Competitive Programming Club

Meeting 2

Shortest Paths

Motivation

Sometimes we could convert a problem in to a graph, and a solution to the problem could be a path in the graph, and the optimal solution to the problem could be the shortest path in the graph.

Graph nodes could be thought of a state in the problem.

Dijkstra

Objective: finding the shortest path from one node to every other node in a weighted graph

- Vector data structure to store the graph
  - E[i] the nodes that could be reached from
- Algorithm template: https://cp-algorithms.com/graph/dijkstra.html
- Proof by induction

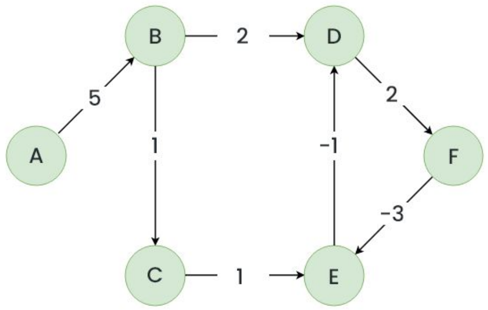
Bellman Ford

Objective: find the shortest path from one node to all other node, accounting for negative edges.(it can also be used to detect negative cycles)

- Algorithm Template: https://cp-algorithms.com/graph/bellman_ford.html
- The algorithm runs in $O(|V| \cdot |E|)$ times
- The Bellman-Ford algorithm finds the shortest paths from one starting point to all other points in a graph by updating the distances step-by-step, making sure each update is closer to the true shortest distance, and it can also detect if some paths have a cycle that makes the distance infinitely short.

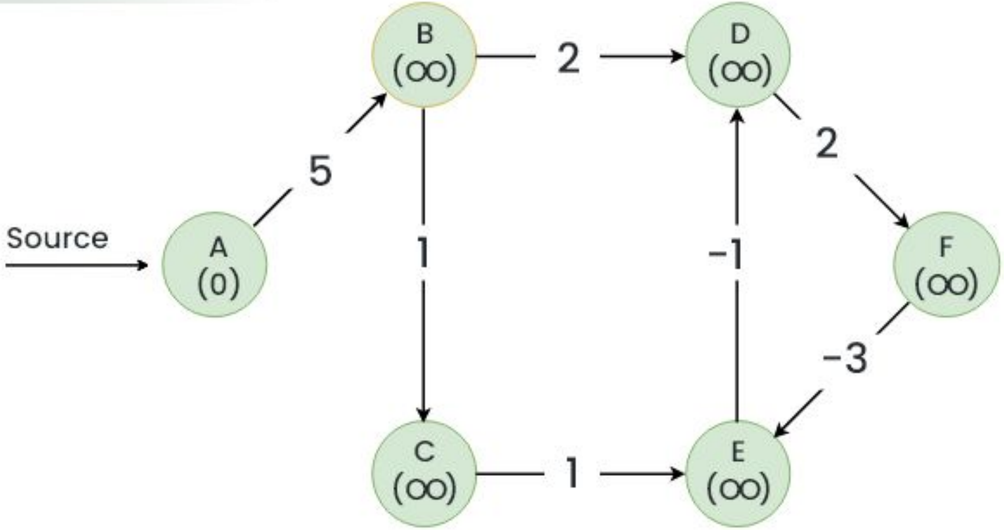
Toy problem

Suppose that we are given a weighted directed graph ***G*** with ***n*** vertices and ***m*** edges, and some specified vertex ***v***. You want to find the length of shortest paths from vertex ***v*** to every other vertex.



Bellman-Ford To Detect A Negative Cycle In A Graph

Initialize The Distance Array

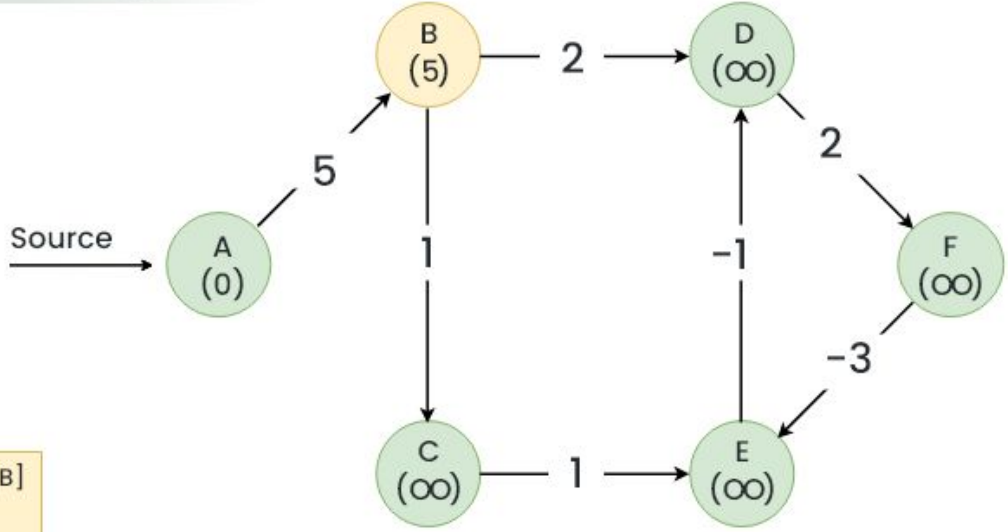


Distance Array
Dist[]

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

Bellman-Ford To Detect A Negative Cycle In A Graph

1st Relaxation Of Edges



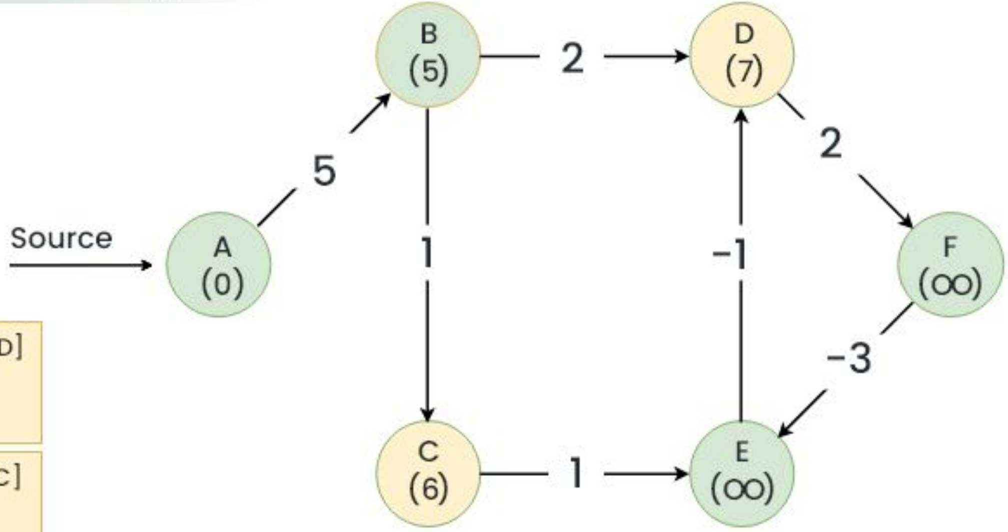
Dist [A] + 5 <Dist[B]
0+5<(∞)
Dist[B] = 5

Distance Array

A	B	C	D	E	F
0	∞	∞	∞	∞	∞

A	B	C	D	E	F
0	5	∞	∞	∞	∞

2nd Relaxation Of Edges



Dist [B] + 2 <Dist[D]
5+2<(∞)
Dist[D] = 7

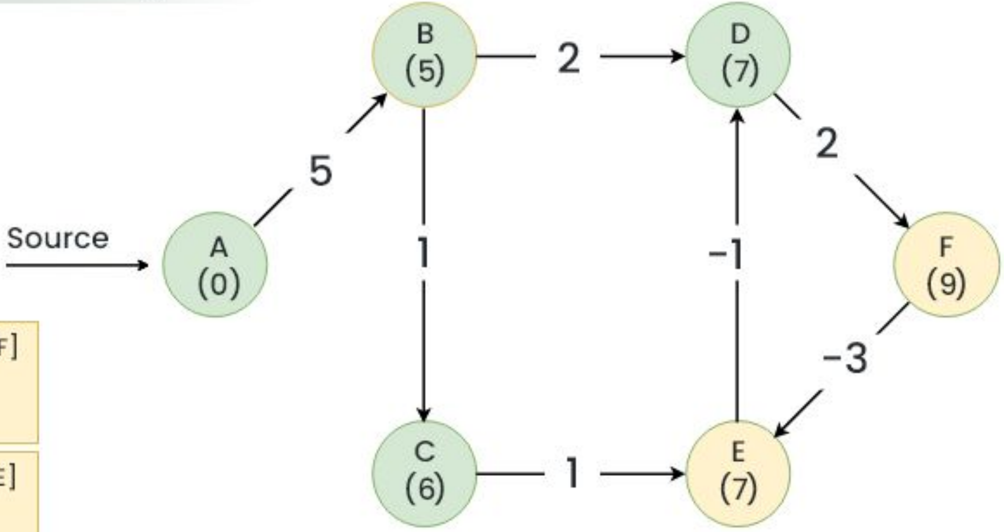
Dist [B] + 1 <Dist[C]
5+1 <(∞)
Dist[C] = 6

Distance Array

A	B	C	D	E	F
0	5	∞	∞	∞	∞

A	B	C	D	E	F
0	5	6	7	∞	∞

3rd Relaxation Of Edges



Dist [D] + 2 <Dist[F]
7+2<(OO)
Dist[F] = 9

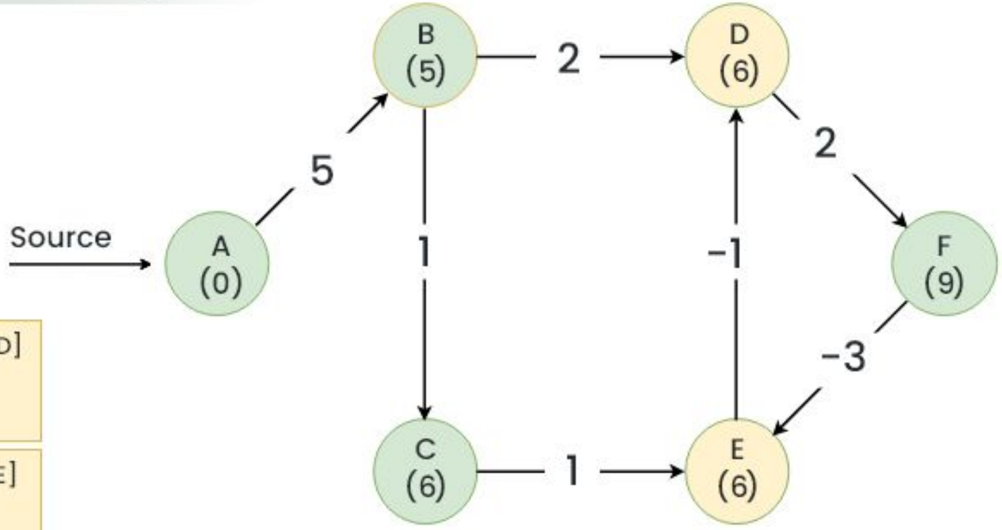
Dist [C] + 1 <Dist[E]
6+1 <(OO)
Dist[E] = 7

Distance Array

A	B	C	D	E	F
0	5	6	7	∞	∞

A	B	C	D	E	F
0	5	6	7	7	9

4th Relaxation Of Edges



Dist [E] + 2 <Dist[D]
7+(-1)< 7
Dist[D] = 6

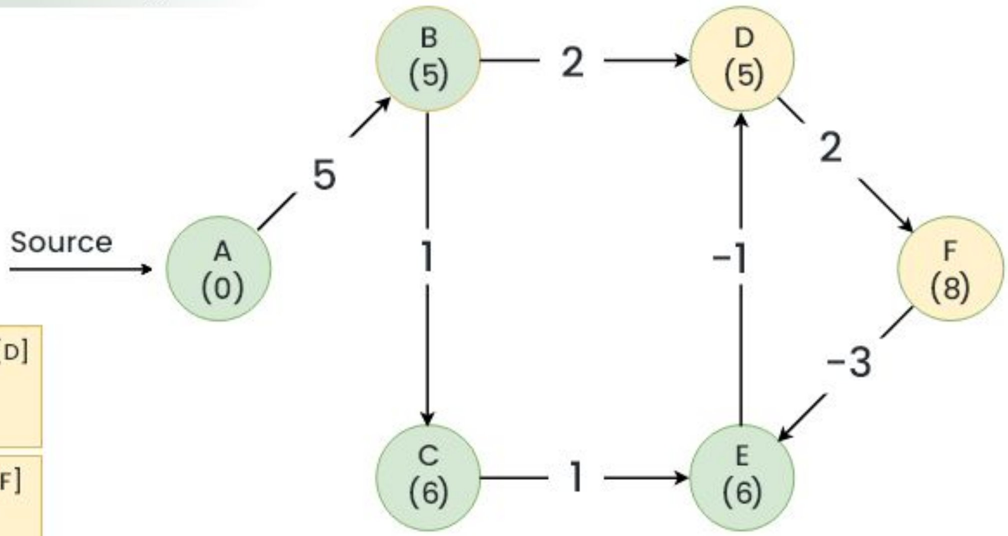
Dist [F] + 1 <Dist[E]
9+(-3) < 6
Dist[E] = 6

Distance Array

A	B	C	D	E	F
0	5	6	7	7	9

A	B	C	D	E	F
0	5	6	6	6	9

5th Relaxation Of Edges



Dist [E] + (-1) < Dist[D]
6+(-1)< 6
Dist[D] = 5

Dist [D] + 2 < Dist[F]
6+2< 9
Dist[F] = 8

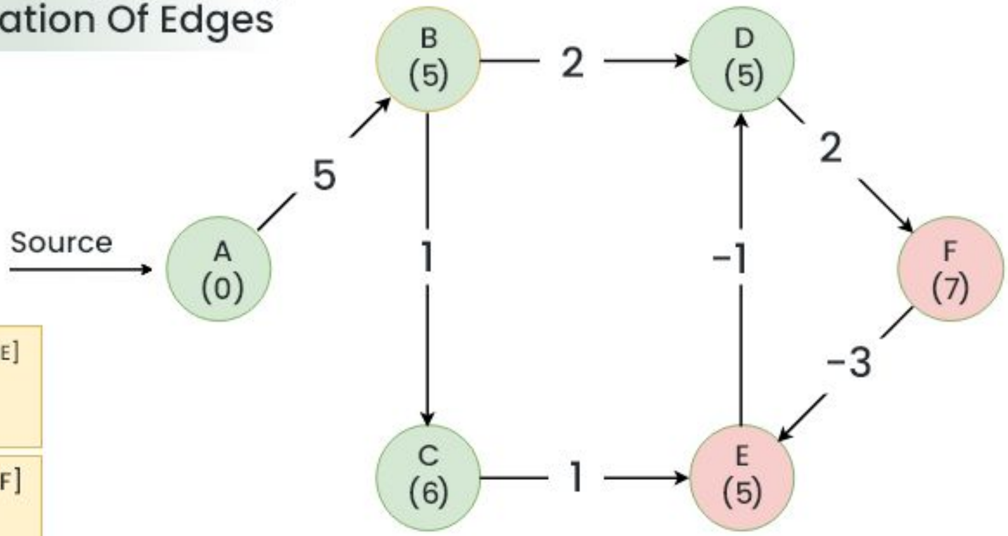
Distance Array

A	B	C	D	E	F
0	5	6	6	6	9

↓

A	B	C	D	E	F
0	5	6	5	6	8

Detecting The Negative Edge By 6Th Relaxation Of Edges



Dist [F] + (-3) < Dist[E]
8+(-3)<6
Dist[E] = 5

Dist [D] + 2 < Dist[F]
6+2<8
Dist[F] = 7

Distance Array

A	B	C	D	E	F
0	5	6	5	6	8

↓

A	B	C	D	E	F
0	5	6	4	4	6

Floyd Warshall

Objective: finding the shortest path between every two nodes

- The use adjacency matrix e[i][j] is the current shortest path for nodes i and j
- Algorithm template: https://cp-algorithms.com/graph/all-pair-shortest-path-floyd-warshall.html
- It could be thought of as a DP thus proved by induction
  - E[i][j] is actually e[k][i][j], the shortest path reached between i and j using a intermediary node no larger than k
  - Thus e[i][j] = min{e[i][j], e[i][k] + e[k][j]} is equivalent of e[k][i][j] = min{e[k-1][i][j], e[k-1][i][k] + e[k-1][k][j]}
  - Keep this property in mind for one of the practice problems

Example Problem

You are given a number, each time you could apply the following operation to the number:

1. X += add[j1], requiring brain_energy[1][j1]
2. X /= div[j2], requiring brain_energy[2][j2]
3. X *= mul[j3], requiring brain_energy[3][j3]
4. X %= mod[j4], requiring brain_energy[4][j4]

Now you have Q queries, each query you want to reach another number qi, what is the answer to each queries? The minimum brain energy you need to reach qi

Q <= 10^6, x <= 10^6

Practice Problems

https://vjudge.net/contest/622365

Password: ucsd_icpc