Design and Analysis of Algorithms

Dina El-Manakhly, Ph. D.

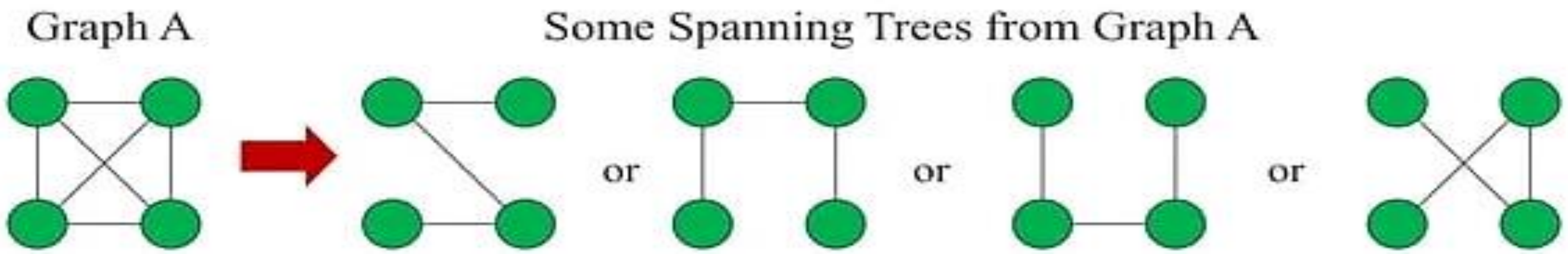
dina_almnakhly@science.suez.edu.eg

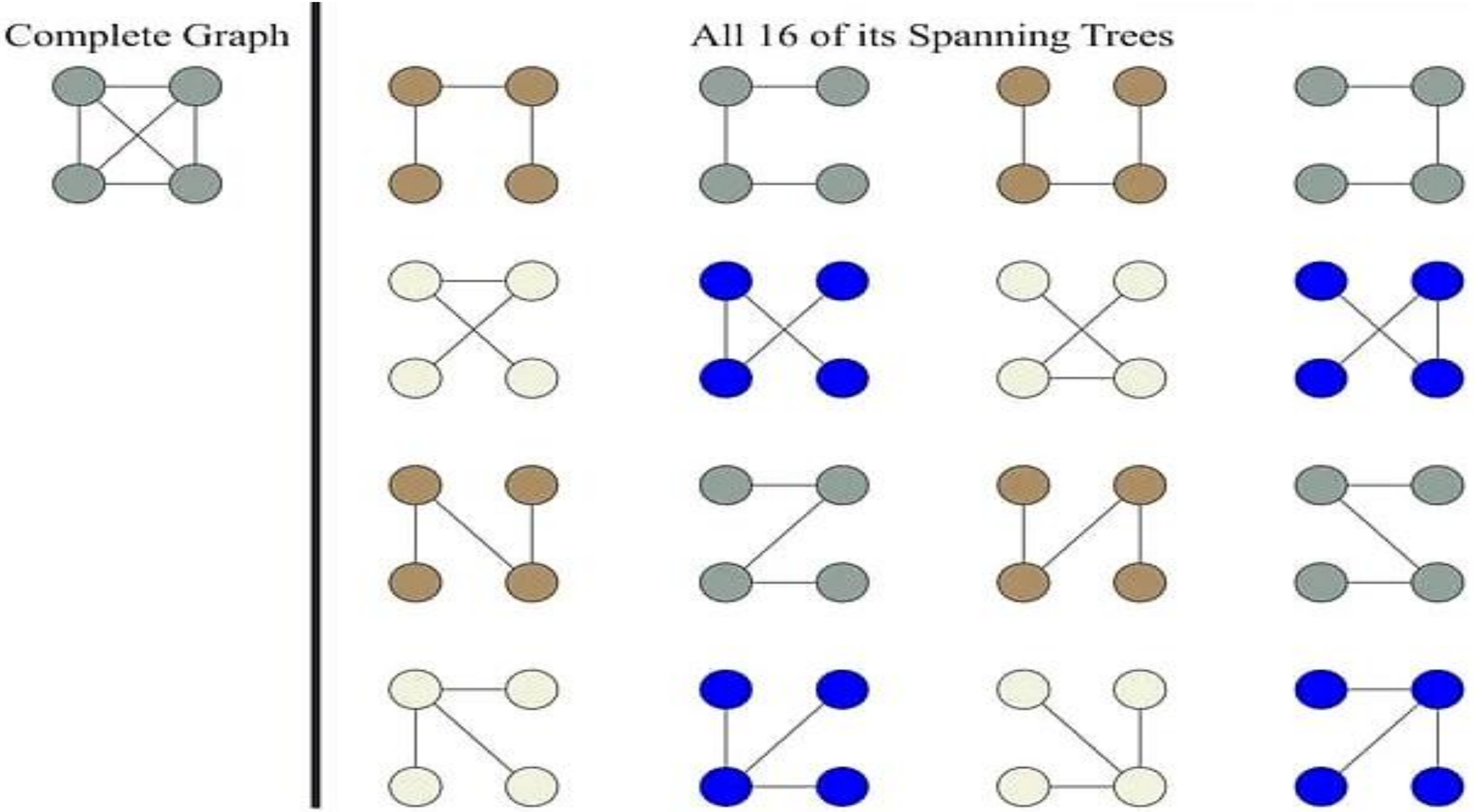
Lecture 8(4/12/2022)

Spanning Trees

A spanning tree of a graph is just a subgraph that contains all the vertices and is a tree.

A graph may have many spanning trees.




Complete Graph
All 16 of its Spanning Trees

Minimum Spanning Tree

- Formally, we are given a connected undirected graph G=(V,E)
- Each edge(u,v) has some numeric weight or cost **w(u,v)**
- We define the cost of spanning tree **T** to be the sum of the costs of edges in the spanning tree

$$\mathbf{w(T)} = \sum_{(u,v)\in T} \mathbf{w(u,v)}$$

- A **MST** is minimum of **w(T)**

Prim's Algorithm

- Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which:
  - ✓ Form a tree that includes every vertex
  - ✓ Has the minimum sum of weights among all the trees that can be formed from the graph

Prim's MST algorithm

Input Connected, undirected, weighted graph, *G*.

Output Minimum - weight spanning tree, *T*

Main Idea

(1) Start by creating two sets of vertices:

X={1} and Y={2,3,…,n}

Visited

Not Visited

(2) Grows a spanning tree, one edge at a time. On each iteration,

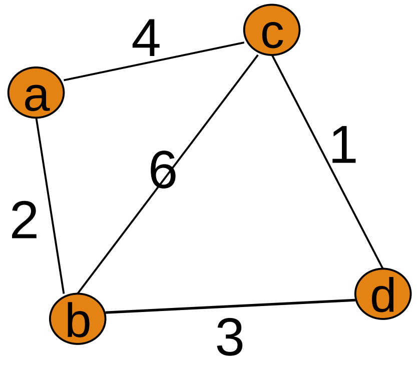
(2-1) Find an edge (x,y) of minimum weight, where $x \in X$ and $y \in Y$

(2-2) Move y from Y to X.

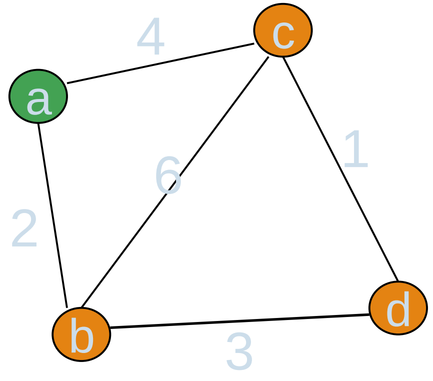
(2-3) Add the edge (x,y) to the current minimum spanning tree edges in T.

(3) Repeat Step 2 until Y becomes empty.

Example



(1) Start by creating two sets of vertices: X={a} and Y={b, c, d}

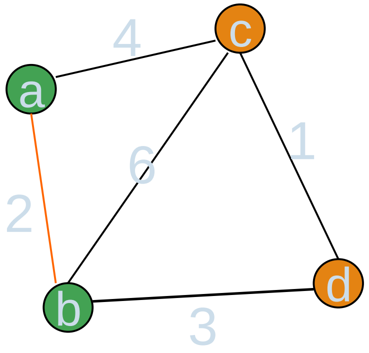


2.1 Find an edge (x,y) of minimum weight, where x∈ X and y ∈ Y

- ❑ (a,b) of weight 2
- ❑ (a,c) of weight 4

select (a,b)

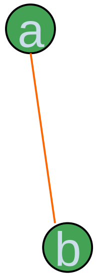
Example



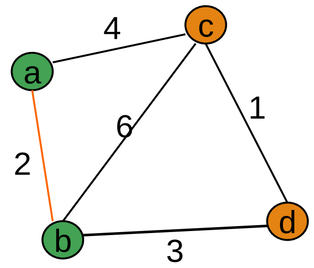
(2.2) Move y from Y to X.

X={a,b} Y={c,d}

(2.3) Add the edge (x,y) to the current minimum spanning tree edges in T.



Example



X={a,b} Y={c,d}

(2.1) Find an edge (x,y) of minimum weight, where x∊ X and y ∊ Y.

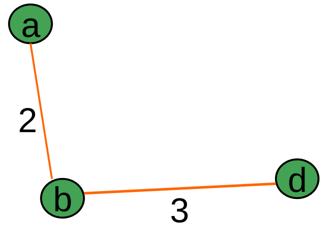
W(a,c)=4
W(b,c)=6
W(b,d)=3

➔ Select (b,d)

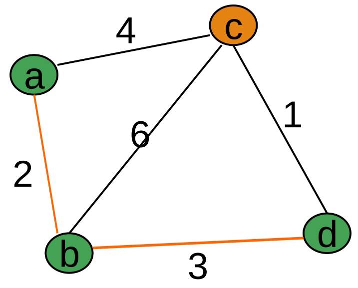
(2.2) Move y from Y to X.

X={a,b,d} Y={c}

(2.3) Add the edge (x,y) to the current minimum spanning tree edges in T.



Example



X={a,b,d} Y={c}

(2.1) Find an edge (x,y) of minimum weight, where x∈ X and y ∈ Y.

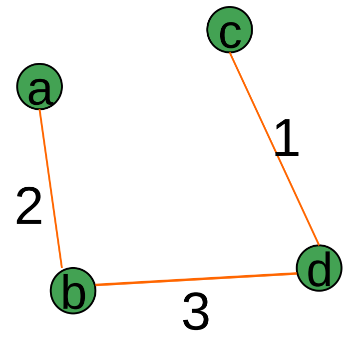
W(a,c)=4
W(b,c)=6
W(d,c)=1

➔ Select (d,c)

(2.2) Move y from Y to X.

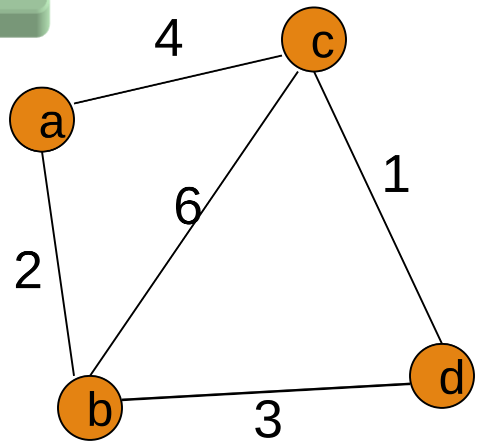
X={a,b,c, d} Y={}

(2.3) Add the edge (x,y) to the current minimum spanning tree edges in T.

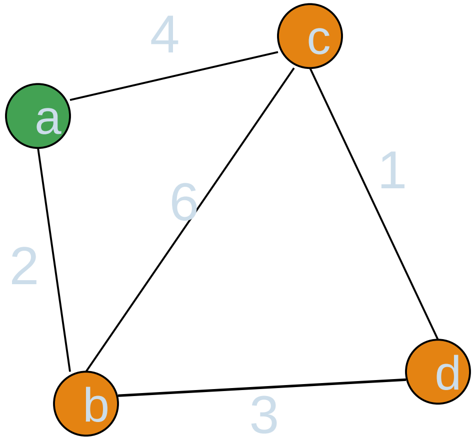


Total cost = 6

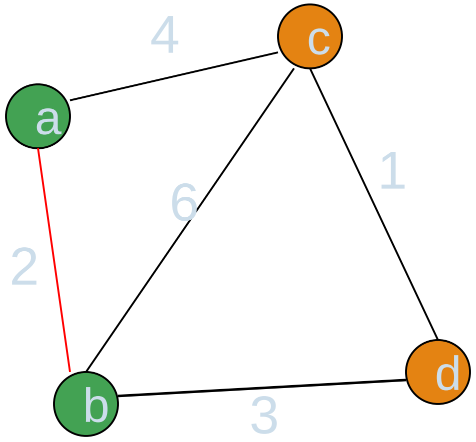
Example



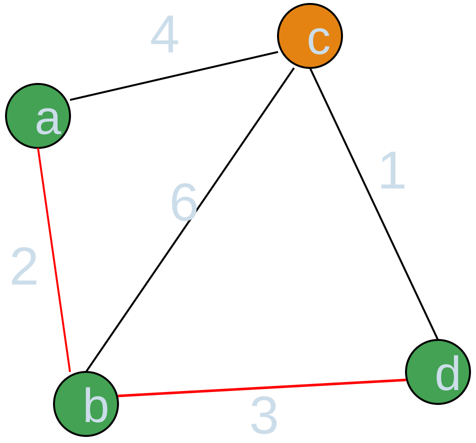
Given



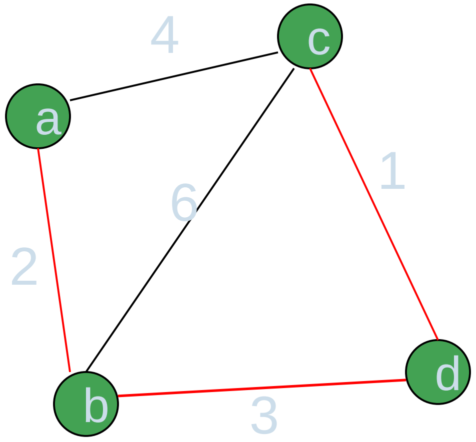
Start



1st Iteration



2nd Iteration



3rd Iteration

Algorithm

Algorithm: **Prim**

Input: A weighted connected undirected graph G=(V,E) with n vertices.

Output: The set of edges T of a minimum cost spanning tree for G.

Begin

1. T={}; X={1}; Y=V-X

2. While Y ≠ { } do

    Let (x,y) be of minimum weight such that x ϵ X and y ϵ Y.

    X = X U {y}

    Y = Y – {y}

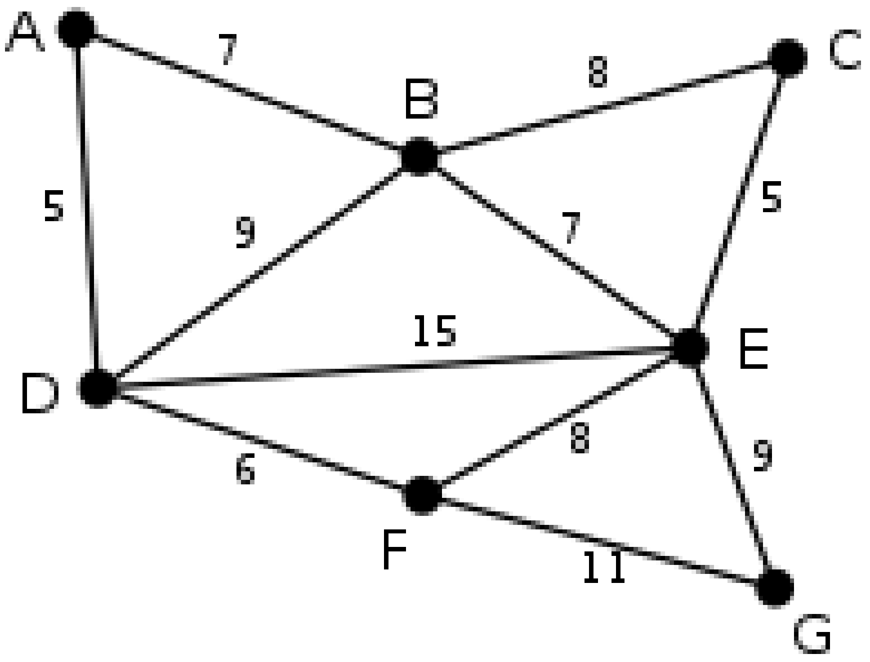
    T = T U {(x,y)}

End.

Running time: $O(n^2)$, why?

For every vertex in the set X, we have to find the all adjacent vertices, The time complexity is $O(n*(n-1))=O(n^2)$

Apply Prim's algorithm on this graph (Assignment)



Dijkstra's algorithm

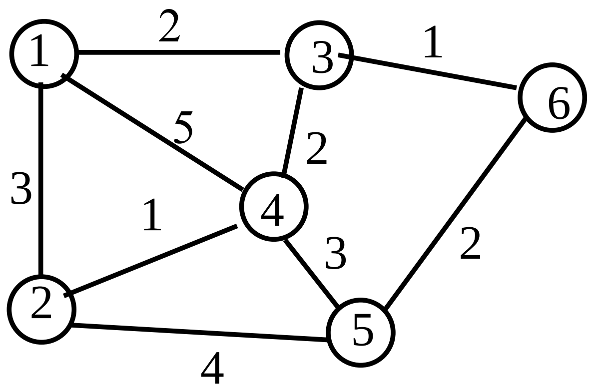
Single Source Shortest Paths Problem:

- Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.
- It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

Dijkstra's algorithm

Dijkstra's Algorithm works on the basis that any sub path B -> D of the shortest path A -> D between vertices A and D is also the shortest path between vertices B and D.

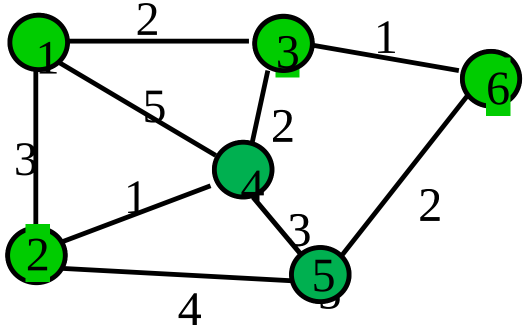
Trace of Dijkstra's algorithm



We need to calculate the shortest path between the vertex "1" and all other vertices.

1^{st} step: calculate the direct distance between the vertex "1" and all other vertices, D_v. If no direct edge between the vertex "1" and any vertex,v, then the distance equals $\propto$, $D_v = \propto$.

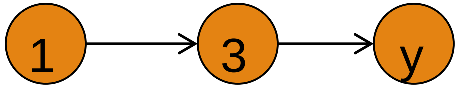
Iteration	**X**	**Y**	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4,5,6}	3	2	5	$\propto$	$\propto$



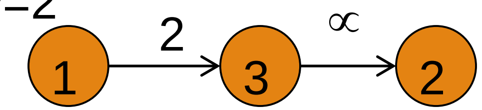
2^{nd} step: (i)select the vertex $y \in Y$ such that D_y is minimum. y=3

2^{nd} step: (ii) Update the distance from the vertex "1" to every veterx via the vertex y (selected)

Iteration	X	Y	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4,5,6}	3	2	5	$\propto$	$\propto$
1	{1,3}	{2,4,5,6}	3	2	4	$\propto$	3

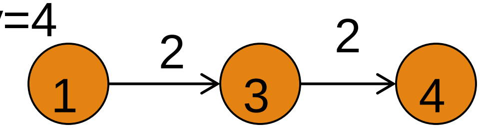


y=2



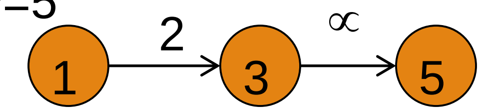
$\propto$ <3 ?

y=4



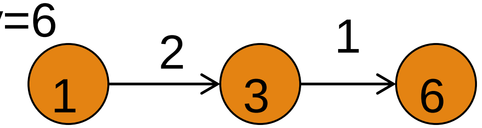
4 <5?

y=5



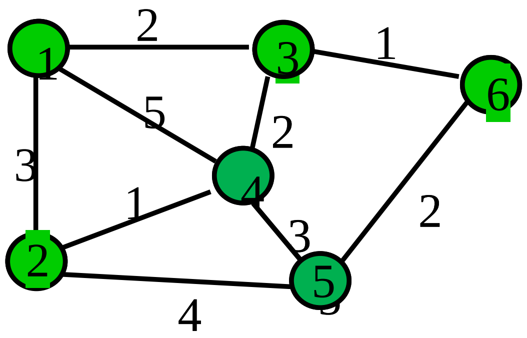
$\propto$ <$\propto$?

y=6



3 <$\propto$?

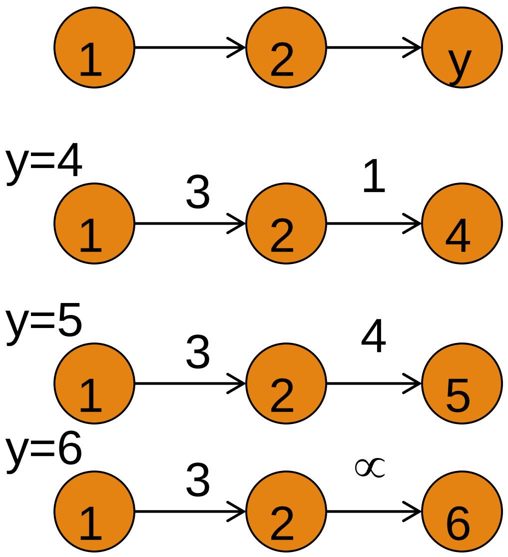
Trace of Dijkstra's algorithm



2nd step: (i)select the vertex $y \in Y$ such that D_y is minimum. y=2

2nd step: (ii) Update the distance from the vertex "1" to every veterx via the vertex y (selected)

Iteration	**X**	**Y**	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4,5,6}	3	2	5	$\propto$	$\propto$
1	{1,3}	{2,4,5,6}	3	2	4	$\propto$	3
2	{1,2,3}	{4,5,6}	3	2	4	7	3

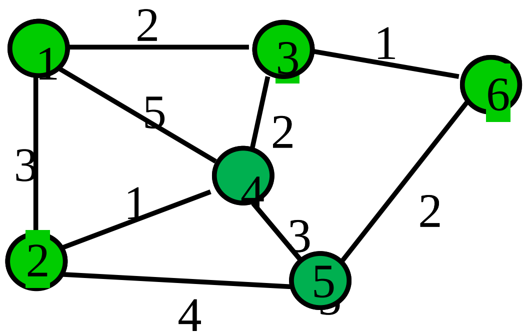


4 <4 ?

$7<\propto$?

$\propto <3$?

Trace of Dijkstra's algorithm



2^{nd} step: (i)select the vertex $y \in Y$ such that D_y is minimum. y=6

2^{nd} step: (ii) Update the distance from the vertex "1" to every veterx via the vertex y (selected)

Iteration	X	Y	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4,5,6}	3	2	5	$\propto$	$\propto$
1	{1,3}	{2,4,5,6}	3	2	4	$\propto$	3
2	{1,2,3}	{4,5,6}	3	2	4	7	3
3	{1,2,3,6}	{4,5}	3	2	4	5	3

$1 \rightarrow 6 \rightarrow y$

y=4

$1 \xrightarrow{3} 6 \xrightarrow{\propto} 4$

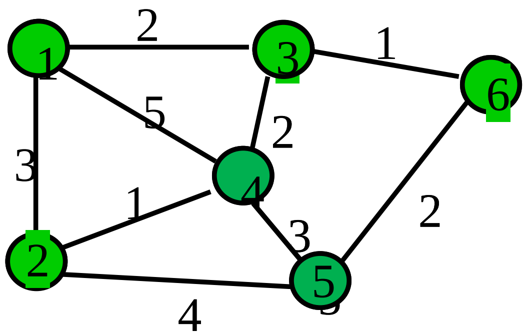
$\propto < 4$?

y=5

$1 \xrightarrow{3} 6 \xrightarrow{2} 5$

5<7?

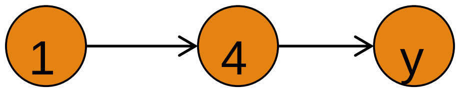
Trace of Dijkstra's algorithm



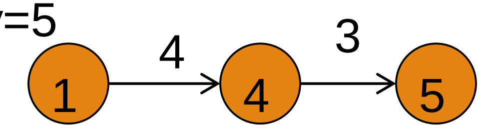
2^{nd} step: (i)select the vertex $y \in Y$ such that D_y is minimum. y=4

2^{nd} step: (ii) Update the distance from the vertex "1" to every veterx via the vertex y (selected)

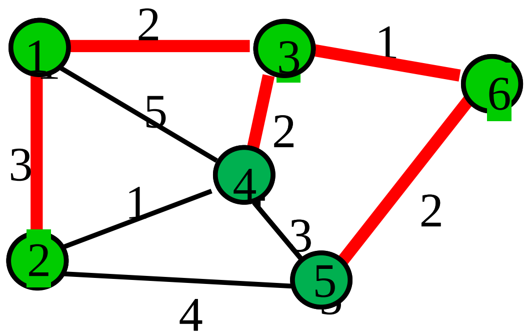
Iteration	X	Y	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4,5,6}	3	2	5	$\propto$	$\propto$
1	{1,3}	{2,4,5,6}	3	2	4	$\propto$	3
2	{1,2,3}	{4,5,6}	3	2	4	7	3
3	{1,2,3,6}	{4,5}	3	2	4	5	3
4	{1,2,3,4,6}	{5}	3	2	4	5	3



y=5



7<5 ?



The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.

Algorithm

Algorithm: **Dijkstra**

Input: A weighted connected graph $G=(V,E)$ with n vertices.

Output: The distance from vertex 1 to every other vertex in G.

Begin

1. $X=\{1\}$; $Y=V-X$; $D[1]=0$

2. For each vertex $v \in V$ if there is an edge from 1 to v then let $D[v]=w(1,v)$.

   Otherwise, $D[v]=\infty$

2. While $Y \neq \{\ \}$ do

   Let $y \in Y$ such that $D[y]$ is minimum

   $X = X \cup \{y\}$

   $Y = Y - \{y\}$

   Update the distance (labels) of those vertices in Y that are adjacent to y.

   // for each edge (y,w): if $w \in Y$ and $D[y] + w(y,w) < D[w]$ then

   $D[w]=D[y]+w(y,w)$ //

End.

Running time: $O(n^2)$, why?