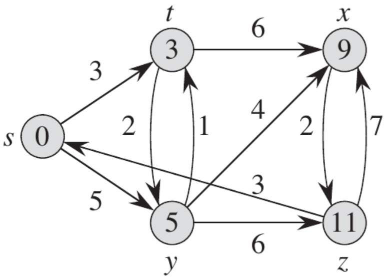
Data Structures and Algorithms [1]

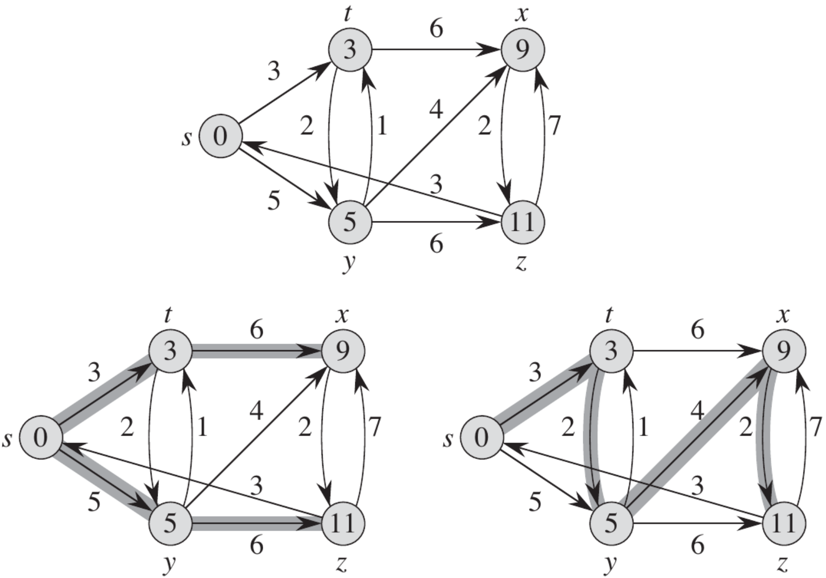
BITS-Pilani K. K. Birla Goa Campus

[1] Material for the presentation taken from Cormen, Leiserson, Rivest and Stein, *Introduction to Algorithms, Fourth Edition*;

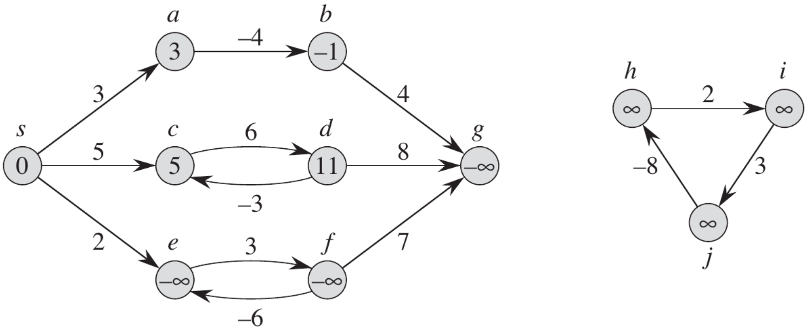
Ch. 22: Single-Source Shortest Paths



Ch. 22: Single-Source Shortest Paths



Effect of a negative weight cycle



Shortest path

$v_0 \quad v_1 \quad \ldots \quad v_i \quad v_c \quad \ldots \quad v_i \quad v_{i+1} \quad \ldots \quad v_j$

Shortest path

$v_0 \quad v_1 \quad \dots \quad v_i \quad v_c \quad \dots \quad v_i \quad v_{i+1} \quad \dots \quad v_j$

- No vertex will occur more than once in any shortest path if there is no negative weight cycle.

Shortest path

$v_0 \quad v_1 \quad \dots \quad v_i \quad v_c \quad \dots \quad v_i \quad v_{i+1} \quad \dots \quad v_j$

- No vertex will occur more than once in any shortest path if there is no negative weight cycle.
- Maximum number of edges in the shortest path will be

Shortest path

$v_0 \quad v_1 \quad \dots \quad v_i \quad v_c \quad \dots \quad v_i \quad v_{i+1} \quad \dots \quad v_j$

- No vertex will occur more than once in any shortest path if there is no negative weight cycle.
- Maximum number of edges in the shortest path will be $(V-1)$.

Shortest path

$v_0 \quad v_1 \quad \ldots \quad v_{i-1} \quad v_i \quad v_{i+1} \quad \ldots \quad v_j$

Shortest path

$v_0 \quad v_1 \quad \ldots \quad v_{i-1} \quad v_i \quad v_{i+1} \quad \ldots \quad v_j$

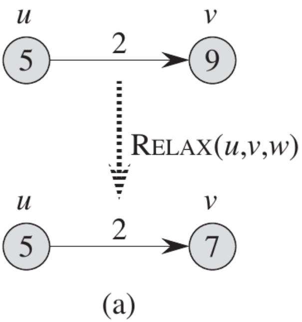
- The shortest path from v_0 to v_j contains shortest path from v_0 to v_i for any i.

Initializing the attributes

INITIALIZE-SINGLE-SOURCE(G, s)

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

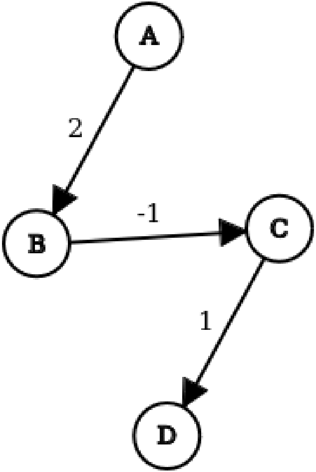
Relaxation



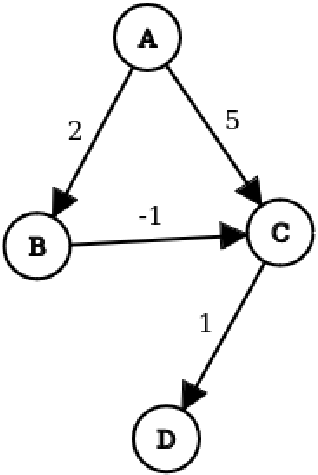
Relaxation

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

Repeated Edge Relaxation



Repeated Edge Relaxation



Relaxation

s	v_1	v_2	$\ldots$	v_{i-1}	v_i	v_{i+1}	$\ldots$	v_j
0	∞	∞	∞	∞	∞	∞	∞	∞

Relaxation

s	v_1	v_2	$\ldots$	v_{i-1}	v_i	v_{i+1}	$\ldots$	v_j
0	∞	∞	∞	∞	∞	∞	∞	∞

- After exactly $V - 1$ iterations, we would have found all the shortest paths containing at most $V - 1$ edges.

Relaxation

s	v_1	v_2	$\ldots$	v_{i-1}	v_i	v_{i+1}	$\ldots$	v_j
0	∞	∞	∞	∞	∞	∞	∞	∞

- After exactly $V - 1$ iterations, we would have found all the shortest paths containing at most $V - 1$ edges.
- What is the assumption here?

Relaxation

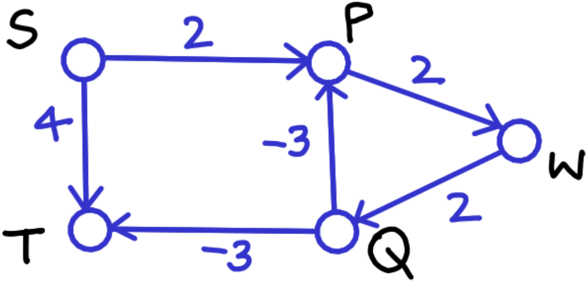
s	v_1	v_2	$\dots$	v_{i-1}	v_i	v_{i+1}	$\dots$	v_j
0	∞	∞	∞	∞	∞	∞	∞	∞

- After exactly $V - 1$ iterations, we would have found all the shortest paths containing at most $V - 1$ edges.
- What is the assumption here?
- If there is a negative weight cycle, we will be able to find shorter paths even after $V - 1$ iterations.

Bellman-Ford Algorithm

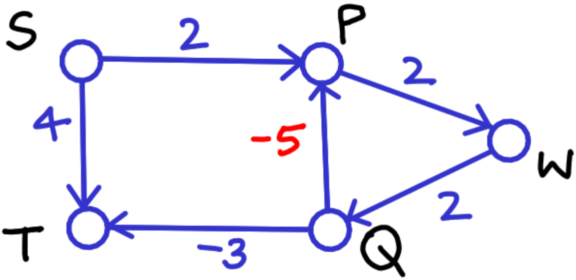
```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Bellman-Ford Algorithm



S	P	W	Q	T
0	∞	∞	∞	∞

Bellman-Ford Algorithm



S	P	W	Q	T
0	∞	∞	∞	∞

Improving Bellman-Ford Algorithm

Can we improve the average running time of Bellman-Ford algorithm?

Improving Bellman-Ford Algorithm

Can we improve the average running time of Bellman-Ford algorithm?

```
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
```

Dijkstra's algorithm

- Single source shortest path problem

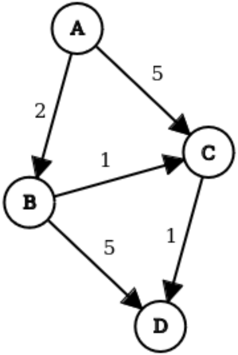
Dijkstra's algorithm

- Single source shortest path problem
- Assumes all the edges have non-negative weights.

Dijkstra's algorithm

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

Dijkstra's algorithm

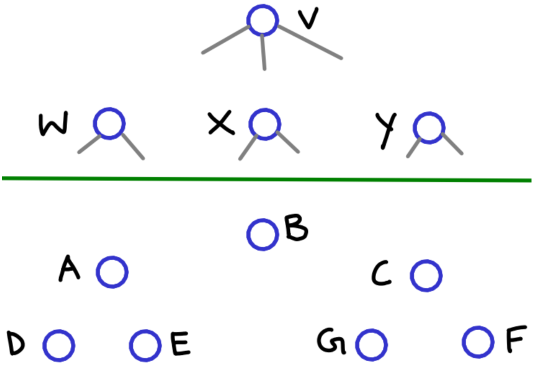


A	B	C	D
0	∞	∞	∞

Dijkstra's algorithm

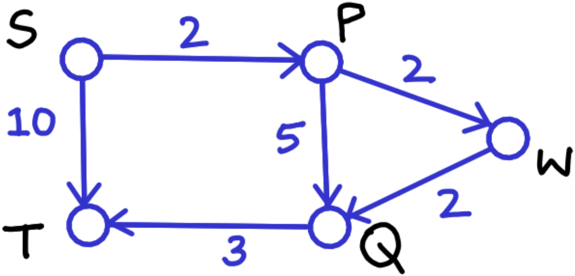
- Weight of a path will never decrease when we add an edge to the path.

Dijkstra's algorithm



- $S = \{V, W, X, Y\}$
- $(A, 5)$, $(B, 3)$, $(C, 7)$, $(G, 10)$, (D, ∞), (E, ∞), (F, ∞),

Dijkstra's algorithm



S	P	W	Q	T
0	∞	∞	∞	∞

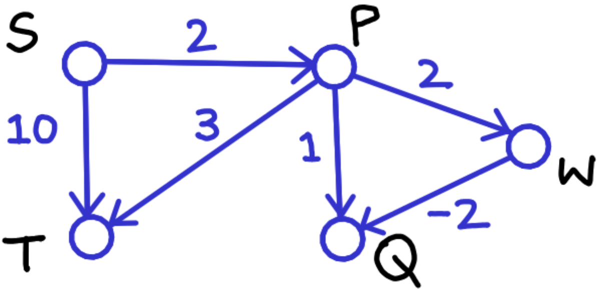
Running time of Dijkstra's algorithm

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = Ø
Q = Ø
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ Ø
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

Prim's algorithm

```
MST-PRIM(G, w, r)
for each vertex u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)            // add u to the tree
    for each vertex v in G.Adj[u] // update keys of u's non-tree neighbors
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
            DECREASE-KEY(Q, v, w(u, v))
```

Dijkstra's algorithm : The problem with negative weight edge



S	P	W	Q	T
0	∞	∞	∞	∞