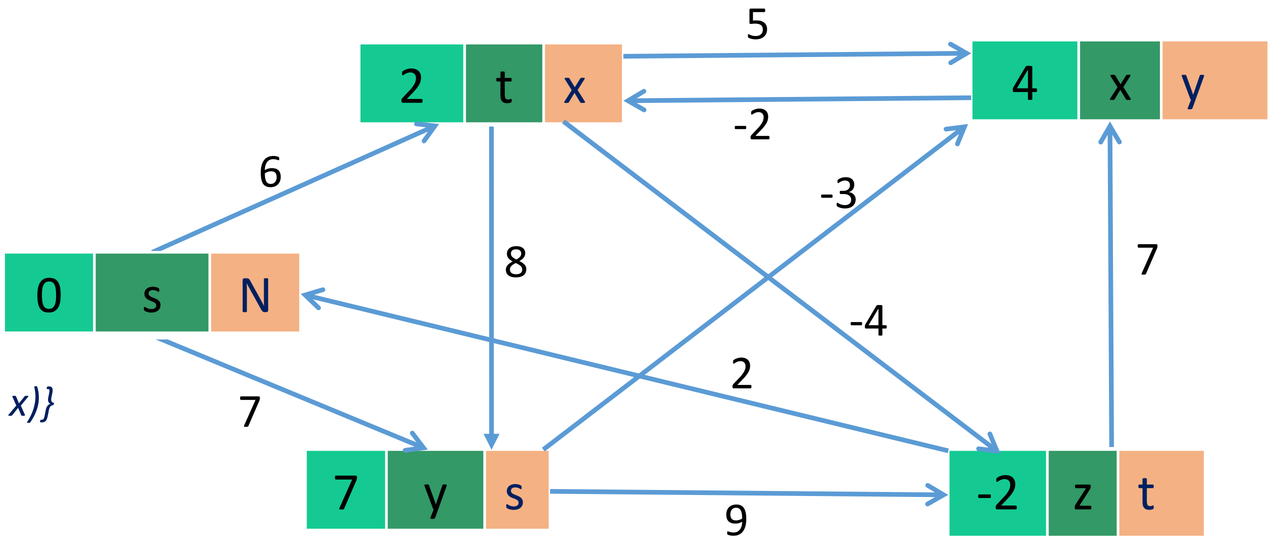
z. π = t

SSSP: Bellman-Ford (8)

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
          return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
    if v. d > u. d + w(u, v)
      v. d =  u. d + w(u, v)
      v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***2***

Step 3 : # of edges =

{ (s, t), (s, y), (t, x), (t, y), (t, z), ***(y, x), (y, z)****, (x, t), (z, s), (z, x)}*

Step 4 : RELAX (y, x, w); *RELAX (y, z, w);*

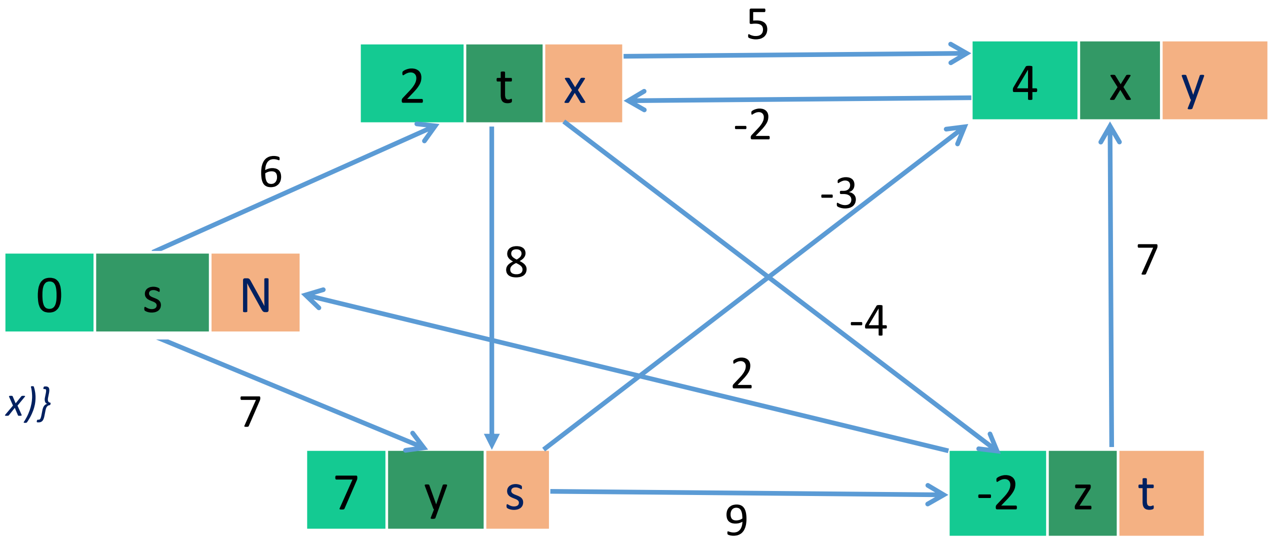
4 > 7 + -3; *- 2 > 7 + 9;*

SSSP: Bellman-Ford (9)

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
          return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
  v. d = ∞
  v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
   v. d =  u. d + w(u, v)
   v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***2***

Step 3 :   # of edges =

{ (s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), ***(x, t)****, (z, s), (z, x)}*

Step 4 :   RELAX (x, t, w);

2 > 4 + -2;

SSSP: Bellman-Ford (10)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
       for each edge (u, v) ∈ G.E
             RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
             if v. d > u. d + w(u, v)
               return False
 return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
 for each v ∈ G.V
     v. d = ∞
     v. π = NIL
 s. d = 0
```

```
RELAX (u, v, w) {
    if v. d > u. d + w(u, v)
       v. d =  u. d + w(u, v)
       v. π = u
```

Step 1 : INITIALIZE-SINGLE-SOURCE

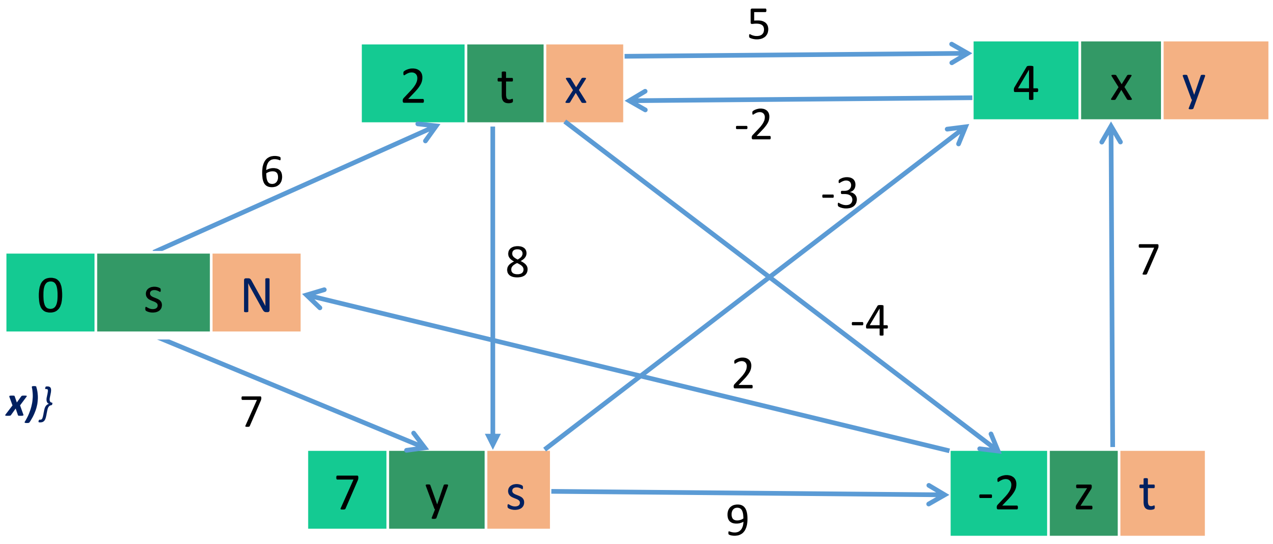
Step 2 : for i = ***2***

Step 3 : # of edges =

{ (s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), ***(z, s), (z, x)****}*

Step 4 : RELAX (z, s, w); RELAX (z, x, w);

0 > -2 + 2; 4 > -2 + 7;

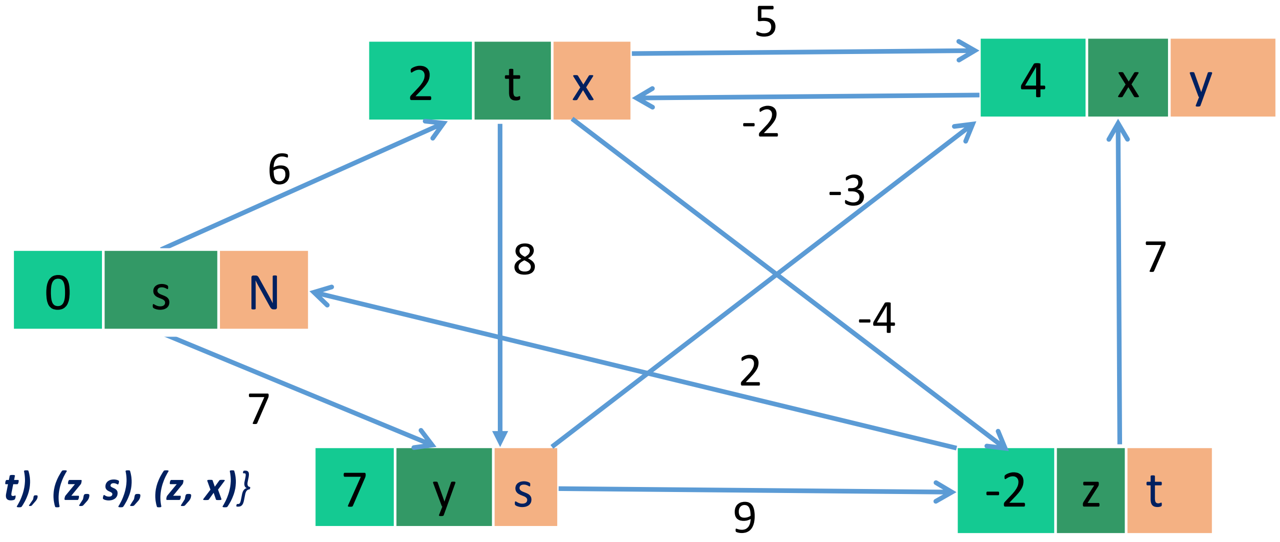


SSSP: Bellman-Ford (11)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
     for each edge (u, v) ∈ G.E
         RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
         if v. d > u. d + w(u, v)
           return False
 return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
 for each v ∈ G.V
   v. d = ∞
   v. π = NIL
 s. d = 0
```

```
RELAX (u, v, w) {
  if v. d > u. d + w(u, v)
     v. d =  u. d + w(u, v)
     v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

*Step 2 : for i = **3***

*Step 3 : # of edges = { **(s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)**}*

Step 4 : RELAX (s, t, w); RELAX (s, y, w); RELAX (t, x, w); RELAX (t, y, w); RELAX (t, z, w); RELAX (y, x, w); RELAX (y, z, w); RELAX (x, t, w);

2 > 0+ 6; 7 > 0 + 7; 4 > 2 + 5; 7 > 2 + 8; -2 > 2 + -4; 4 > 7 + -3 ; -2 > 7 + 9; 2 > 4 + -2

RELAX (z, s, w); RELAX (z, x, w);

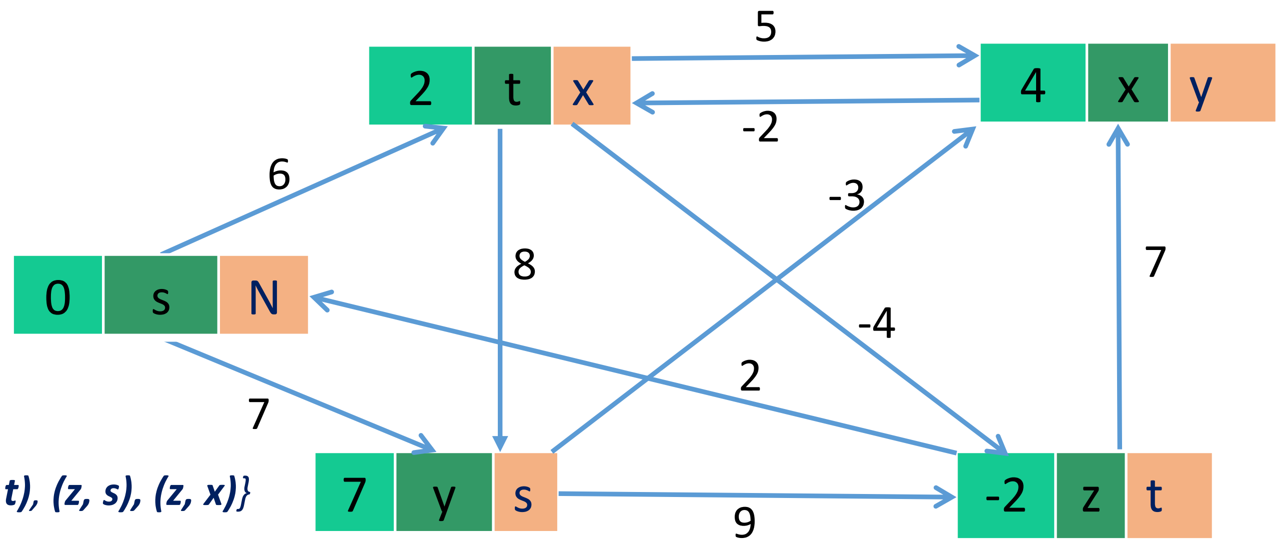
0 > -2 + 2; 4 > -2 + 7 ;

SSSP: Bellman-Ford (12)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
      for each edge (u, v) ∈ G.E
            RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
            if v. d > u. d + w(u, v)
              return False
return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ∈ G.V
   v. d = ∞
   v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
   if v. d > u. d + w(u, v)
     v. d =  u. d + w(u, v)
     v. π = u
```



Step 1 : INITIALIZE-SINGLE-SOURCE

Step 2 : for i = ***4***

Step 3 : # of edges = { ***(s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)****}*

Step 4 : RELAX (s, t, w); RELAX (s, y, w); RELAX (t, x, w); RELAX (t, y, w); RELAX (t, z, w); RELAX (y, x, w); RELAX (y, z, w); RELAX (x, t, w);

2 > 0+ 6; 7 > 0 + 7; 4 > 2 + 5; 7 > 2 + 8; -2 > 2 + -4; 4 > 7 + -3 ; -2 > 7 + 9; 2 > 4 + -2

RELAX (z, s, w); RELAX (z, x, w);

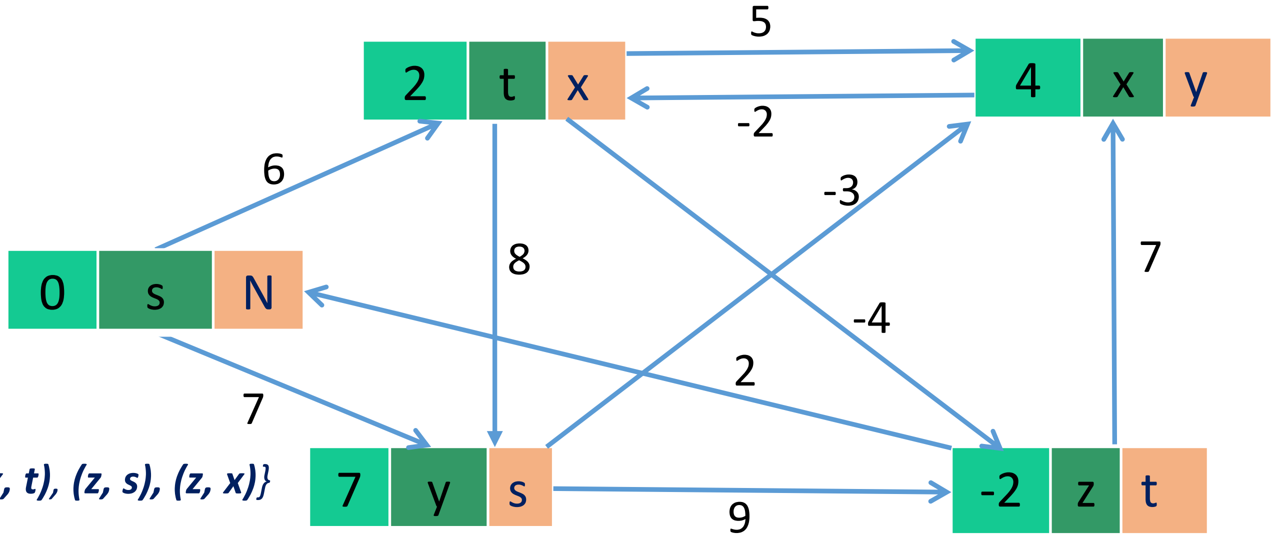
0 > -2 + 2; 4 > -2 + 7 ;

SSSP: Bellman-Ford (13)

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) $\in$ G.E
4 RELAX (*u, v, w*)
5 for each edge (u, v) $\in$ G.E
6 if v. d > u. d + w(u, v)
7 return False
8 return True

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

RELAX (*u, v, w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u



Step 5 : # of edges = { ***(s, t), (s, y), (t, x), (t, y), (t, z), (y, x), (y, z), (x, t), (z, s), (z, x)***}

Step 6-7: RELAX (s, t, w); RELAX (s, y, w); RELAX (t, x, w); RELAX (t, y, w); RELAX (t, z, w); RELAX (y, x, w); RELAX (y, z, w); RELAX (x, t, w);
2 > 0+ 6; 7 > 0 + 7; 4 > 2 + 5; 7 > 2 + 8; -2 > 2 + -4; 4 > 7 + -3 ; -2 > 7 + 9; 2 > 4 + -2
RELAX (z, s, w); RELAX (z, x, w);
0 > -2 + 2; 4 > -2 + 7 ;

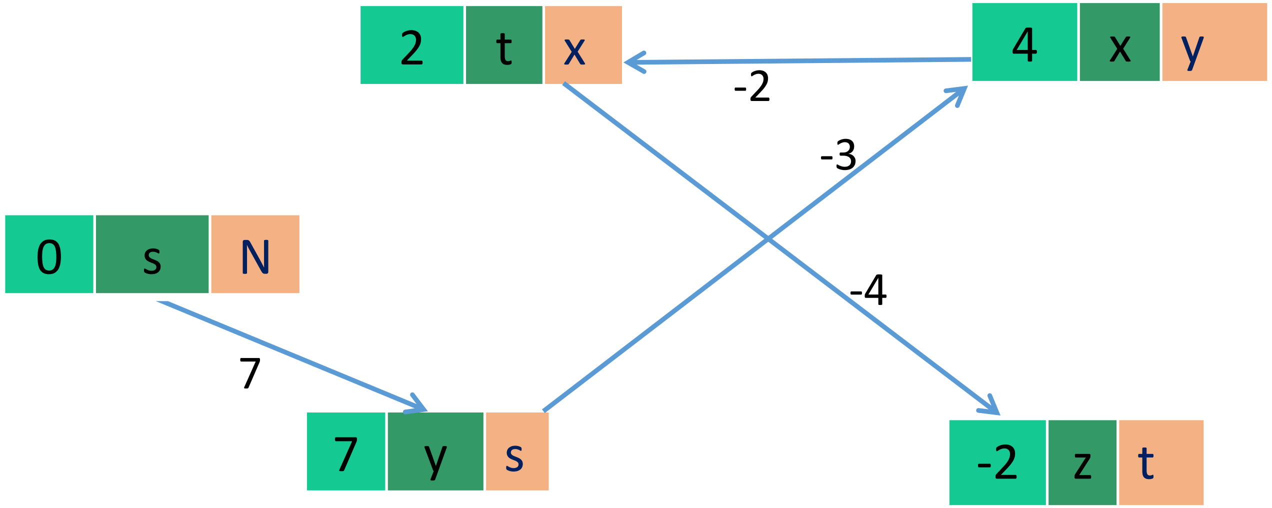
Step 8: return

SSSP: Bellman-Ford (14)

```
BELLMAN-FORD (G, w, s) {
 INITIALIZE-SINGLE-SOURCE (G, s)
 for i = 1 to |G.V| - 1
     for each edge (u, v) ∈ G.E
         RELAX (u, v, w)
 for each edge (u, v) ∈ G.E
         if v. d > u. d + w(u, v)
           return False
 return True
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
 for each v ∈ G.V
   v. d = ∞
   v. π = NIL
 s. d = 0
```

```
RELAX (u, v, w) {
   if v. d > u. d + w(u, v)
     v. d =  u. d + w(u, v)
     v. π = u
```



SSSP: Bellman-Ford → Time complexity analysis

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
          return False
return True
```

# of vertices → O (V)

Outer loop → O (V); Inner loop → O (E) → O (VE)

O (E)

Total time complexity: <u>O(V) + O (VE) + O(E)</u>

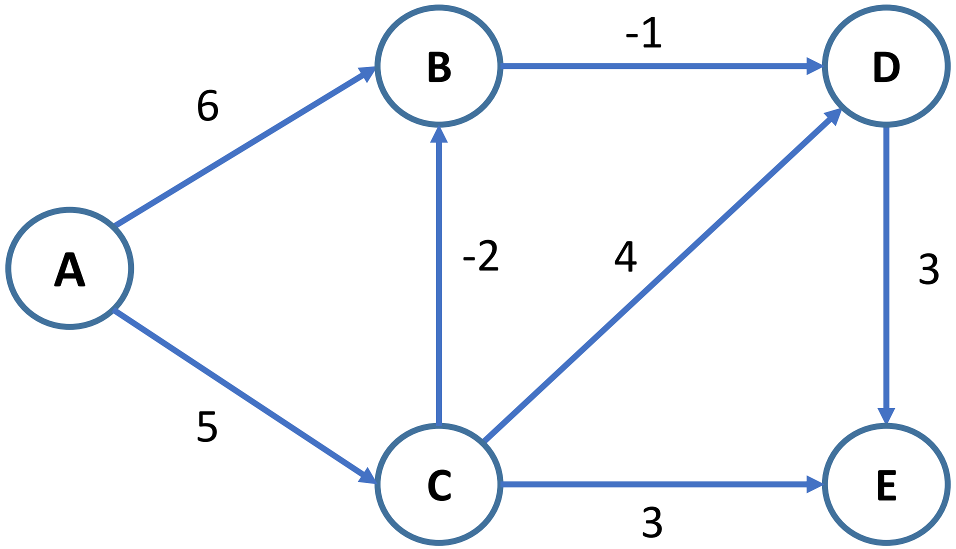
```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
   v. d = ∞
   v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
    v. d =  u. d + w(u, v)
    v. π = u
```

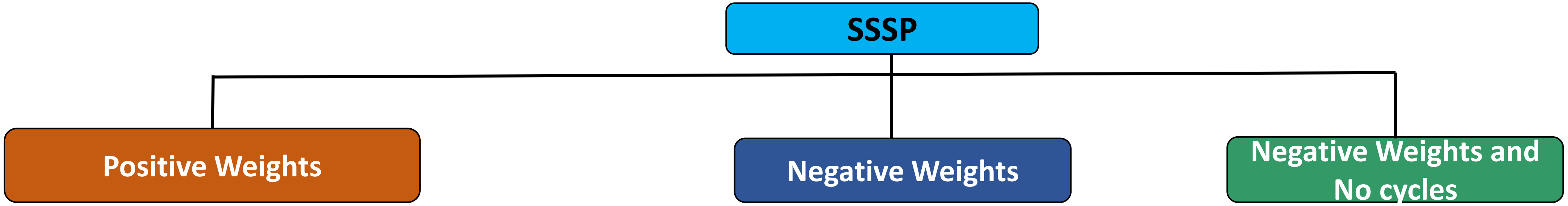
Exercise: Bellman-Ford

Consider the following digraph starting at vertex A and apply Bellman-Ford single source shortest path algorithm on it.

- What is the minimum distance between vertex A and E
  a. 3
  b. 2
  c. 1
  d. 5



Recap: SSSP



<u>Dijkstra</u> | ***<u>Bellman-Ford</u>*** | ***<u>Directed Acyclic Graph (DAG)</u>***

```
DIJKSTRA (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
S = ∅
Q = G. V
while Q ≠ ∅ ;
  u = EXTRACT-MIN(Q)
  S = S ∪ {u}
  for each vertex v ∊ Q. Adj[u]
    RELEAX (u, v, w)
}
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ∊ G.V
  v. d = ∞
  v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
  if v.d > u.d + w(u, v)
    v. d =  u.d + w(u, v)
    v. π = u
```

```
BELLMAN-FORD (G, w, s) {
INITIALIZE-SINGLE-SOURCE (G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX (u, v, w)
for each edge (u, v) ∈ G.E
        if v. d > u. d + w(u, v)
            return False
return True
```

Total time complexity: <u>O(V) + O (V log V) + O(V) + O(E logV)</u>
: O(ElogV)

Total time complexity: <u>O(V) + O (VE) + O(E)</u>
: O(VE)

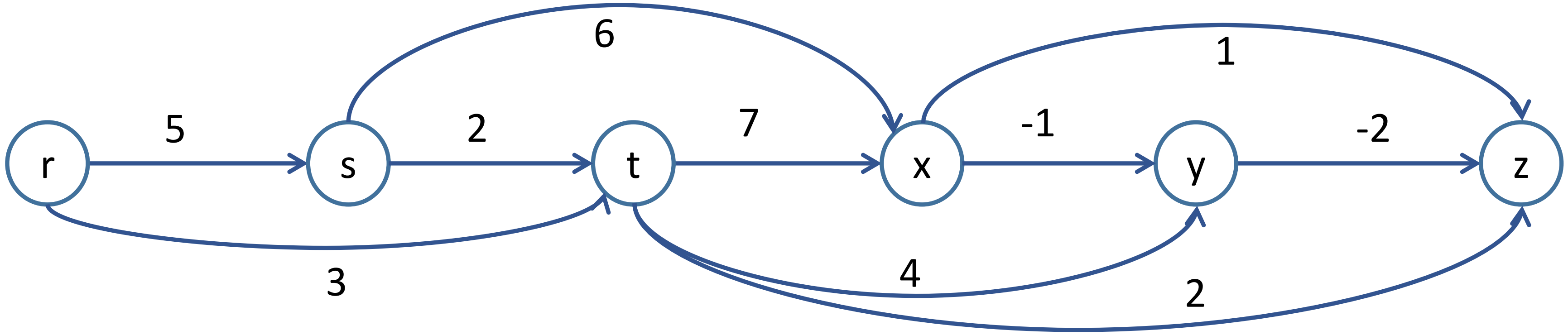
SSP: Directed Acyclic Graph (DAG)

- Directed Graph
- No cycles
- Topological sort can be applied on any DAG → O (V+E)
- Works for negative edges

```
DAG-SHORTEST-PATH (G, w, s) {
Topological sort the vertices of G
INITIALIZE-SINGLE-SOURCE (G, s)
for each vertex u, taken in topologically sorted order
    for each edge v ∈ G. Adj [u]
        RELAX (u, v, w)
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ϵ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
    v. d =  u. d + w(u, v)
    v. π = u
```



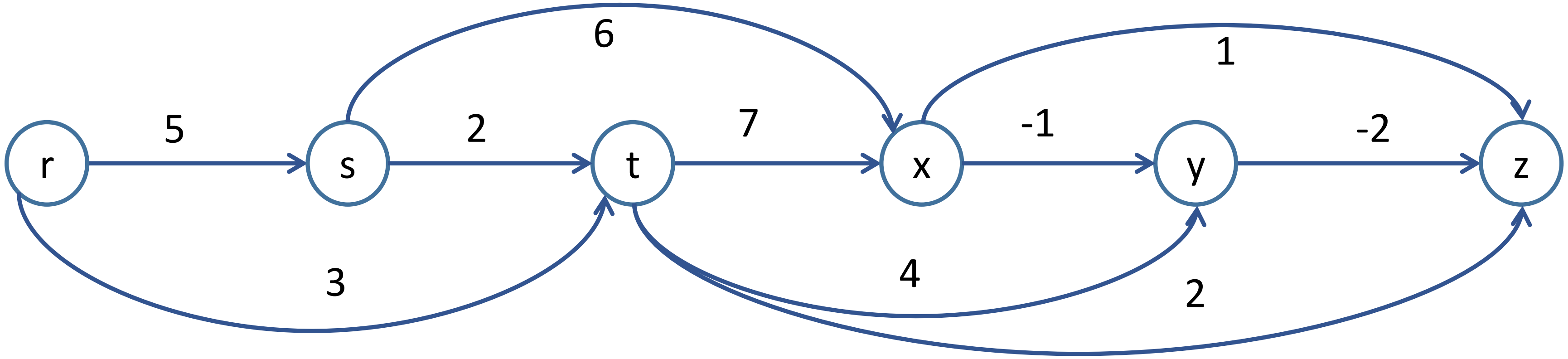
SSSP: DAG (1)

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

```
INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each v ∈ G.V
2    v. d = ∞
3    v. π = NIL
4 s. d = 0
```

```
RELAX (u, v, w) {
1    if v. d > u. d + w(u, v)
2        v. d =  u. d + w(u, v)
3        v. π = u
```

Step 1: Topological order

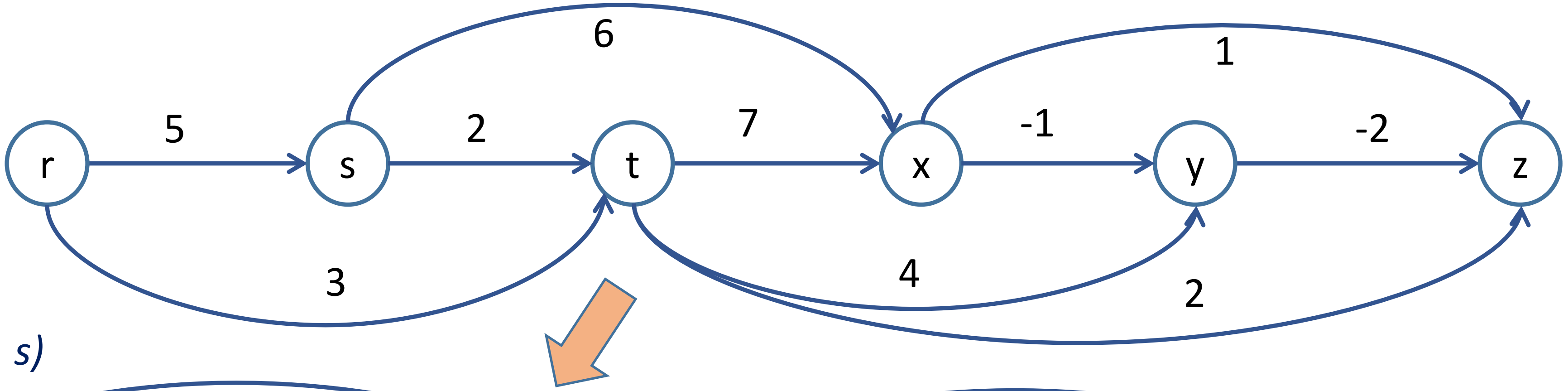


SSSP: DAG (1)

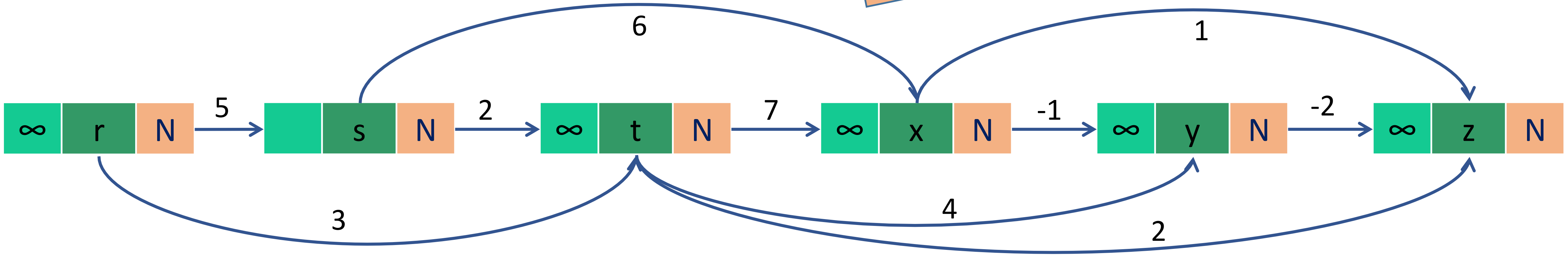
DAG-SHORTEST-PATH (G, w, s) {
Topological sort the vertices of G
INITIALIZE-SINGLE-SOURCE (G, s)
for each vertex *u,* taken in topologically sorted order
    for each edge $v \in$ G. *Adj* [*u*]
        RELAX (*u, v, w*)

INITIALIZE-SINGLE-SOURCE (G, s) {
for each $v \in$ G.*V*
    v. d = ∞
    v. π = NIL
s. d = 0

RELAX (*u, v, w*) {
if v. d > u. d + w(u, v)
    v. *d* = u. d + w(u, v)
    v. π = u



Step 1: Topological order
Step 2: INITIALIZE-SINGLE-SOURCE (G, s)



SSSP: DAG (2)

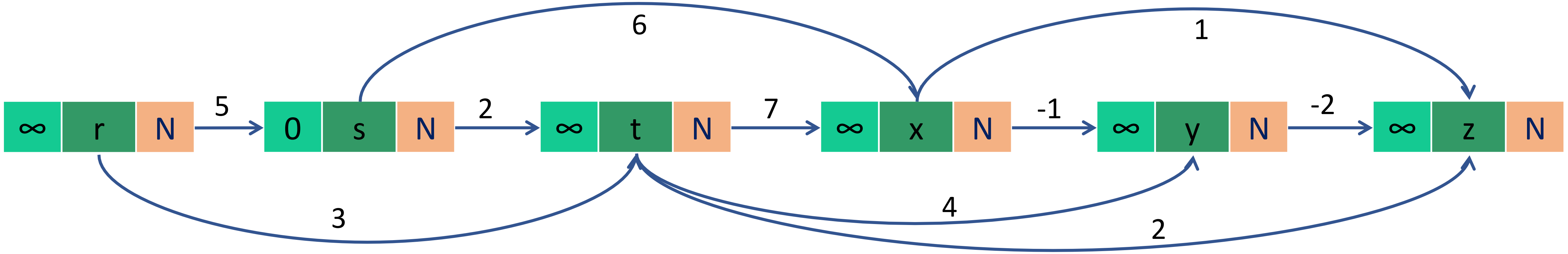
DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u*, taken in topologically sorted order
4 for each edge *v* ∈ G. *Adj* [*u*]
5 RELAX (*u, v, w*)

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

RELAX (*u, v, w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u

Step 1: Topological order
Step 2: INITIALIZE-SINGLE-SOURCE (G, s)
Step 3: u = {r, s, t, x, y, z}
u = r
Step 4: v = G. Adj [r]
= {s, t}

Step 5: RELAX (r, s, w); RELAX (r, t, w)
0 > ∞ + 5; ∞ > ∞ +3;



SSSP: DAG (3)

DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u*, taken in topologically sorted order
4 for each edge $v \in$ G. *Adj* [*u*]
5 RELAX (*u*, *v*, *w*)

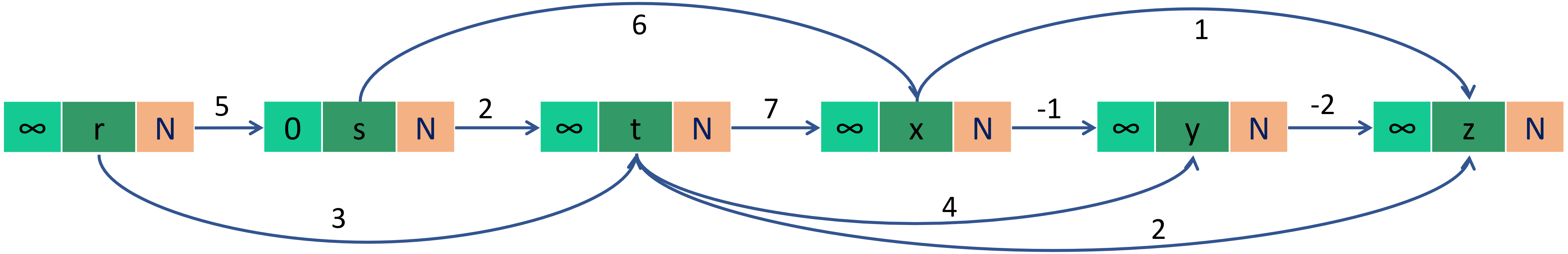
INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each $v \in$ G.*V*
2 *v*. *d* = ∞
3 *v*. *π* = NIL
4 *s*. *d* = 0

RELAX (*u*, *v*, *w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v*. *π* = u

Step 3: u = {r, s, t, x, y, z}
u = s
Step 4: v = G. Adj [s]
= {t, x}

Step 5: RELAX (s, t, w); RELAX (s, x, w)

∞ > 0 + 2;	*∞ > 0 + 6;*
t. d = 2	*x. d* = 6
t. π = s	*x. π* = s



SSSP: DAG (4)

DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u*, taken in topologically sorted order
4 for each edge $v \in$ G. *Adj* [*u*]
5 RELAX (*u, v, w*)

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each $v \in$ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

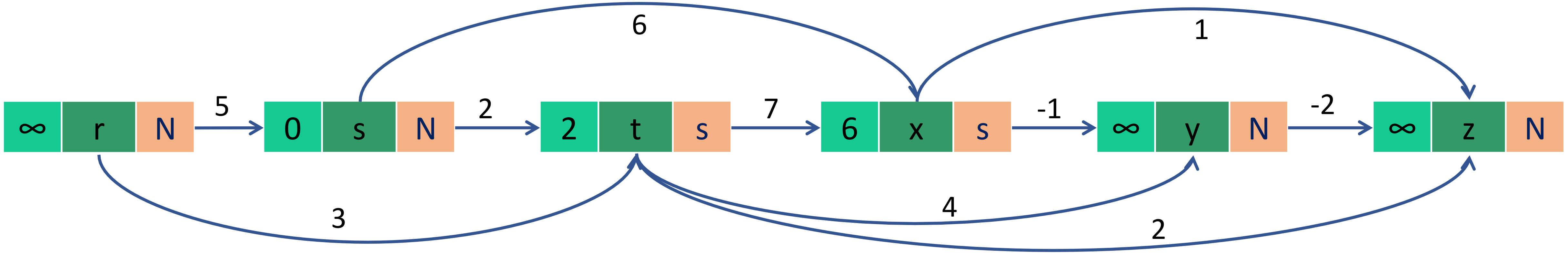
RELAX (*u, v, w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u

Step 3: u = {r, s, t, x, y, z}
u = t
Step 4: v = G. Adj [t]
= {x, y, z}

Step 5: RELAX (t, x, w);
6 > 2 + 7

RELAX (t, y, w);
∞ > 2 + 4
y. d = 6
y. π = t

RELAX (t, z, w)
∞ > 2 + 2
z. d = 4
z. π = t



SSSP: DAG (5)

DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u*, taken in topologically sorted order
4 for each edge $v \in$ G. *Adj* [*u*]
5 RELAX (*u*, *v*, *w*)

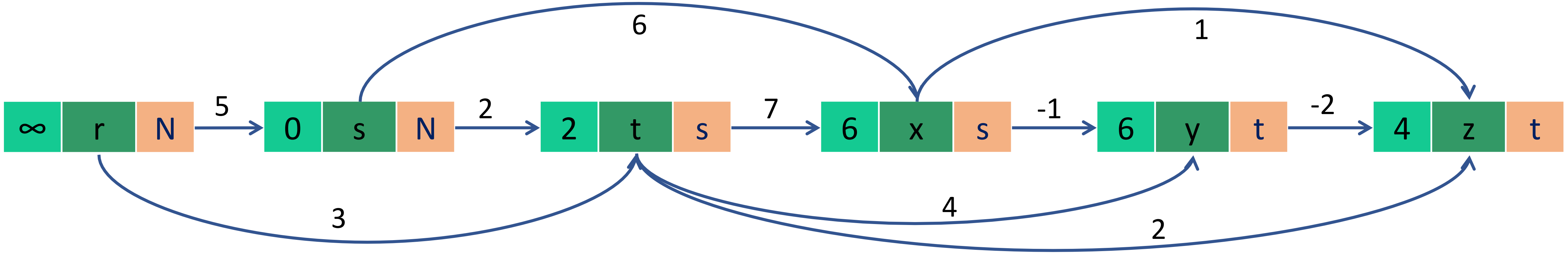
INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

RELAX (*u*, *v*, *w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u

Step 3: u = {r, s, t, x, y, z}
u = x
Step 4: v = G. Adj [x]
= {y, z}

Step 5: RELAX (x, y, w);
6 > 6 + -1
y. d = 5
y. π = x

RELAX (x, z, w)
4 > 6 + 1



SSSP: DAG (6)

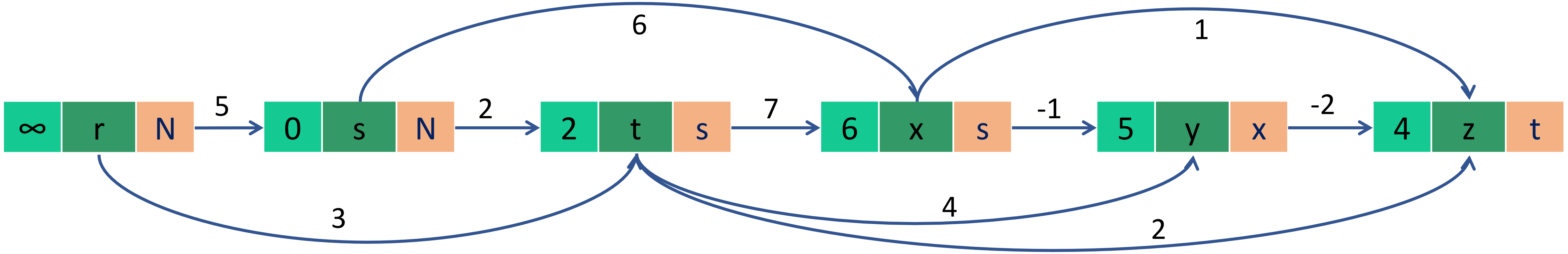
DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u*, taken in topologically sorted order
4 for each edge $v \in$ G. *Adj* [*u*]
5 RELAX (*u*, *v*, *w*)

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each $v \in$ G.V
2 $v.\ d = \infty$
3 $v.\ \pi$ = NIL
4 $s.\ d = 0$

RELAX (*u*, *v*, *w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 $v.\ \pi$ = u

Step 3: u = {r, s, t, x, y, z}
u = y
Step 4: v = G. Adj [y]
= {z}

Step 5: RELAX (y, z, w);
4 > 5 + -2
z. d = 3
$z.\ \pi$ = y



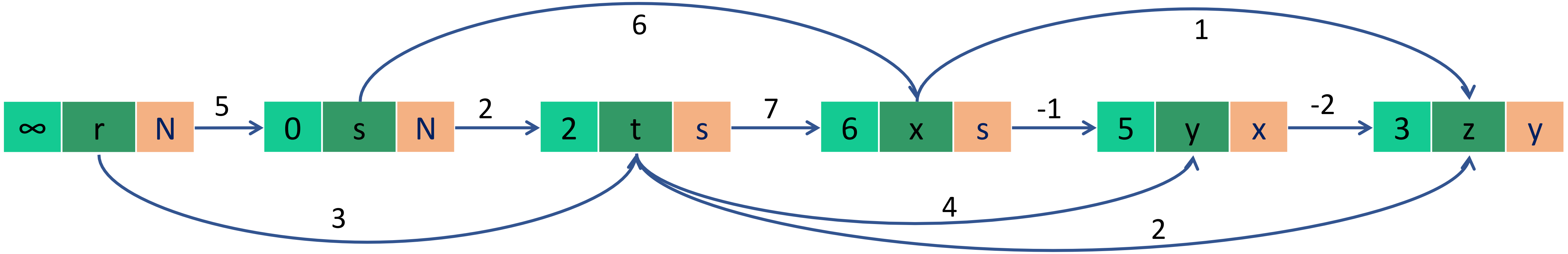
SSSP: DAG (7)

DAG-SHORTEST-PATH (G, w, s) {
1 Topological sort the vertices of G
2 INITIALIZE-SINGLE-SOURCE (G, s)
3 for each vertex *u,* taken in topologically sorted order
4 for each edge $v \in$ G. *Adj* [*u*]
5 RELAX (*u, v, w*)

INITIALIZE-SINGLE-SOURCE (G, s) {
1 for each *v* ϵ G.*V*
2 *v. d* = ∞
3 *v. π* = NIL
4 *s. d* = 0

RELAX (*u, v, w*) {
1 if v. d > u. d + w(u, v)
2 v. *d* = u. d + w(u, v)
3 *v. π* = u

Step 3: u = {r, s, t, x, y, z}
u = z
Step 4: v = G. Adj [z]
= { }

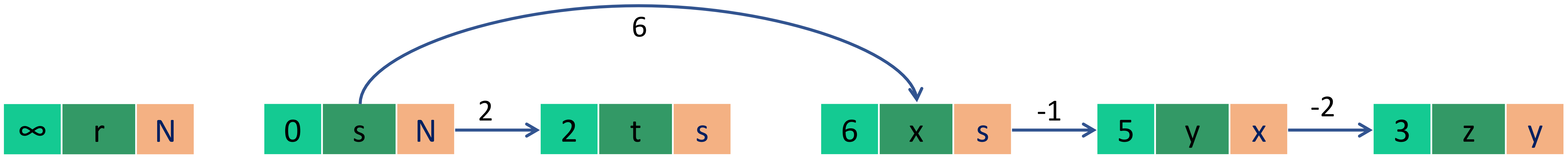


SSSP: DAG (8)

```
DAG-SHORTEST-PATH (G, w, s) {
Topological sort the vertices of G
INITIALIZE-SINGLE-SOURCE (G, s)
for each vertex u, taken in topologically sorted order
    for each edge v ∈ G. Adj [u]
        RELAX (u, v, w)
```

```
INITIALIZE-SINGLE-SOURCE (G, s) {
for each v ∈ G.V
    v. d = ∞
    v. π = NIL
s. d = 0
```

```
RELAX (u, v, w) {
if v. d > u. d + w(u, v)
    v. d =  u. d + w(u, v)
    v. π = u
```



SSSP: DAG → Time Complexity Analysis

DAG-SHORETEST-PATH (G, w, s) {

Topological sort the vertices of G — **O (V + E)**

INITIALIZE-SINGLE-SOURCE (G, s) — **O (V)**

for each vertex *u*, taken in topologically sorted order
    for each edge $v \in$ G. *Adj* [*u*]
        RELAX (*u, v, w*)

Outer loop → O (V); Inner loop → O (E)

Total time complexity: <u>O (V+E)</u>

INITIALIZE-SINGLE-SOURCE (G, s) {

for each $v \in G.V$
    $v.\ d = \infty$
    $v.\ \pi$ = NIL
$s.\ d = 0$

RELAX (*u, v, w*) {

if v. d > u. d + w(u, v)
    $v.\ d =$ u. d + w(u, v)
    $v.\ \pi$ = u

thank you!

email:
k.kondepu@iitdh.ac.in