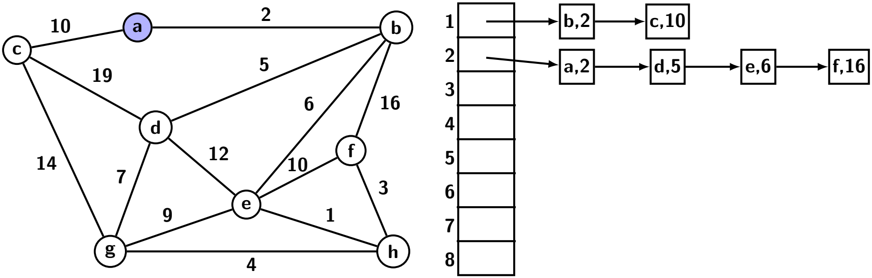
Example



a
b
c
d
e
f
g
h
2
10
19
5
6
16
14
7
12
10
3
9
1
4
1
2
3
4
5
6
7
8
b,2
c,10
a,2
d,5
e,6
f,16

Shortest Paths

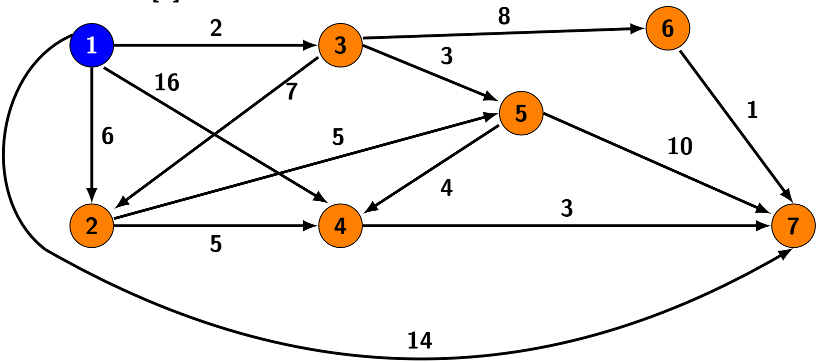
- Two possible variations in problems:
  1. Single source shortest paths
  2. All pairs of shortest paths
- Single source shortest path can be executed n times each with a different source vertex for all pairs shortest path.
- So, let us first examine how single source shortest path can be solved.

Djikstra's Algorithm

- Dijkstra's algorithm partitions the set of vertices logically into three partitions.
  - Set S: consists of vertices $u \in V$ such that dist[v] (from source s) already known.
  - Set I_1: consists of vertices $v \in V - S$ such that each $v \in I_1$ is directly connected to a vertex $x \in S$.
  - Set I_2: consists of vertices $w \in V - S - I_1$.
- Dijkstra's algorithm iteratively expands set S to include all vertices in V.

Example

Source vertex: d[1]=0

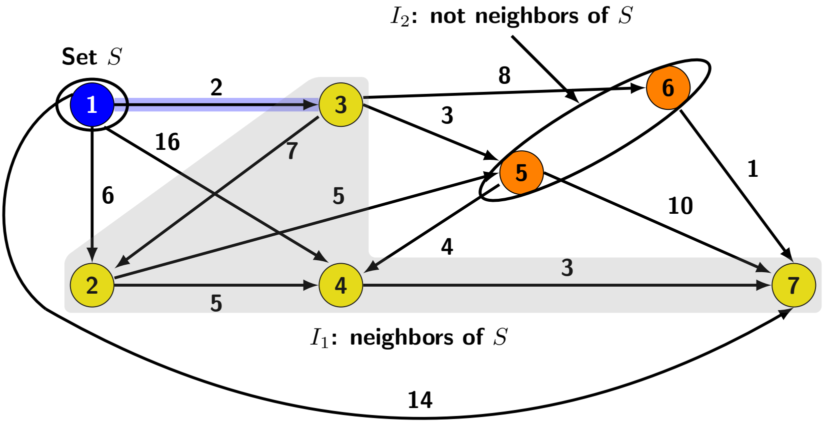


With source vertex 1, set $S = \{1\}$ and consider edges incident on vertices of S for relaxation.

Edge from S	old d[v]	new d[v]	old p[v]	new p[v]
2	∞	6	undef	1
3	∞	2	undef	1
4	∞	16	undef	1
7	∞	14	undef	1

Select vertex 3 (minimum d value) for inclusion into set S. So, $S = \{1, 3\}$ and $p[3] = 1$.

Blue colored vertices are in set S, yellow colored vertices are in set I_1, and organge colored are in set I_2.



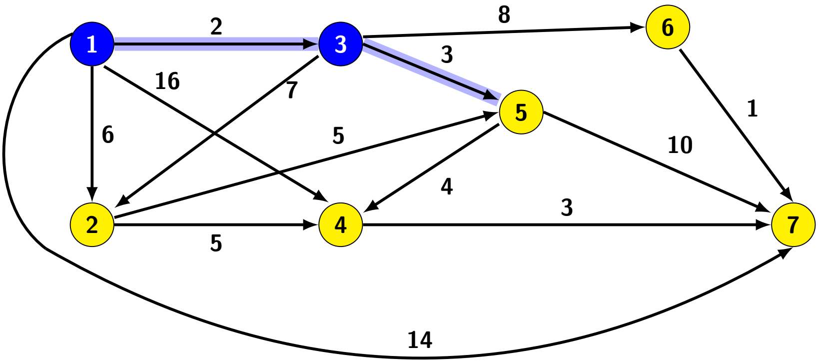
Shortest Paths

Now set $S = \{1, 3\}$ and only edges with end points 1 and 3 are considered for relaxation.

Edge from S	old d[v]	new d[v]	old p[v]	new p[v]
2	6	6	1	1
4	16	16	1	1
5	∞	5	undef	3
6	∞	10	undef	3
7	14	14	1	1

Select vertex 5 for inclusion into set S.
So $S = \{1, 3, 5\}$, and $p[3] = 1, p[5] = 3$

Blue colored vertices are in set S, yellow colored vertices are in set I_1.



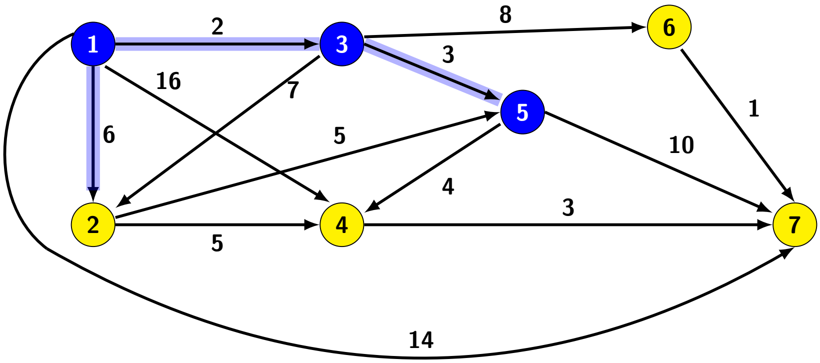
Shortest Paths

Now set $S = \{1, 3, 5\}$ and only edges with end points 1, 3 and 5 are considered for relaxation.

Edge from S	old d[v]	new d[v]	old p[v]	new p[v]
2	6	6	1	1
4	16	9	1	5
6	10	10	3	3
7	14	14	1	1

Select vertex 2 for inclusion into set S. So $S = \{1, 2, 3, 5\}$.
$p[3] = 1, p[5] = 3, p[2] = 1$

Blue colored vertices are in set S, yellow colored vertices are in set I_1.



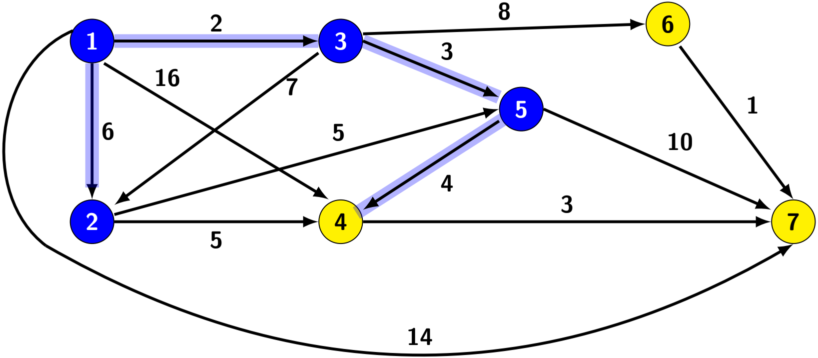
Shortest Paths

Now set $S = \{1, 2, 3, 5\}$ and only edges with end points 1, 2, 3 and 5 are considered for relaxation.

Edge from S	old d[v]	new d[v]	old p[v]	new p[v]
4	16	9	5	5
6	10	10	3	3
7	14	14	1	1

Select vertex 4 for inclusion into set S. So $S = \{1, 2, 3, 5\}$.
$p[3] = 1, p[5] = 3, p[2] = 1, p[4] = 5$

Blue colored vertices are in set 1, yellow colored vertices are in set 2.



Shortest Paths

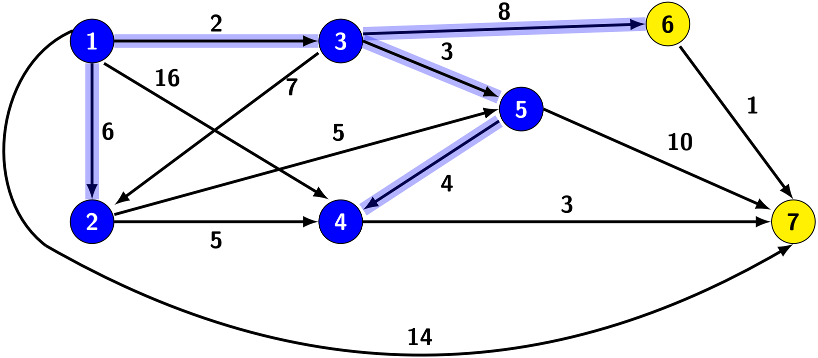
Now set $S = \{1, 2, 3, 4, 5\}$ and only edges with end points in S are considered for relaxation.

Edge from S	old d[v]	new d[v]	old p[v]	new p[v]
6	10	10	3	3
7	14	14	1	1

Select vertex 6 for inclusion into set S. So $S = \{1, 2, 3, 4, 5\}$.
$p[3] = 1, p[5] = 3, p[2] = 1, p[4] = 5, p[6] = 3$

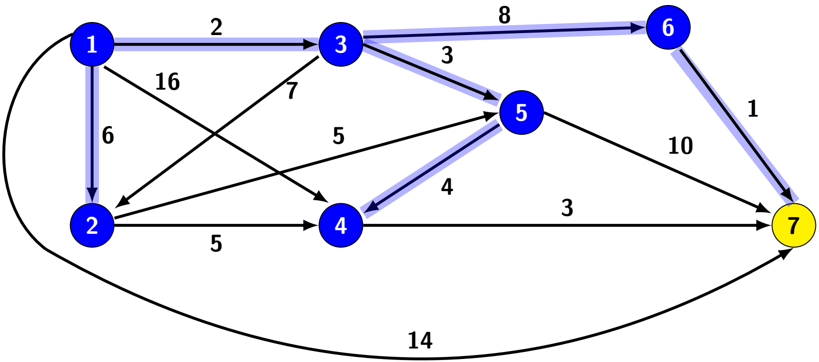
Shortest Paths

Blue colored vertices are in set 1, yellow colored vertices are in set 2.



Shortest Paths

The remaining vertex 7 is included in S the last iteration.
$S = \{1, 2, 3, 4, 5, 6, 7\}$ and
$p[3] = 1, p[5] = 3, p[2] = 1, p[4] = 5, p[6] = 3,\ p[7] = 6$.



Pseudo Code for Edge Relaxation

```
Relax(u, v) {
    new_d = min {d[v], d[u] + w(u,v)};
    if (new_d[v] < d[v]) {
        d[v] = new_d;
        p[v] = u;
    }
}
```

Pseudo Code for Dijkstra's Algorithm

```
for all (v ∈ V) {
    dist[v] = ∞; // Initialize distances
    prev[v] = undef;
}
choose(s); // Source
dist[s] = 0; // Initialize source distance
Q = V; // Initialize queue
while (!isEmpty(Q)) {
   u = min(d[u]); // Add new vertex
   Q = Q − {u}; // Update Q
   for each (v ∈ ADJ_G(u))
       Relax(u,v);
}
```