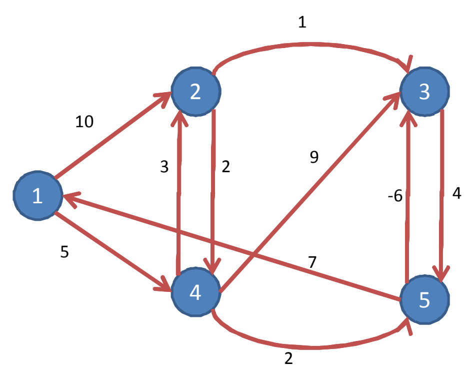
Single-source shortest paths

- A **path** of a weighted, directed graph is a sequence of vertices: $< v_1, v_2, \dots, v_k >$
- The **weight of a path** is the sum of weights of edges that make the path:

weight($< v_1, v_2, \dots, v_k >$)=$\sum_{i=1}^{k-1} w_{(v_i, v_{i+1})}$

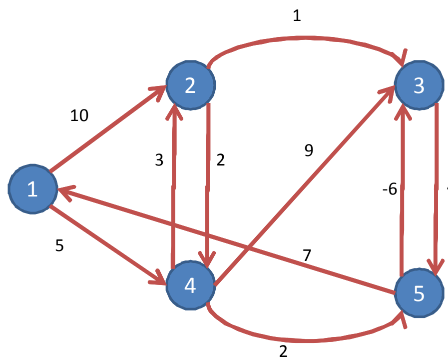
Single-source shortest paths



Given this weighted, directed graph, what is the weight of path: <1, 2, 4>?

10+2=12

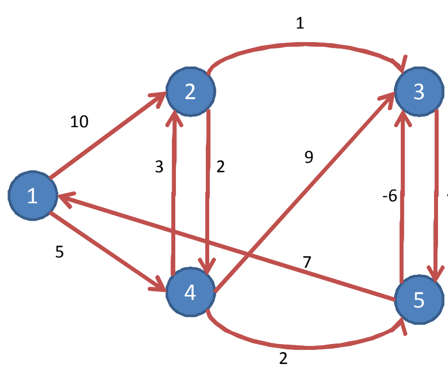
Single-source shortest paths



Given this weighted, directed graph, what is the weight of path: <1, 2, 4, 2, 4>?

10+2 + 3 + 2 = 17

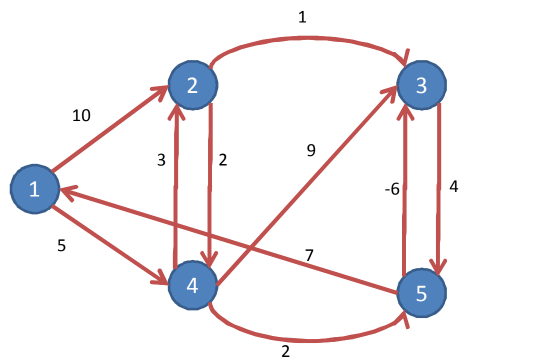
Single-source shortest paths



Given this weighted, directed graph, what is the weight of path: <1, 2, 4, 1>?

$10 + 2 + \infty = \infty$

Single-source shortest paths



Given this weighted, directed graph, what is the weight of path: <5, 3, 5> and <5, 3, 5, 3, 5>??

4-6 = -2 and -6+4-6+4 = -4

Negative cycle There is no shortest path from 1 to 5

Single-source shortest paths

- **Shortest path** of a pair of vertices <u, v>: a path from u to v, with minimum path weight
- Applications:
  - Your GPS navigator
  - If weights are time, it produces the fastest route
  - If weights are gas cost, it produces the lowest cost route
  - If weights are distance, it produces the shortest route

Single-source shortest paths

- **Single-source shortest path problem:** given a **weighted**, **directed** graph G=(V, E) with source vertex s, find all the shortest (least weight) paths from s to all vertices in V.

Single-source shortest paths

- Two classic algorithms to solve single-source shortest path problem
  - Dijkstra's algorithm
    - A greedy algorithm
    - Works when weights are all non-negative
  - Bellman-Ford algorithm
    - A dynamic programming algorithm
    - Works when some weights are negative
    - Dijkastra is faster than Bellman-Ford

Bellman-Ford algorithm

Observation:

- **If there is a negative cycle, there is no solution, but negative weight edges are allowed**
  - Add this cycle again can always produces a less weight path
- If there is no negative cycle, a shortest path has at most |V|-1 edges

Idea:

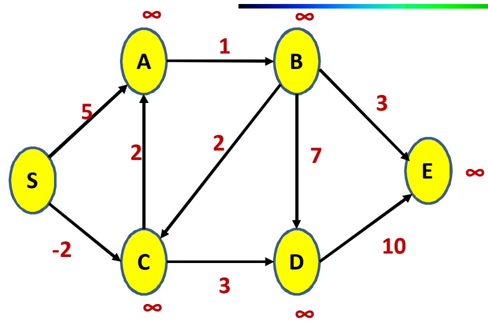
- Solve it using dynamic programming
- For all the paths have at most 0 edge, find all the shortest paths
- For all the paths have at most 1 edge, find all the shortest paths
- …
- For all the paths have at most |V|-1 edge, find all the shortest paths

Bellman-Ford algorithm

```
Bellman-Ford(G, s)
    for each v in G.V{   //Initialize 0-edge shortest paths
        if(v==s) d_{s,v}=0; else d_{s,v}= ∞; //set the 0-edge shortest distance
                                              from s to v
        π_{s,v} = NIL; //set the predecessor of v on the shortest path
    }
    Repeat |G.V|-1 times {   //bottom-up construct 0-to-(|V|-1)-edges shortest paths
        for each edge (u, v) in G.E{
            if(d_{s,v} > d_{s,u}+ w_{(u,v)}){
                d_{s,v} = d_{s,u}+ w_{(u,v)};
                π_{s,v} = u;
            }
    }
    for each edge (u, v) in G.E{  //test negative cycle
        If (d_{s,v} > d_{s,u}+ w_{(u,v)}) return false; // there is no solution
    }
    return true;
```

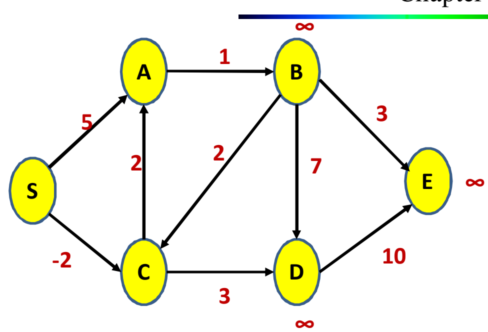
T(n)=O(VE)=O(V^3)

Chapter 5: Dynamic Programming

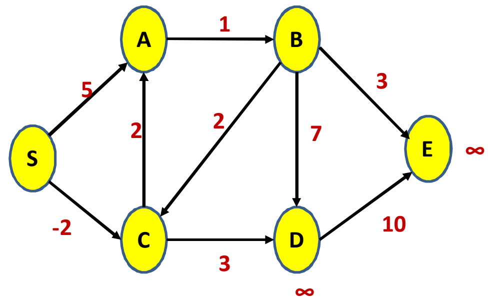


	S	A	B	C	D	E
d(v)	0	Inf	Inf	Inf	Inf	Inf
pi(v)						

Chapter 5: Dynamic Programming

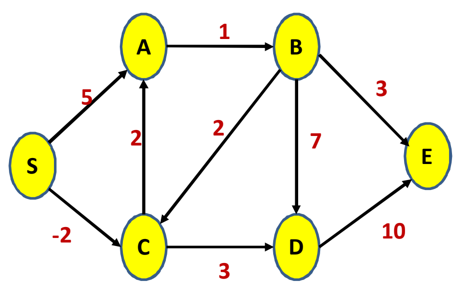


	S	A	B	C	D	E
d(v)	0	~~Inf~~ 5	Inf	~~Inf~~ -2	Inf	Inf
pi(v)		S		S		

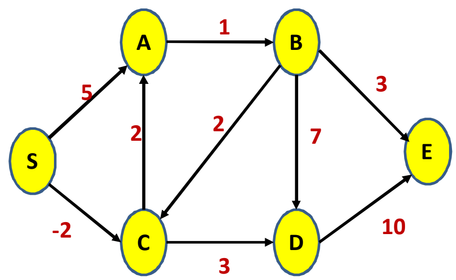


	S	A	B	C	D	E
d(v)	0	~~Inf~~ 5	~~Inf~~ 6	~~Inf~~ -2	Inf	Inf
pi(v)		S	A	S		

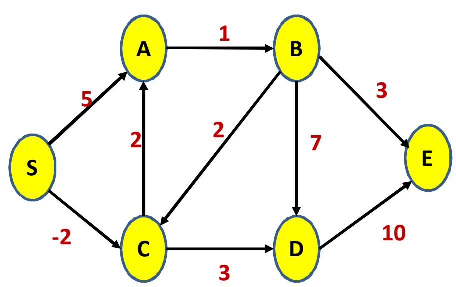
Chapter 5: Dynamic Programming



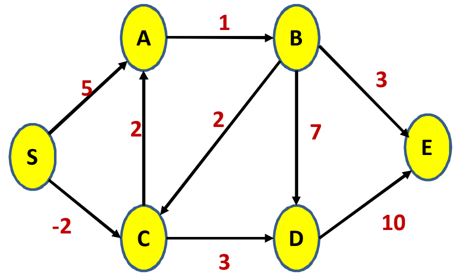
	S	A	B	C	D	E
d(v)	0	~~Inf~~ 5	~~Inf~~ 6	~~Inf~~ -2	~~Inf~~ 13	~~Inf~~ 9
pi(v)		S	A	S	B	B



	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ 6	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ 9
pi(v)		~~S~~ C	A	S	~~B~~ C	B



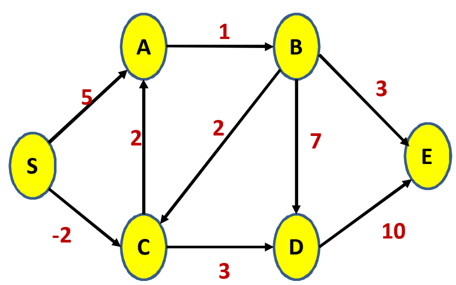
	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ 6	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ 9
pi(v)		~~S~~ C	A	S	~~B~~ C	B



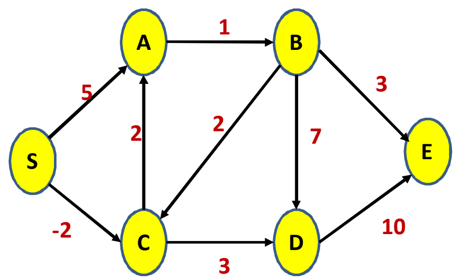
	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ 6	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ 9
pi(v)		~~S~~ C	A	S	~~B~~ C	B

First Iteration is over

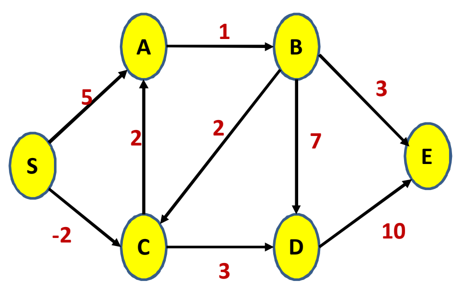
Chapter 5: Dynamic Programming



	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ 6	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ 9
pi(v)		~~S~~ C	A	S	~~B~~ C	B

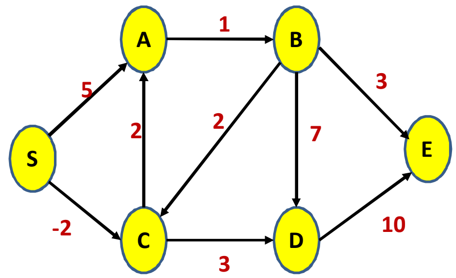


	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ 9
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	B



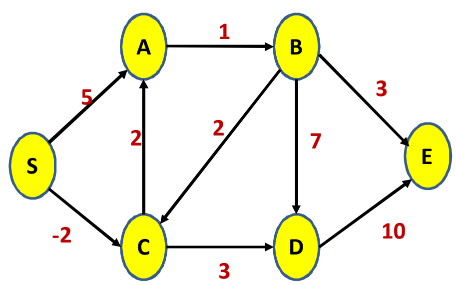
	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ ~~9~~ 4
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	~~B~~ B

Chapter 5: Dynamic Programming

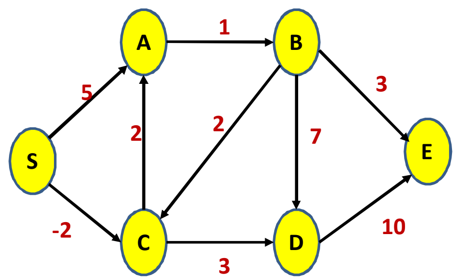


	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ ~~9~~ 4
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	~~B~~ B

Chapter 5: Dynamic Programming

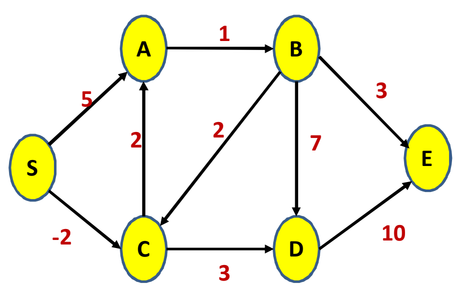


	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ ~~9~~ 4
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	~~B~~ B



	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ ~~9~~ 4
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	~~B~~ B

Chapter 5: Dynamic Programming



	S	A	B	C	D	E
d(v)	0	~~Inf~~ ~~5~~ 0	~~Inf~~ ~~6~~ 1	~~Inf~~ -2	~~Inf~~ ~~13~~ 1	~~Inf~~ ~~9~~ 4
pi(v)		~~S~~ C	~~A~~ A	S	~~B~~ C	~~B~~ B

End of Chapter 5