Fall, 2023

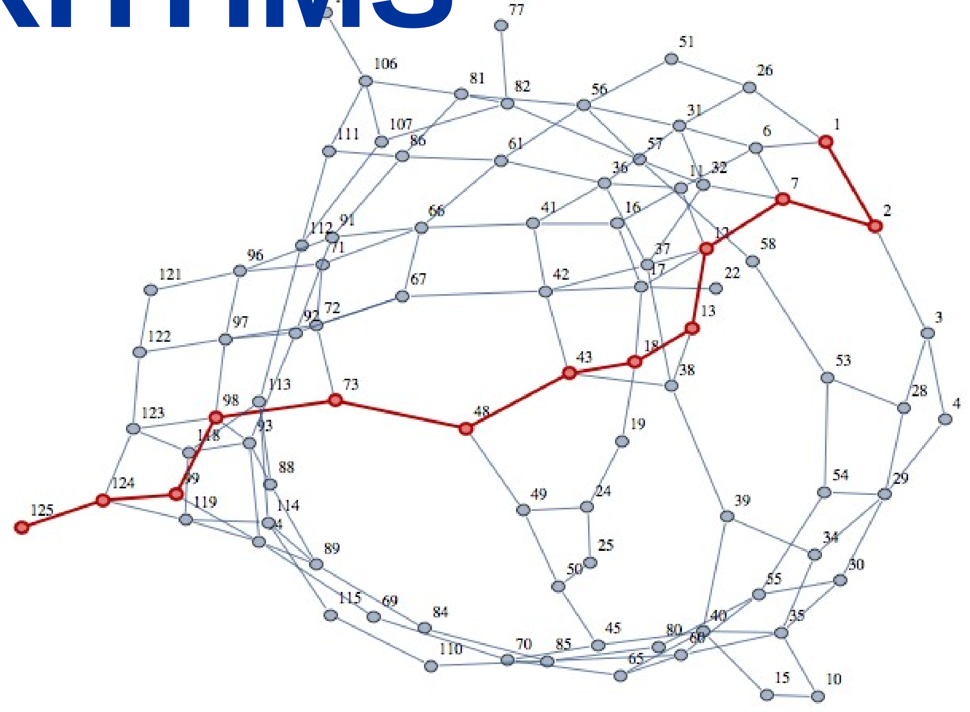
Discrete Mathematics

Lecture 10: Graph – Part 2

Ta Quang Hung

Contents

SHORTEST PATH ALGORITHMS



Introduction

- Perhaps the most intuitive graph-processing problem is one that you encounter regularly, when using a map application or a navigation system to get directions from one place to another. A graph model is immediate: vertices correspond to intersections and edges correspond to roads, with weights on the edges that model the cost, perhaps distance or travel time. The possibility of one-way roads means that we will need to consider edge-weighted *digraphs*. In this model, the problem is easy to formulate:

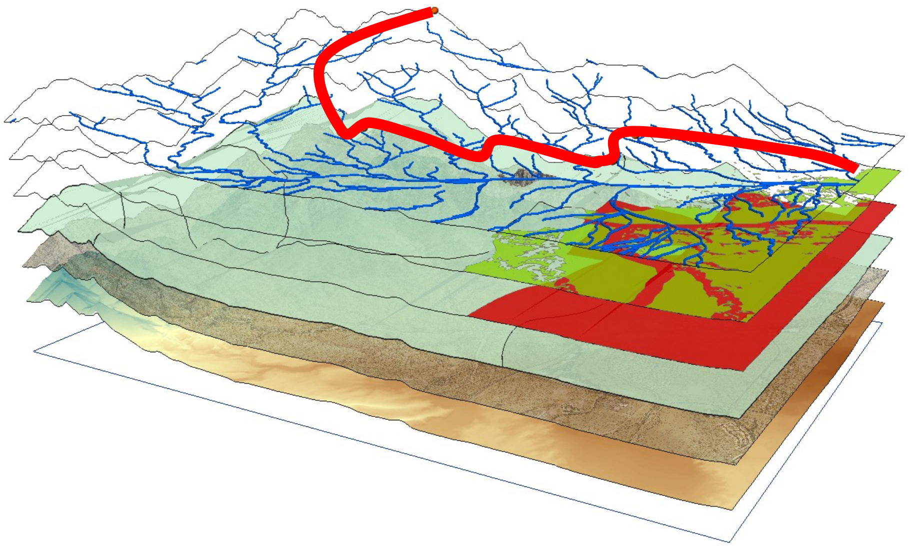
Find the lowest-cost way to get from one vertex to another

Example: Navigation System

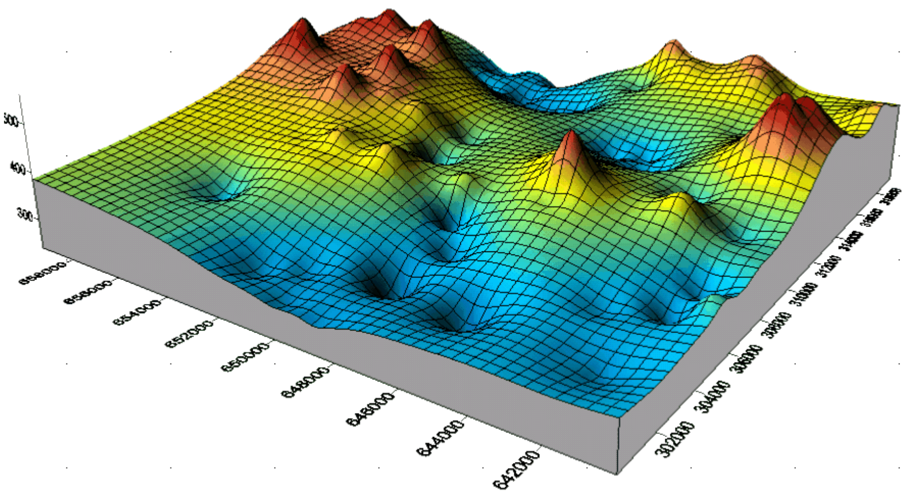
Vertices correspond to intersections and edges correspond to roads.

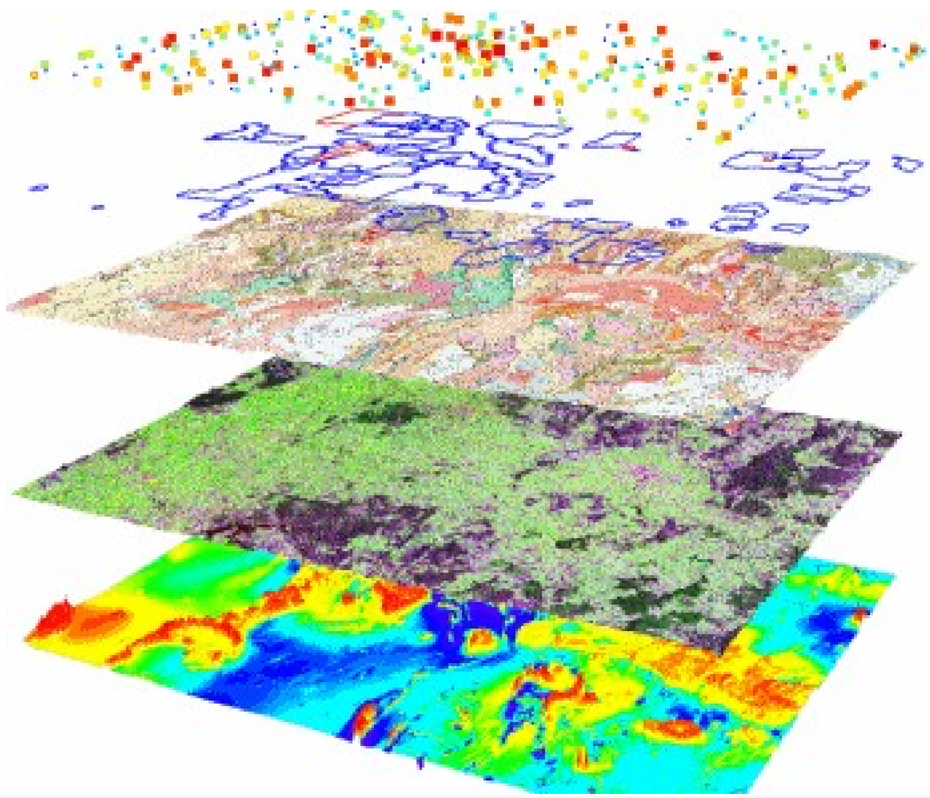
Weights on the edges that model the cost, perhaps distance or travel time.

Example: Geographic Information System (GIS)

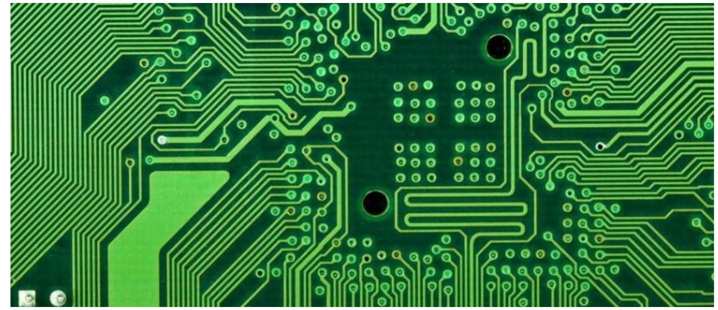


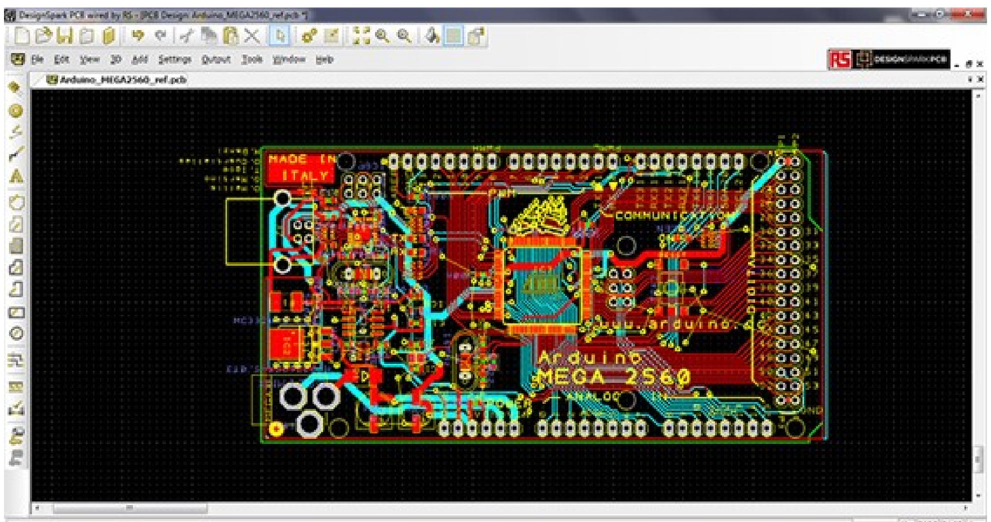
Multiple layers are combined to build GIS

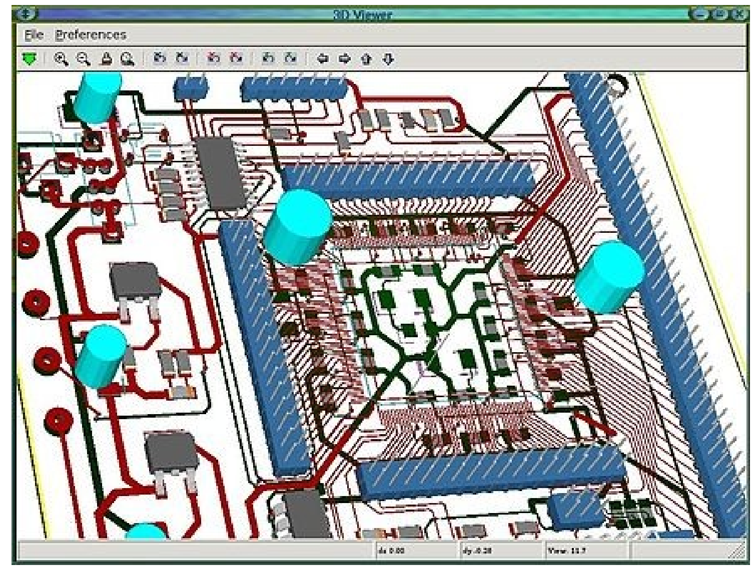




Example: Printed Circuit Board (PCB)

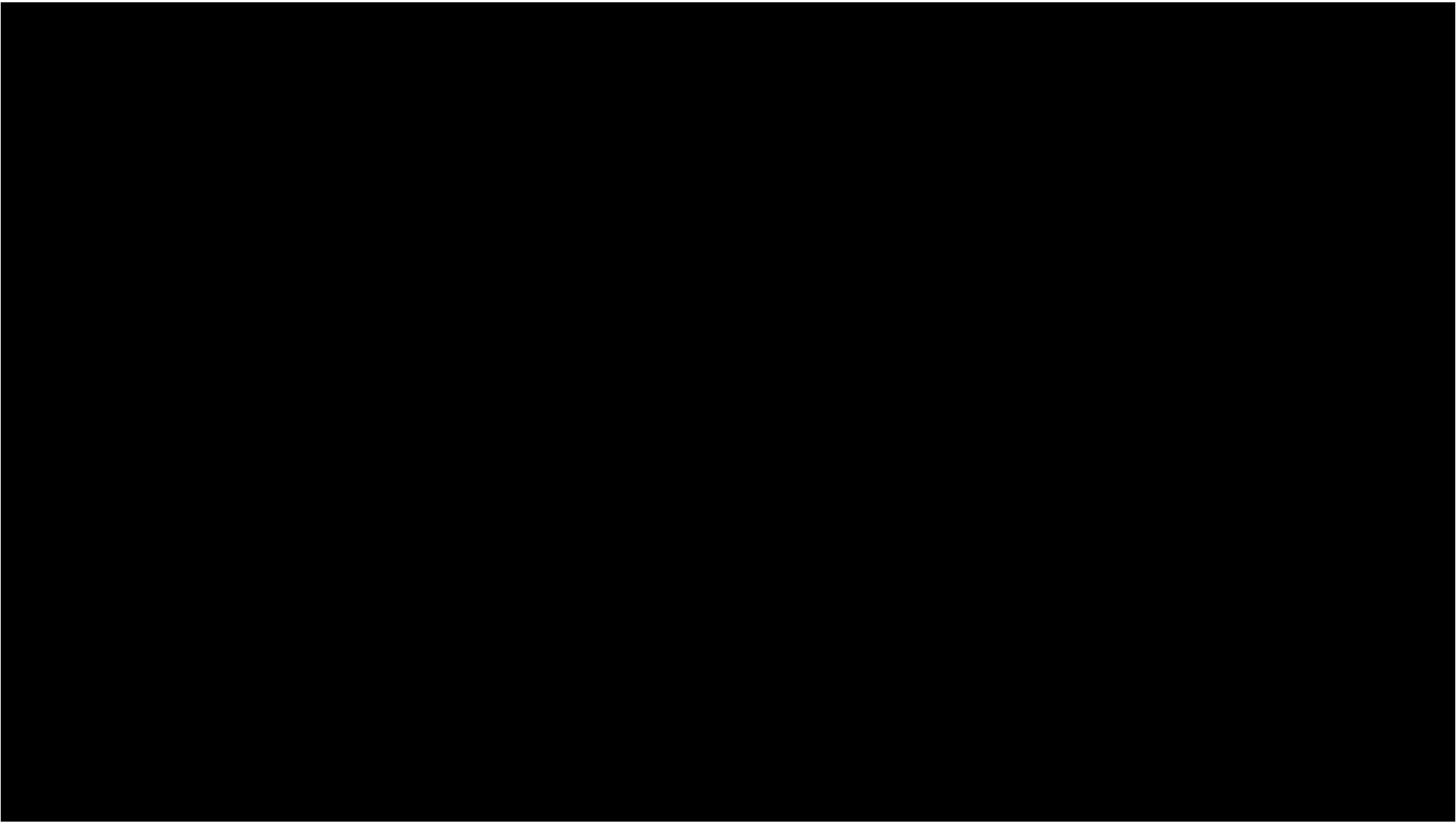




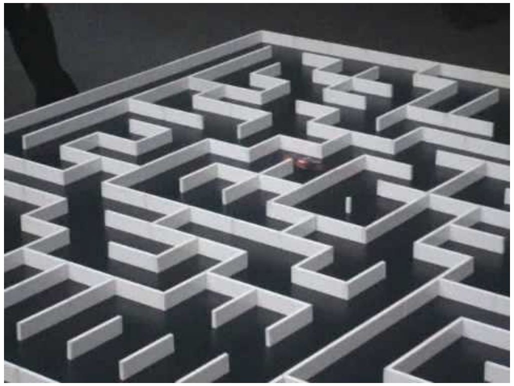


Autoroute Mode: Find the shortest path for connections to minimize resistances.

Example: PCB layout for DDR3 RAM interface



Example: Game

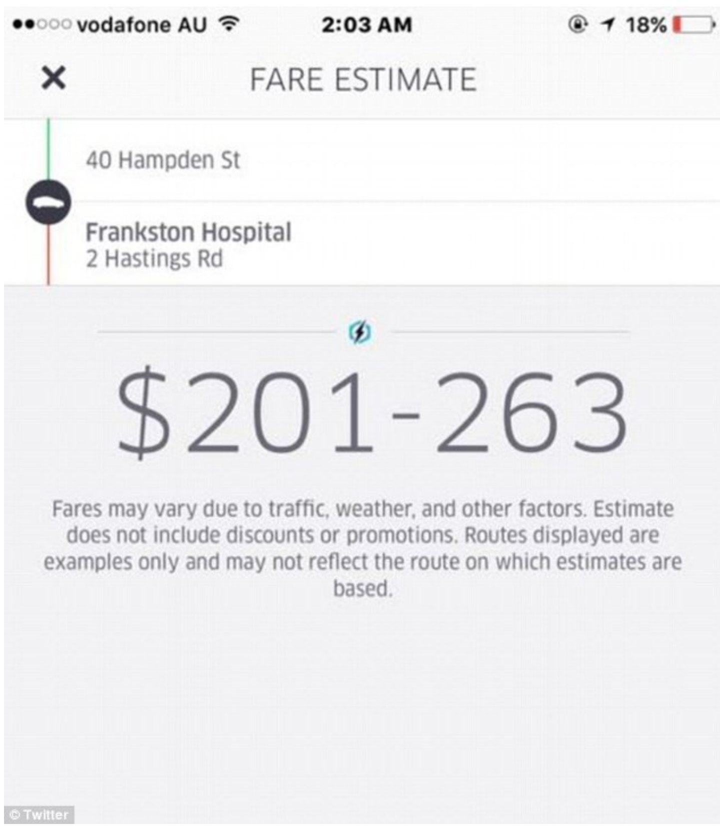


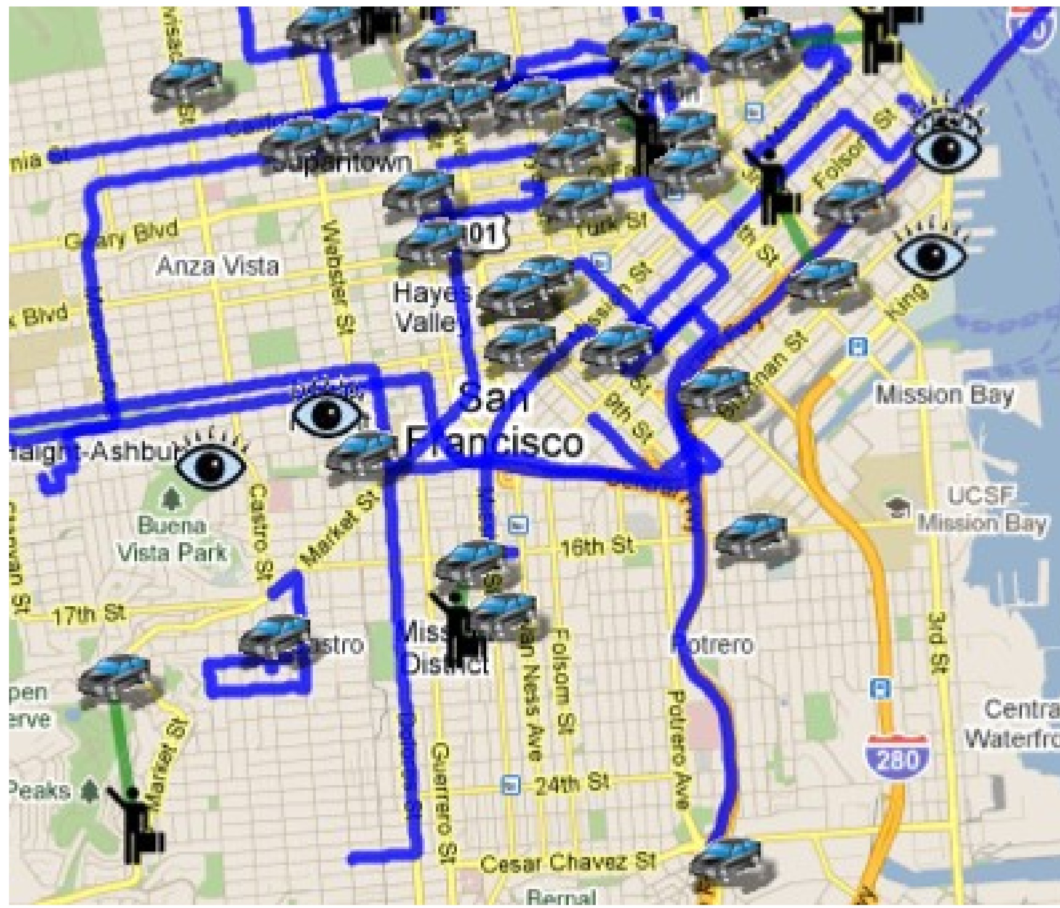
Game map: Look like a maze.

Base under attack: Troops must find the shortest path back home.

Example: Uber Taxi

Fare Estimate: calculate the minimum fee for the trip.





Properties of Shortest Path

- ***Paths are directed***. A shortest path must respect the direction of its edges.
- ***The weights are not necessarily distances.*** *W*e use examples where vertices are points in the plane and weights are Euclidean distances, such as the digraph on the facing page. But the weights might represent time or cost or an entirely different variable and do not need to be proportional to a distance at all. We are emphasizing this point by using mixed-metaphor terminology where we refer to a *shortest* path of minimal *weight* or *cost*.
- ***Not all vertices need be reachable***. If t is not reachable from s, there is no path at all, and therefore there is no shortest path from s to t. For simplicity, our small running example is strongly connected (every vertex is reachable from every other vertex).

Properties of Shortest Path (con't)

- ***Negative weights introduce complications***. For the moment, we assume that edge weights are positive (or zero).
- ***Shortest paths are normally simple***. Algorithms ignore zero-weight edges that form cycles, so that the shortest paths they find have no cycles.
- ***Shortest paths are not necessarily unique***. There may be multiple paths of the lowest weight from one vertex to another; we are content to find any one of them.
- ***Parallel edges and self-loops may be present***. Only the lowest-weight among a set of parallel edges will play a role, and no shortest path contains a self-loop (except possibly one of zero weight, which we ignore). In the text, we implicitly assume that parallel edges are not present for convenience in using the notation v->w to refer unambiguously to the edge from v to w.

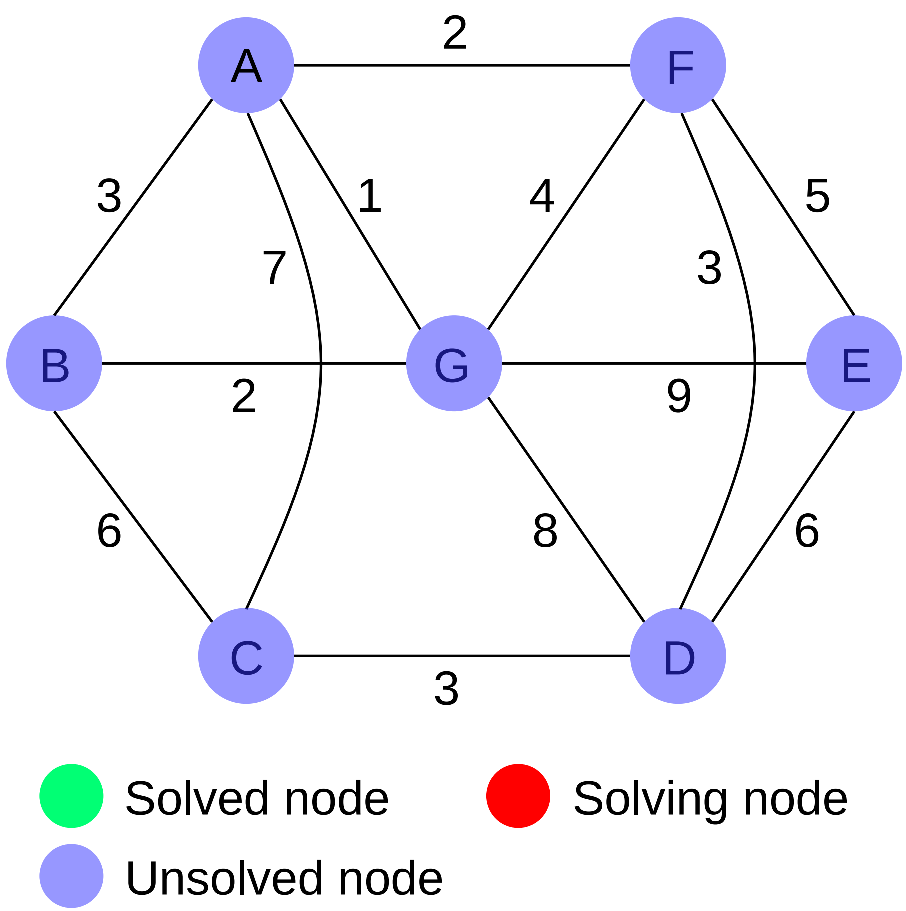
DIJKSTRAS ALGORITHM

Dijkstras Algorithm

- This algorithm can't work with negative edge weights.
- Dijkstras algorithm runs in $O(|n|+|m|\log|m|)$ where n vertices and m edges.

Dijkstras Algorithm: Example

Step 0



Find path from A to all other vertices?

Denote:

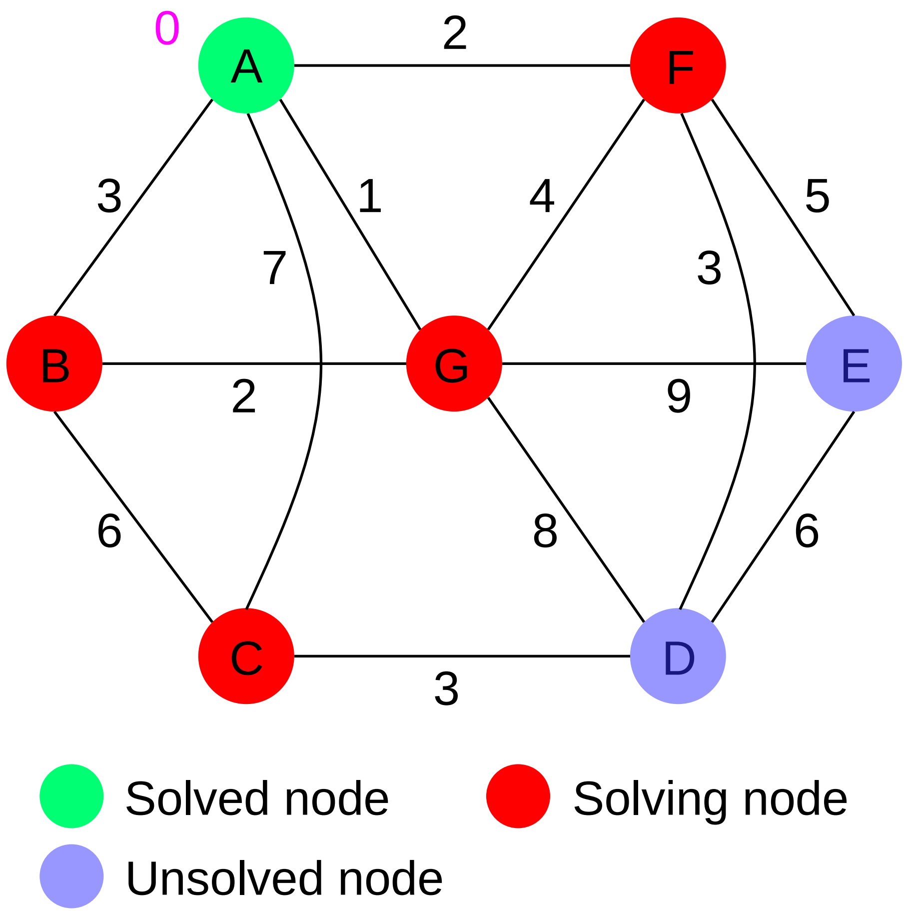
- w(A,B) = weighting coefficient in the path from A to B. So, from the graph, we have w(A,B) = 3.
- L(Y) = Label of Y. Minimum cost of Y or the length of the shortest path from orignating vertice to Y. L(Y) is a sum of all weighting coefficient and the cost of all vertices on the shortest path.
- S: A set of all vertices that the shortest path go through.

Set L(A) = 0

S = {A}

Dijkstras Algorithm: Example

Step 1



Check all paths from A to adjiacent vertices: F, G, B, C

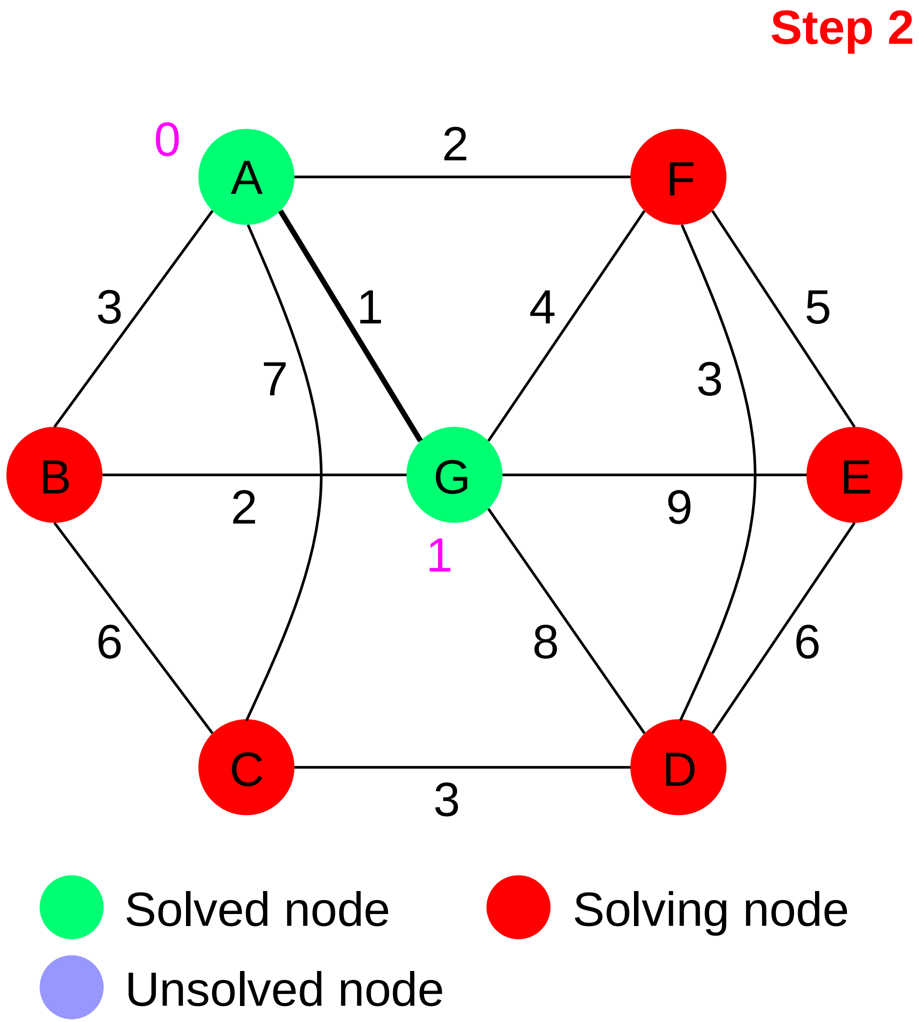
$\boxed{L(A) + w(A,G) = 1}$
L(A) + w(A,B) = 3
L(A) + w(A,F) = 2
L(A) + w(A,C) = 7

Set L(G) = 1

Next vertice is G, add G to set S

S = {A,G}

Dijkstras Algorithm: Example



Check all paths from A, G: F, B, E, D, C.

L(A) + w(A,B) = 3

$\boxed{\text{L(A) + w(A,F) = 2}}$

L(A) + w(A,C) = 7

L(G) + w(G,F) = 5

L(G) + w(G,E) = 10

L(G) + w(G,D) = 9

L(G) + w(G,B) = 3

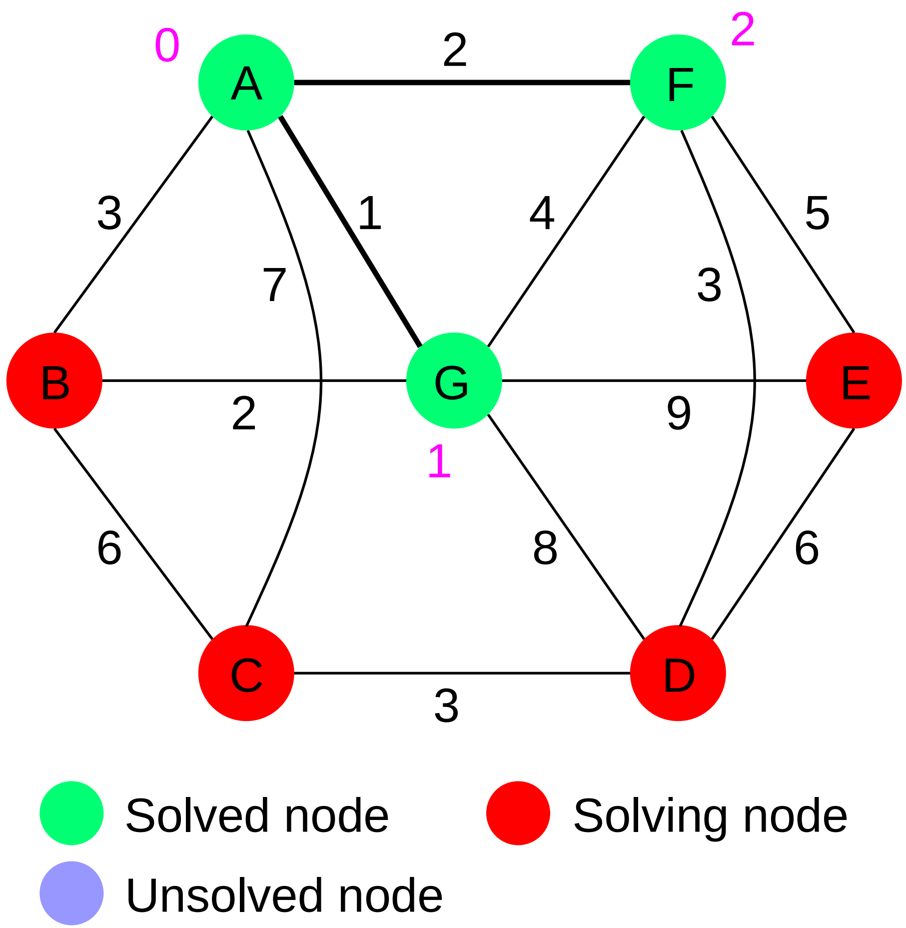
Next vertice is F, set L(F)=2, add F to set S:

S = {A,G,F}

Dijkstras Algorithm: Example

Step 3

Check all paths from A, G, F to adjiacent vertices: B, E, C, D.



$\boxed{L(A) + w(A,B) = 3}$

$L(A) + w(A,C) = 7$

$L(G) + w(G,E) = 10$

$L(G) + w(G,D) = 9$

$L(G) + w(G,B) = 3$

$L(F) + w(F,E) = 7$

$L(F) + w(F,D) = 5$

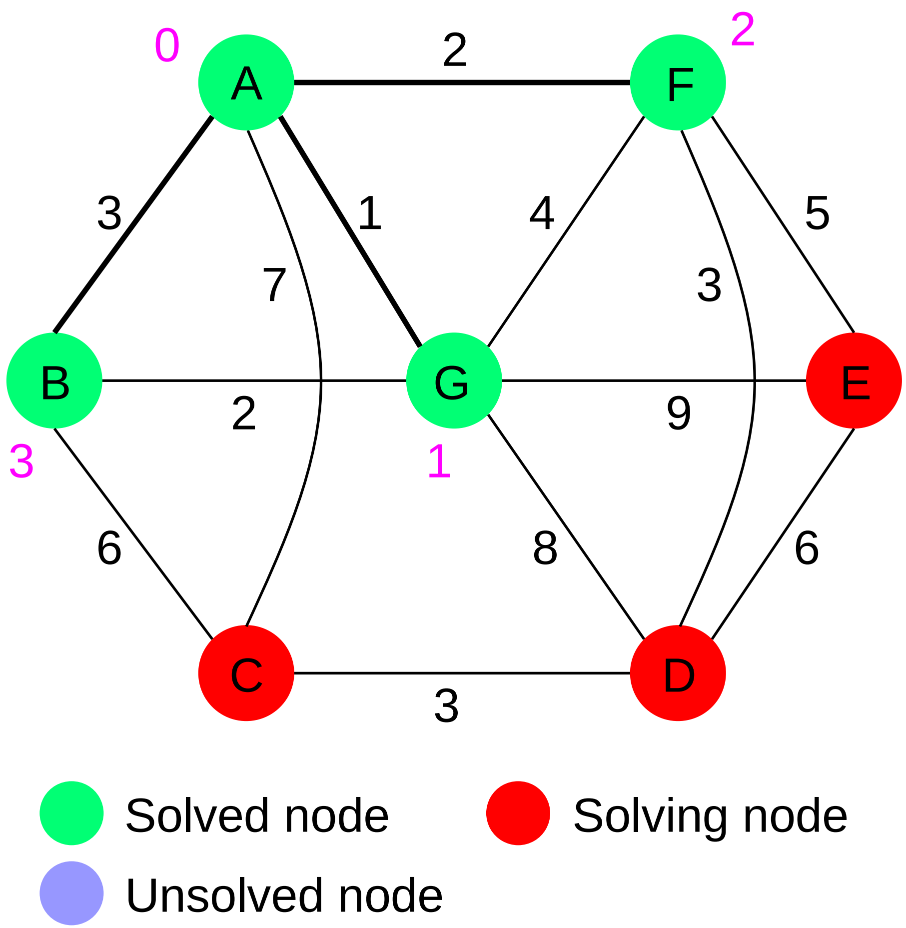
Next vertice is B, set L(B) = 3, add B to set S:

S = {A,G,F,B}

Dijkstras Algorithm: Example

Step 4

Check all paths from A, G, F, B to adjiacent vertices: E, C, D.



L(A) + w(A,C) = 7

L(G) + w(G,E) = 10

L(G) + w(G,D) = 9

L(F) + w(F,E) = 7

$\boxed{\textbf{L(F) + w(F,D) = 5}}$

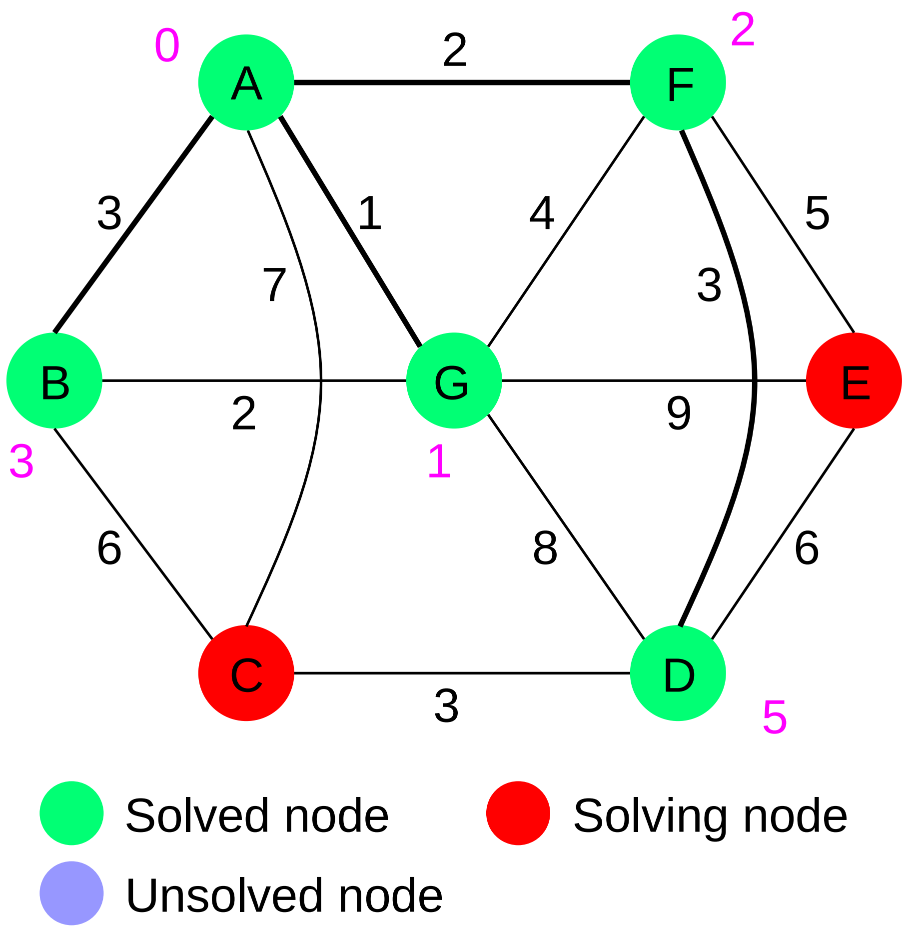
L(B) + w(B,C) = 9

Next vertice is D, set L(D) = 5, add D to set S:

S = {A,G,F,B,D}

Dijkstras Algorithm: Example

Step 5



Check all paths from A, G, F, B, D to adjiacent vertices: E, C.

$\boxed{L(A) + w(A,C) = 7}$

L(G) + w(G,E) = 10

L(F) + w(F,E) = 7

L(B) + w(B,C) = 9

L(D) + w(D,C) = 8
L(D) + w(D,E) = 11

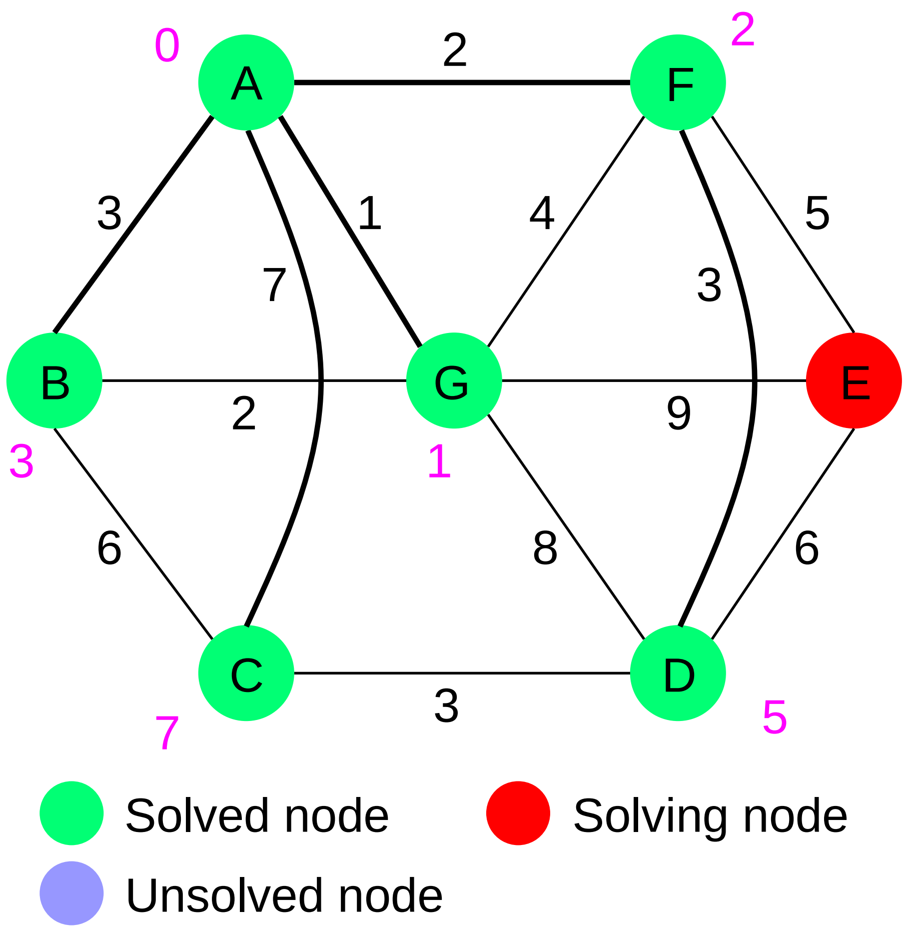
Next vertice is C, set L(C) = 7, add C to set S:

S = {A,G,F,B,D,C}

Dijkstras Algorithm: Example

Step 6

Check all paths from A, G, F, B, D, C to adjiacent vertices: E.



L(G) + w(G,E) = 10

$\boxed{\textbf{L(F) + w(F,E) = 7}}$

L(D) + w(D,E) = 11

Next vertice is E, set L(E) = 7, add E to set S:

S = {A,G,F,B,D,C,E}

Dijkstras Algorithm: Example

n	A	G	F	B	D	C	E	Shortest Path
0	**0**	☜	☜	☜	☜	☜	☜	**A**
1	0	**(1,A)**	(2,A)	(3,A)	☜	(7,A)	☜	A,**G**
2	0	(1,A)	**(2,A)**	(3,A)	(9,G)	(7,A)	(10,G)	A,G,**F**
3	0	(1,A)	(2,A)	**(3,A)**	(5,F)	(7,A)	(7,F)	A,G,F,**B**
4	0	(1,A)	(2,A)	(3,A)	**(5,F)**	(7,A)	(7,F)	A,G,F,B,**D**
5	0	(1,A)	(2,A)	(3,A)	(5,F)	**(7,A)**	(7,F)	A,G,F,B,D,**C**
6	0	(1,A)	(2,A)	(3,A)	(5,F)	(7,A)	**(7,F)**	A,G,F,B,D,C,**E**
	0	(1,A)	(2,A)	(3,A)	(5,F)	(7,A)	(7,F)	**A,G,F,B,D,C,E**

Denote:

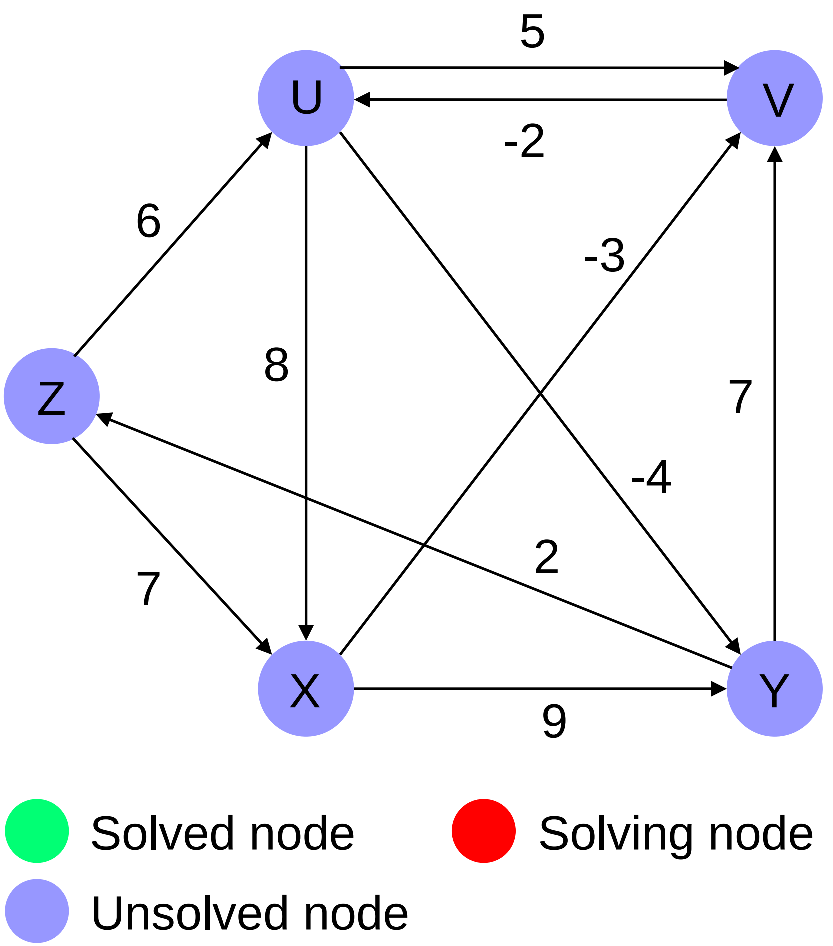
- (L,P): where L is label of the vertice, and P is preceding vertice in the path.

BELLMAN-FORD ALGORITHM

Bellman-Ford Algorithm

- This algorithm can work with negative edge weights.
- Can detect a negative cycle.
- Bellman-Ford runs in O(|n|-|m|) where n vertices and m edges.

Bellman-Ford Algorithm: Example



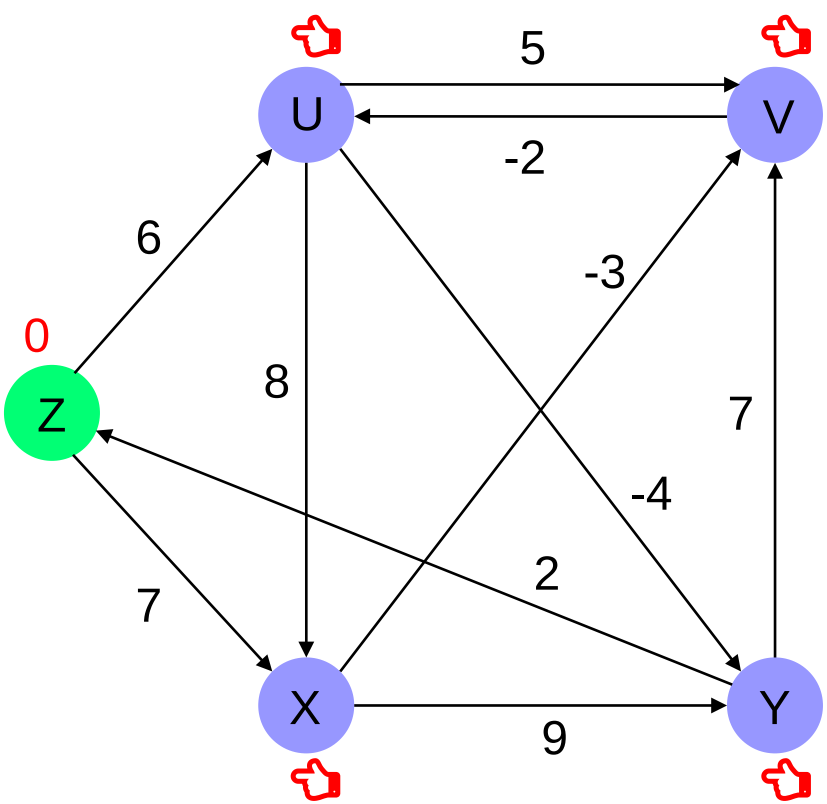
Find path from Z to all other vertices?

Denote:

- w(A,B) = weighting coefficient in the path from A to B. So, from the graph, we have w(A,B) = 3.

Bellman-Ford Algorithm: Example

Step 0

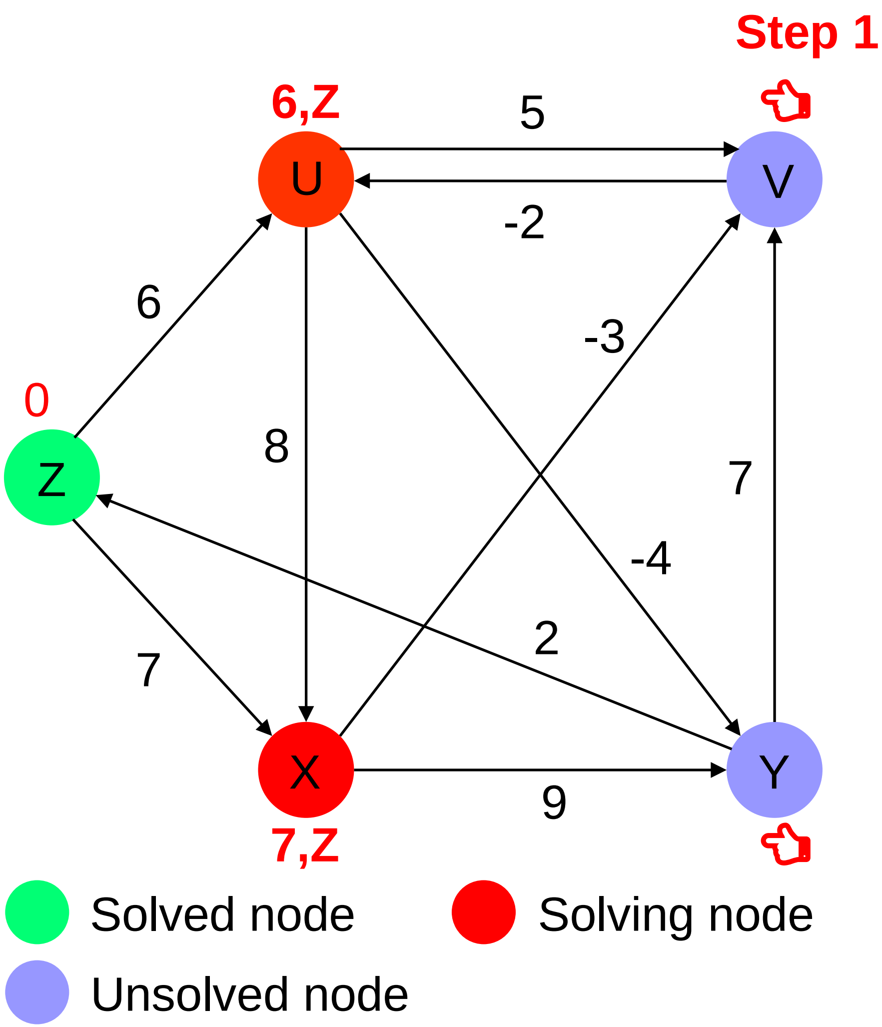


n	Z	U	X	V	Y
0	0				

Solved node

Solving node

Unsolved node



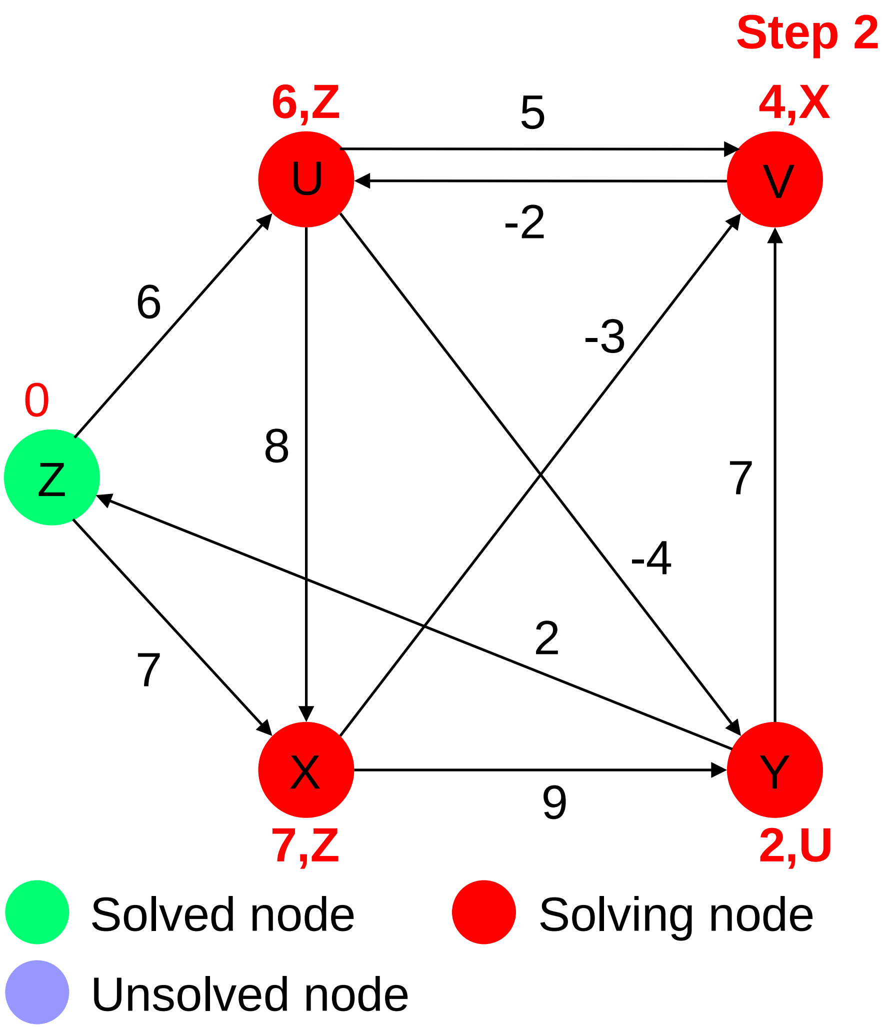
n	Z	U	X	V	Y
0	0				
1	0	6,Z	7,Z		

From Z, can go to U, and X. Then assign labels to U, X by adding cost of Z (in previous step – step 0) to w(Z,U) and w(Z,X). We denote

$L_1(U) = 6; P_1(U) = Z$

$L_1(X) = 7; P_1(X) = Z$

Bellman-Ford Algorithm: Example



n	Z	U	X	V	Y
0	0	∞	∞	∞	∞
1	0	6,Z	7,Z	∞	∞
2	0	6,Z	7,Z	4,X	2,U

Recalculate U,X,V,Y by (L,P) in step 2:

$L_2(U) = \min\{L_1(Z)+w(Z,U), L_1(V)+w(V,U)\}$

$= \min\{0+6, \infty\} = 6;\ P_2(U) = Z$

$L_2(V) = \min\{L_1(U)+w(U,V), L_1(X)+w(X,V)\}$

$= \min\{6+5, 7-3\} = 4,\ P_2(V) = X$

$L_2(Y) = \min\{L_1(U)+w(U,Y), L_1(X)+w(X,Y),$

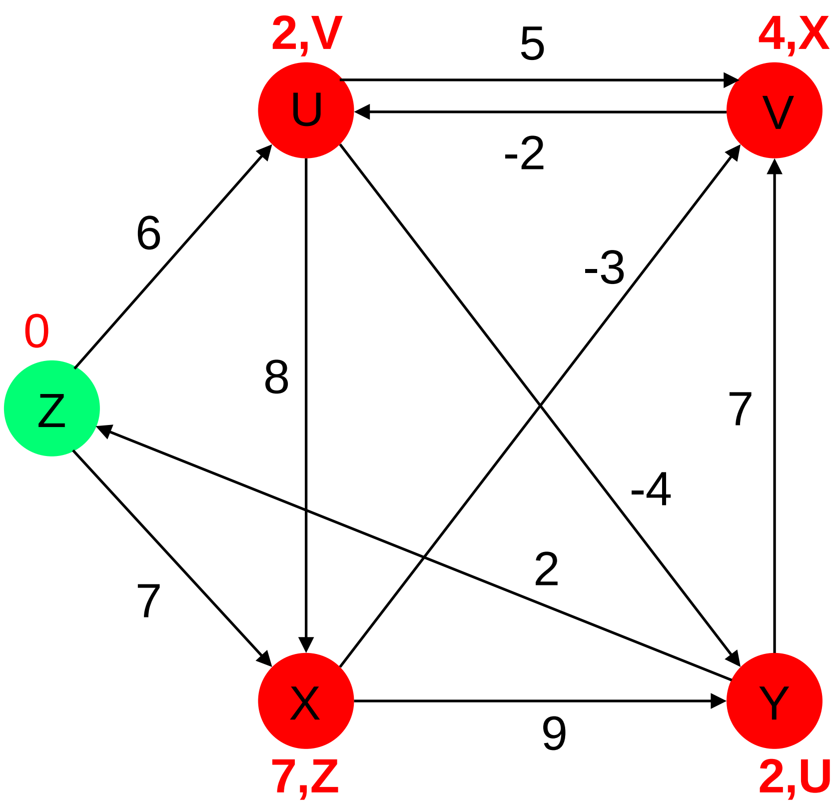
$= \min\{6-4, 7+9\} = 2;\ P_2(Y) = U$

$L_2(X) = \min\{L_1(U)+w(U,X), L_1(Z)+w(Z,X),$

$= \min\{6+8, 0+7\} = 7;\ P_2(X) = Z$

Bellman-Ford Algorithm: Example

Step 3



Solved node

Solving node

Unsolved node

n	Z	U	X	V	Y
0	0	∞	∞	∞	∞
1	0	6,Z	7,Z	∞	∞
2	0	6,Z	7,Z	4,X	2,U
3	0	2,V	7,Z	4,X	2,U

Recalculate U,X,V,Y by (L,P) in step 3:

$L_3(U) = \min\{L_2(Z)+w(Z,U), L_2(V)+w(V,U)\}$

$= \min\{0+6, 4-2\} = 2$; $P_3(U) = 2,V$

$L_3(V) = \min\{L_2(U)+w(U,V), L_2(X)+w(X,V)\}$

$= \min\{6+5, 7-3\} = 4$, $P_3(V) = 4,X$

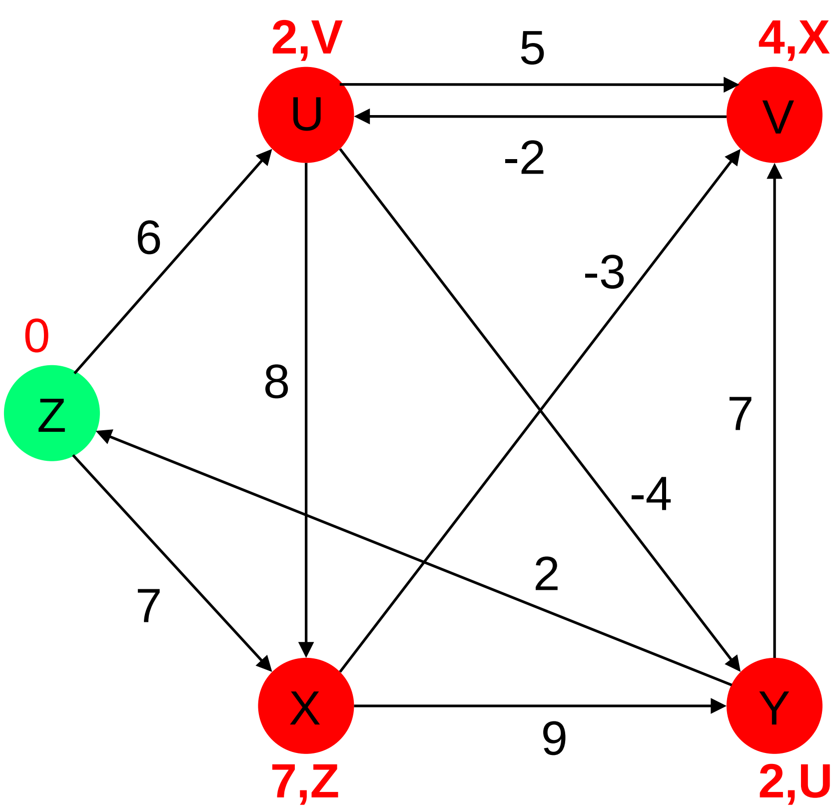
$L_3(Y) = \min\{L_2(U)+w(U,Y), L_2(X)+w(X,Y)$,

$= \min\{6-4, 7+9\} = 2$; $P_3(Y) = 2,U$

$L_3(X) = \min\{L_2(U)+w(U,X), L_2(Z)+w(Z,X)$,

$= \min\{6+8, 0+7\} = 7$; $P_3(X) = 7,Z$

Step 4



Solved node

Solving node

Unsolved node

n	Z	U	X	V	Y
0	0	∞	∞	∞	∞
1	0	6,Z	7,Z	∞	∞
2	0	6,Z	7,Z	4,X	2,U
3	0	2,V	7,Z	4,X	2,U
4	0	2,V	7,Z	4,X	-2,U

Recalculate U,X,V,Y by (L,P) in step 3:

$L_4(U) = \min\{L_3(Z)+w(Z,U), L_3(V)+w(V,U)\}$

$= \min\{0+6, 4-2\} = 2$; $P_4(U) = V$

$L_4(V) = \min\{L_3(U)+w(U,V), L_3(X)+w(X,V)\}$

$= \min\{2+5, 7-3\} = 4$, $P_4(V) = X$

$L_4(Y) = \min\{L_3(U)+w(U,Y), L_3(X)+w(X,Y),$

$= \min\{2-4, 7+9\} = -2$; $P_4(Y) = U$

$L_4(X) = \min\{L_3(U)+w(U,X), L_3(Z)+w(Z,X),$

$= \min\{2+8, 0+7\} = 7$; $P_4(X) = Z$

Summary

1. Shortest Path Algorithms
2. Dijkstras Algorithm
3. Bellman-Ford Algorithm