Single-Source All-Destinations Shortest Paths With General Weights

- Directed weighted graph.

- Edges may have negative cost.

- No cycle whose cost is < 0.

- Find a shortest path from a given source vertex s to each of the n vertices of the digraph.

Single-Source All-Destinations Shortest Paths With General Weights

- Dijkstra's $O(n^2)$ single-source greedy algorithm doesn't work when there are negative-cost edges.

Bellman-Ford Algorithm

- Single-source all-destinations shortest paths in digraphs with negative-cost edges.

- Uses dynamic programming.

- Runs in $O(n^3)$ time when adjacency matrices are used.

- Runs in $O(ne)$ time when adjacency lists are used.

Strategy

- To construct a shortest path from the source to vertex v, decide on the max number of edges on the path and on the vertex that comes just before v.

- Since the digraph has no cycle whose length is < 0, we may limit ourselves to the discovery of cycle-free (acyclic) shortest paths.

- A path that has no cycle has at most $n-1$ edges.

Cost Function d

- Let $d(v,k)$ ($dist^k[v]$) be the length of a shortest path from the source vertex to vertex v under the constraint that the path has at most k edges.

- $d(v,n-1)$ is the length of a shortest unconstrained path from the source vertex to vertex v.

- We want to determine $d(v,n-1)$ for every vertex v.

Value Of d(*,0)

- d(v,0) is the length of a shortest path from the source vertex to vertex v under the constraint that the path has at most 0 edges.

(s)

- d(s,0) = 0.
- d(v,0) = infinity for v != s.

Recurrence For d(*,k), k > 0

- $d(v,k)$ is the length of a shortest path from the source vertex to vertex v under the constraint that the path has at most k edges.

- If this constrained shortest path goes through no more than $k-1$ edges, then $d(v,k) = d(v,k-1)$.

Recurrence For d(*,k), k > 0

- If this constrained shortest path goes through k edges, then let w be the vertex just before v on this shortest path (note that w may be s).

- We see that the path from the source to w must be a shortest path from the source vertex to vertex w under the constraint that this path has at most $k-1$ edges.

- $d(v,k) = d(w,k-1) + \text{length of edge } (w,v)$.

Recurrence For d(*,k), k > 0

- d(v,k) = d(w,k-1) + length of edge (w,v).

S ⟶ W → V

- We do not know what w is.

- We can assert

  ▪ d(v,k) = min{d(w,k-1) + length of edge (w,v)}, where the min is taken over all w such that (w,v) is an edge of the digraph.

- Combining the two cases considered yields:

  ▪ d(v,k) = min{d(v,k-1),

     min{d(w,k-1) + length of edge (w,v)}}

Pseudocode To Compute d(*,*)

// initialize d(*,0)

d(s,0) = 0;

d(v,0) = infinity, v != s;

// compute d(*,k), 0 < k < n

for (int k = 1; k < n; k++)

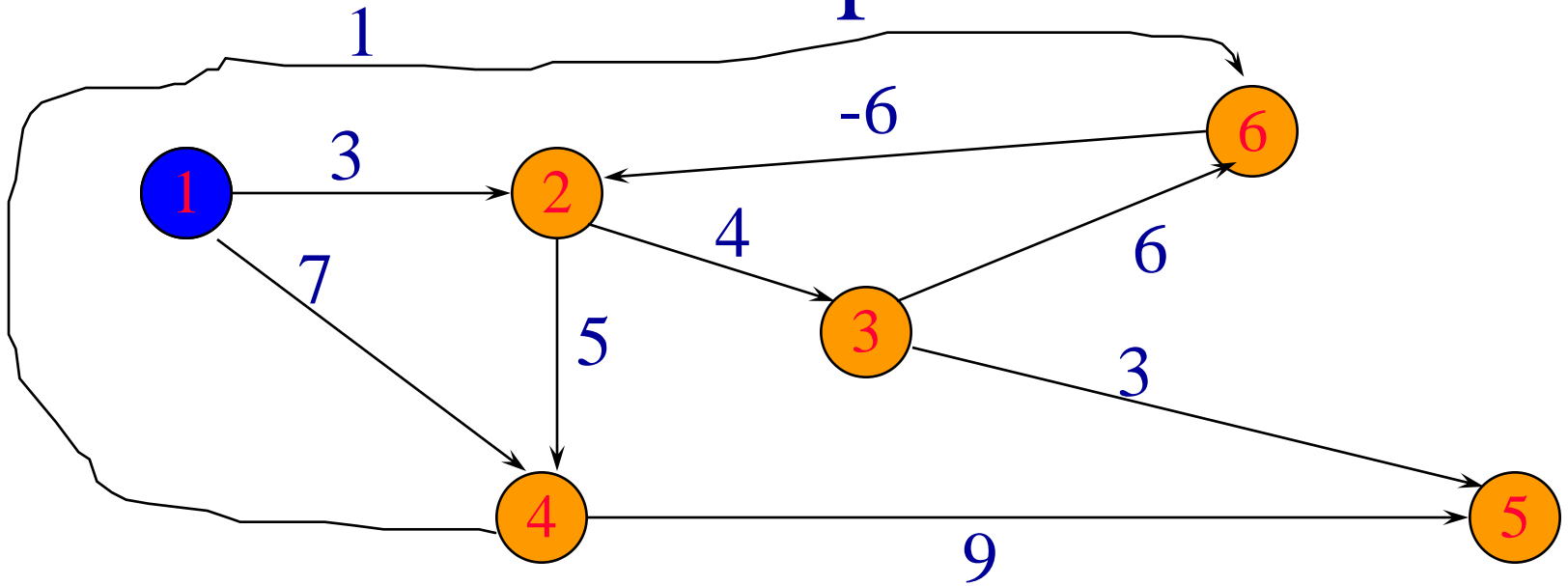{

    d(v,k) = d(v,k-1), 1 <= v <= n;

    for (each edge (u,v))

        d(v,k) = min{d(v,k), d(u,k-1) + cost(u,v)}

}

p(*,*)
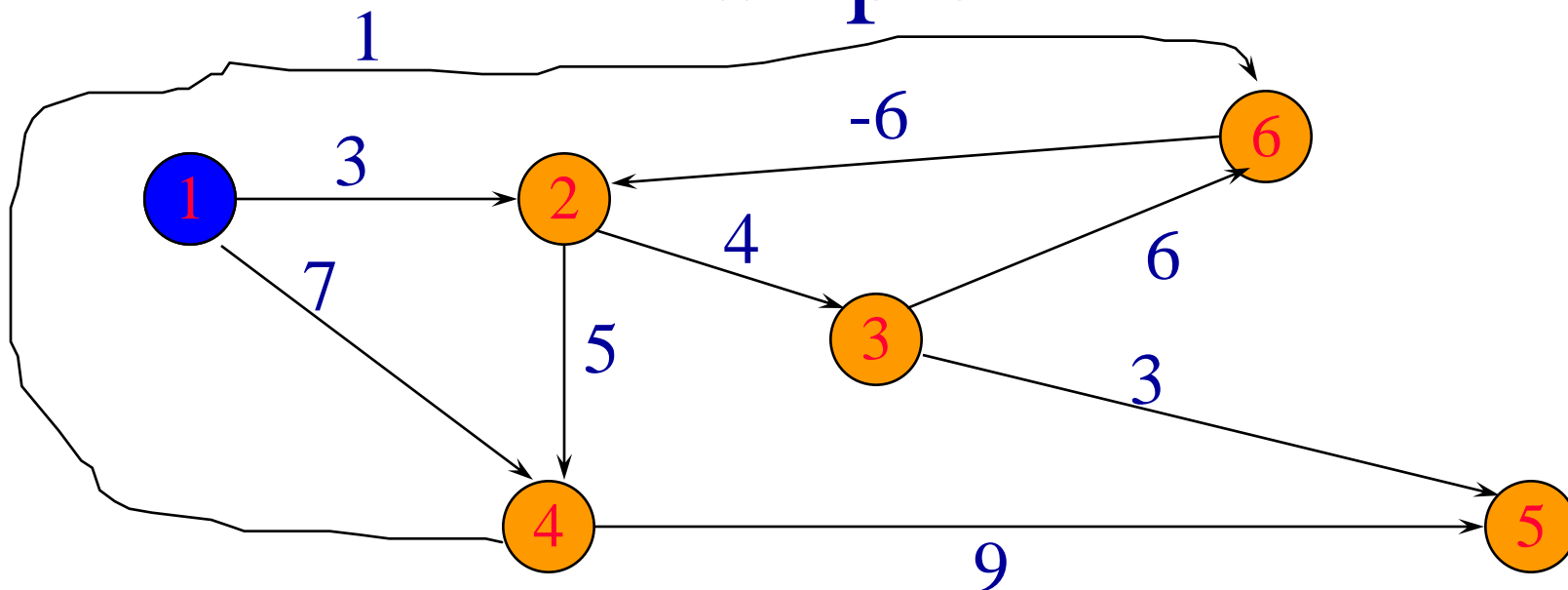
- Let p(v,k) be the vertex just before vertex v on the shortest path for d(v,k).

- p(v,0) is undefined.

- Used to construct shortest paths.
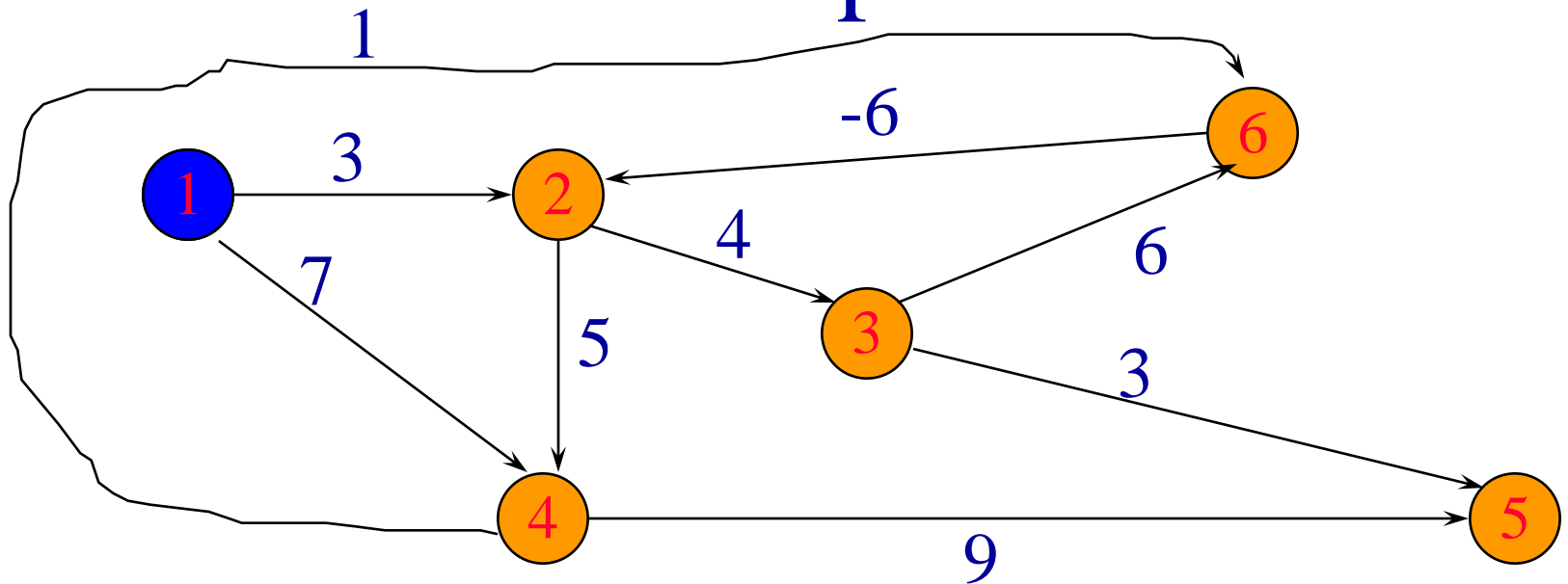
Example

Source vertex is 1.

Example

d(v,k)

	1	2	3	4	5	6
0	0	-	-	-	-	-
1	0	3	-	7	-	-
2	0	3	7	7	16	8
3	0	2	7	7	10	8
4	0	2	6	7	10	8

p(v.k)

v →					k ↓
-	-	-	-	-	-
-	1	-	1	-	-
-	1	2	1	4	4
-	6	2	1	3	4
-	6	2	1	3	4

Example

d(v,k)

	1	2	3	4	5	6
4	0	2	6	7	10	8
5	0	2	6	7	9	8

p(v.k)

	-	6	2	1	3	4
	-	6	2	1	3	4

Shortest Path From 1 To 5
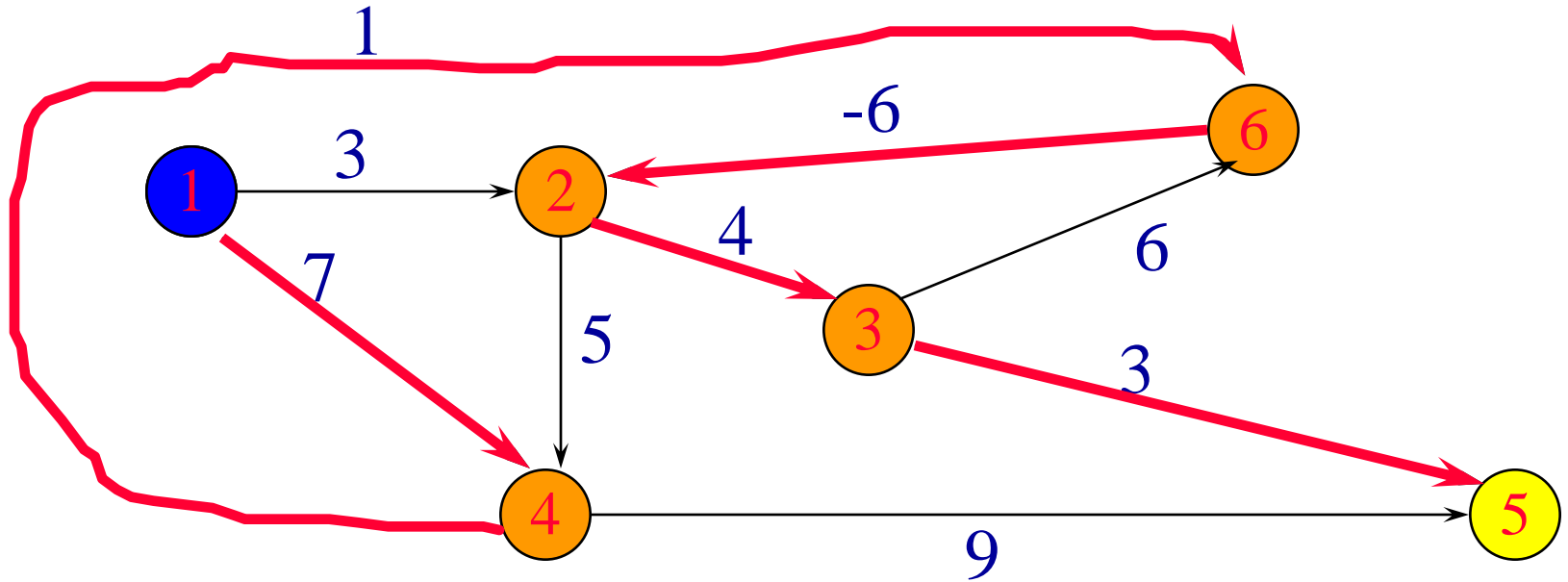
Observations

- d(v,k) = min{d(v,k-1),

             min{d(w,k-1) + length of edge (w,v)}}

- d(s,k) = 0 for all k.

- If d(v,k) = d(v,k-1) for all v, then d(v,j) = d(v,k-1), for all j >= k-1 and all v.

- If we stop computing as soon as we have a d(*,k) that is identical to d(*,k-1) the run time becomes
  - $O(n^3)$ when adjacency matrix is used.
  - O(ne) when adjacency lists are used.

Observations

- The computation may be done in-place.

  $d(v) = \min\{d(v), \min\{d(w) + \text{length of edge } (w,v)\}\}$

  instead of

  $d(v,k) = \min\{d(v,k-1),$

  $\min\{d(w,k-1) + \text{length of edge } (w,v)\}\}$

- Following iteration k, $d(v,k+1) <= d(v) <= d(v,k)$

- On termination $d(v) = d(v,n-1)$.

- Space requirement becomes $O(n)$ for $d(*)$ and $p(*)$.