CHAPTER 4 – PART 2

TREE

Introduction

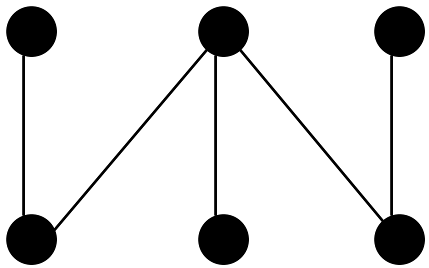
Definition 1. A tree is **a connected undirected graph with no simple circuits.**

Theorem 1. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

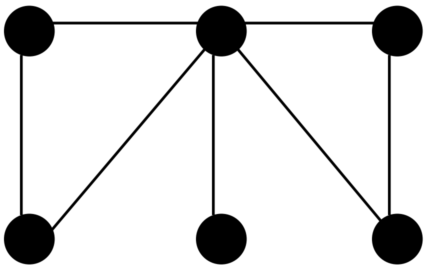
Theorem 2 . A tree with **m**-vertices has **m-1** edges

Which graphs are trees?

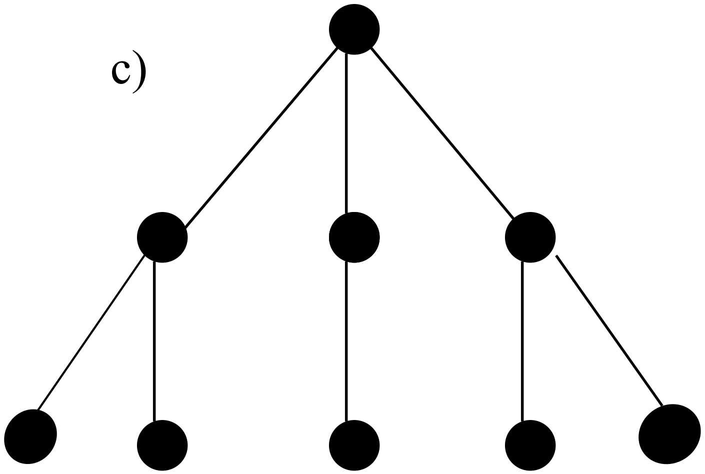
a)



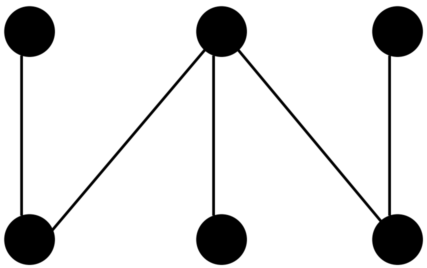
b)



c)



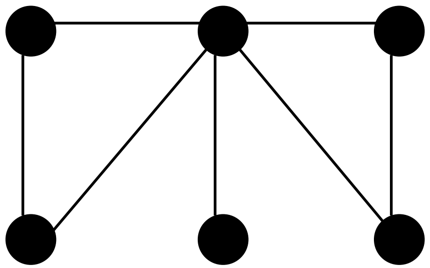
Solution



tree

vertices = 6

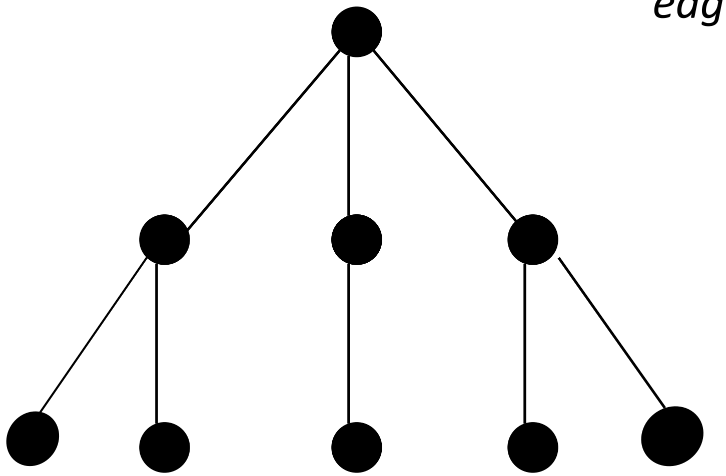
edges = 5



Not a tree

vertices = 6

edges = 7



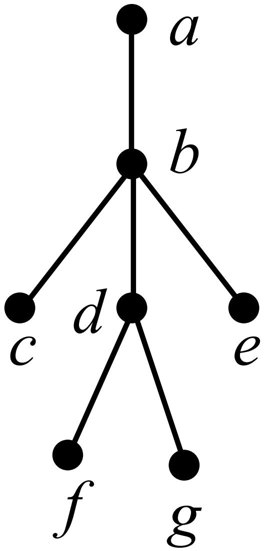
tree

vertices = 9

edges = 8

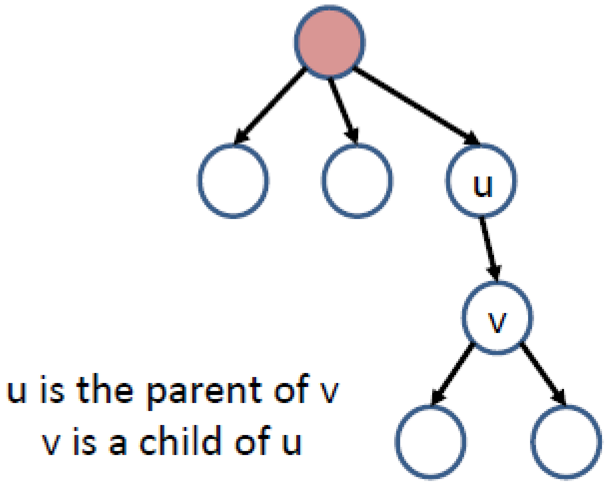
Rooted tree

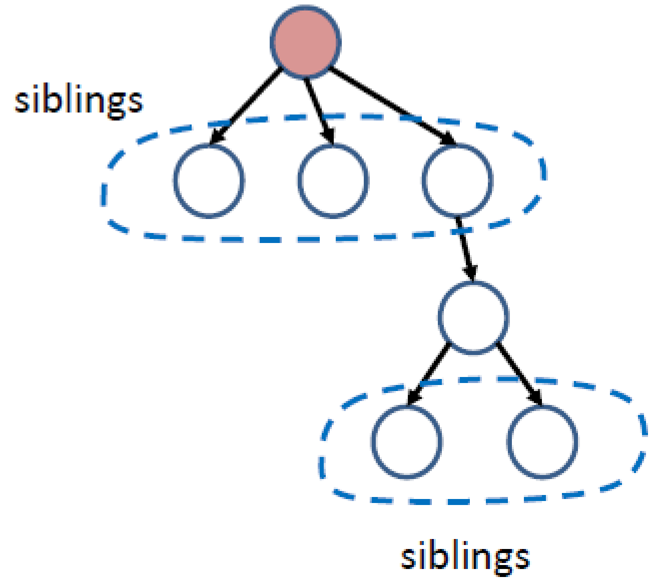
Definition 2. A **rooted tree** is a tree in which one vertex has been designed as the **root** and every edge is directed away from the root.



Rooted Tree - Terminologies

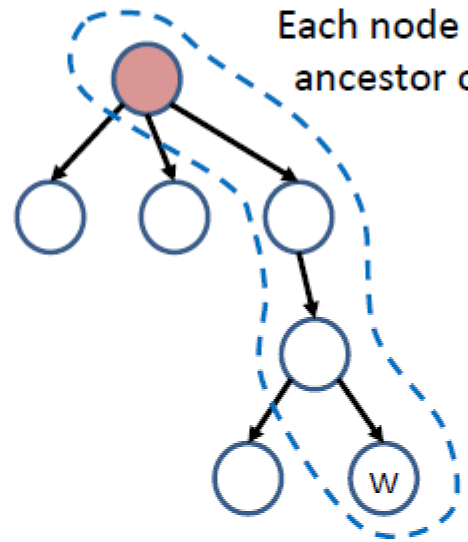
- Each edge is from a parent to a child
- Vertices with the same parent are siblings





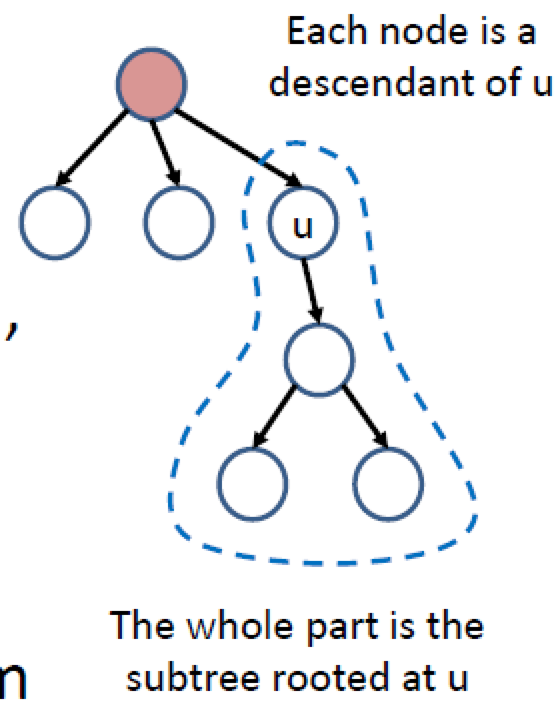
Rooted Tree - Terminologies

- The ancestors of a vertex w include all the nodes in the path from the root to w
- The proper ancestors of a vertex w are the ancestors of w, but excluding w



Rooted Tree - Terminologies

- The descendants of a vertex u include all the nodes that have u as its ancestor
- The proper descendants of a vertex u are the descendants of u, but excluding u
- The subtree rooted at u includes all the descendants of u, and all edges that connect between them



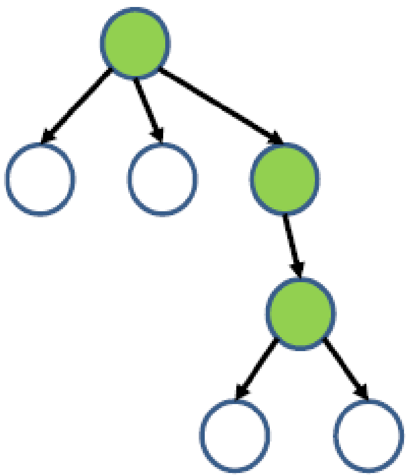
Rooted Tree - Terminologies

- Vertices with no children are called leaves ;

  Otherwise, they are called internal nodes

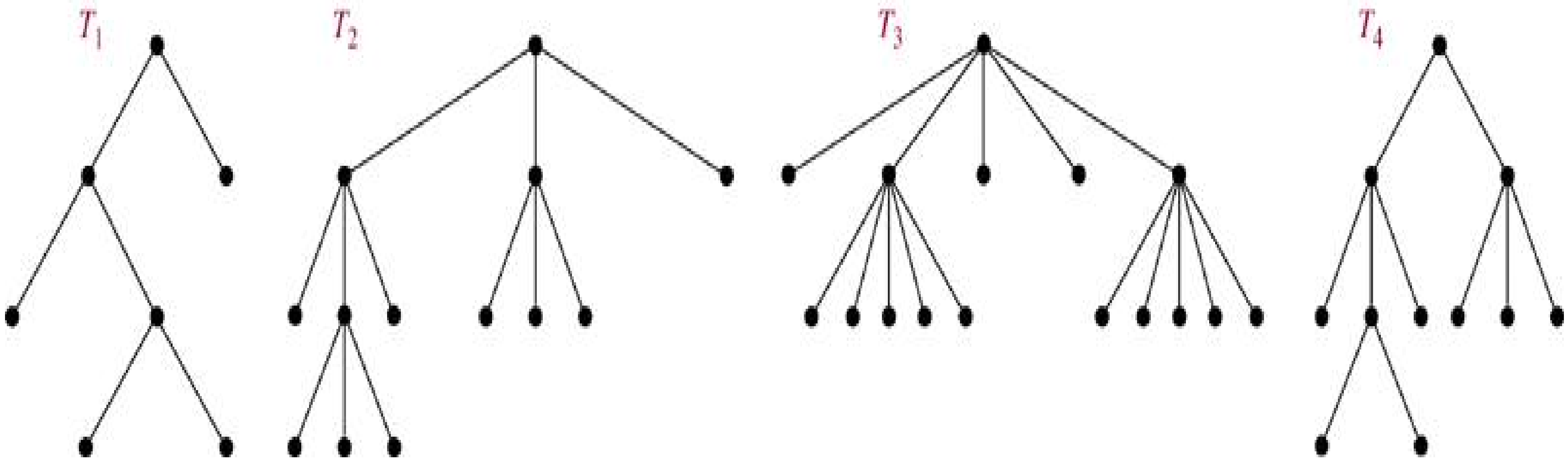
- If every internal node has no more than m children, the tree is called an m-ary tree
  - Further, if every internal node has exactly m children, the tree is a full m-ary tree

All internal nodes are colored



The tree is ternary (3-ary), but not full

Examples



full binary tree | full 3-ary tree | full 5-ary tree | not full 3-ary tree

Properties of Trees

- Theorem : A tree with n nodes has n-1 edges
- Theorem : A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

Corollary: A full m-ary tree with n vertices contains $(n-1)/m$ internal vertices, and hence $n - (n-1)/m = ((m-1)n+1)/m$ leaves

$$i = \frac{n-1}{m} \qquad l = \mathrm{n} - \frac{(n-1)}{m} = \frac{(m-1)n+1}{m}$$

Properties of Trees

Theorem – A full **m-ary** tree with

- ***n*** vertices has ***i = (n-1)/m*** internal vertices and ***l = [(m-1)n+1]/m*** leaves.

$$i = \frac{n-1}{m} \qquad l = \frac{(m-1)n+1}{m}$$

Properties of Trees

Theorem – A full **m-ary** tree with

- ***i*** internal vertices has ***n =mi+1*** vertices and ***l = (m-1)i + 1*** leaves

$$n = mi + 1 \qquad l = (m-1)i + 1$$

Properties of Trees

Theorem – A full **m-ary** tree with

- ***l*** leaves has *n =(m**l** -1)/(m-1)* vertices and *i =(**l** -1)/(m-1)* internal vertices

$$n = \frac{ml-1}{m-1} \qquad i = \frac{l-1}{m-1}$$

Example

Ex : Peter starts out a chain mail. Each person receiving the mail is asked to send it to four other people. Some people do this, and some don't

Now, there are 100 people who received the letter but did not send it out

Assuming no one receives more than one mail. How many people have sent the letter ?

Solution

- The chain letter can be represented using **4-ary** tree. The internal vertices correspond to people who sent out the letter, and the leaves correspond to people who did not send it out. Since 100 people did not send out the letter, the number of leaves in this rooted tree is, l=100. The number of people have seen the letter is *n=(4x100-1)/(4-1)=133*. The number of internal vertices is (100-1)/(4-1) or *133-100=33*, people sent the letter.

Exercise

- How many matches are played in a tennis tournament of 27 players

Exercise

Suppose 1000 people enter a chess tournament. Use a rooted tree model of the tournament to determine how many games must be played to determine a champion, if a player is eliminated after one loss and games are played until only one entrant has not lost. (Assume there are no ties.)

Solution

By Theorem 4(3) with m = 2 and l = 1000. We know that:

$$i = \frac{(1000-1)}{(2-1)}$$

$$= \frac{999}{1}$$

$$= 999$$

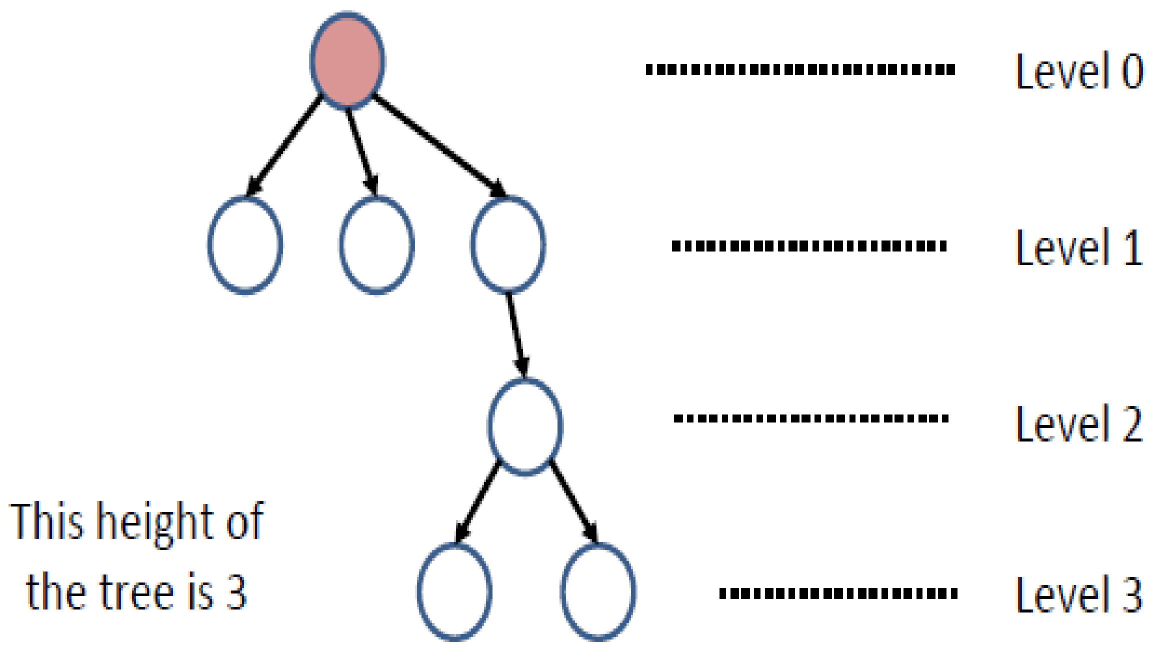
We have 999 internal vertices, so we know 999 games must be played to determine the champion.

Properties of Trees

- The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.
  The level of the root is defined to be zero.
  The height of a rooted tree is the maximum of the levels of vertices.

Example

- Ex :



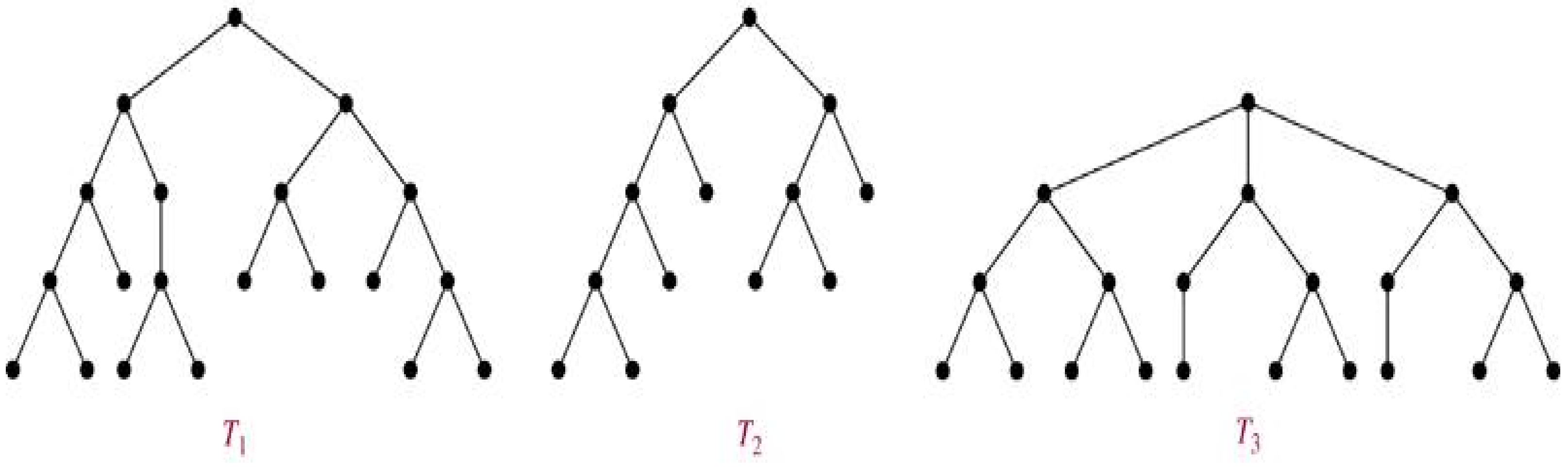
Properties of Trees

- **Definition:** A rooted m-ary tree of height h is balanced if all leaves are at levels h or $h-1$.

- **Theorem.** There are at most m^h leaves in an m-ary tree of height h.

Example

Which of the rooted trees shown below are balanced?



Sol. T_1, T_3

Tree Traversal

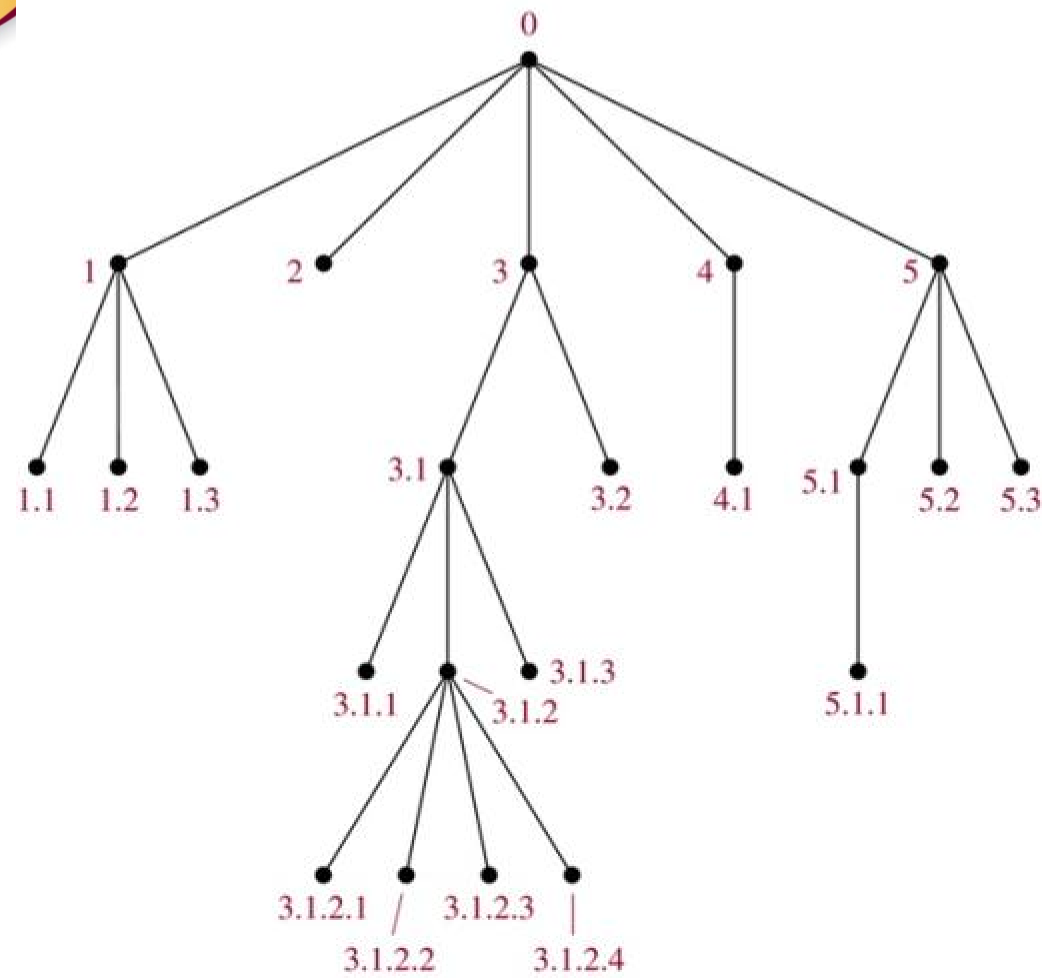
Universal Address Systems

Label vertices:

1.root $\rightarrow$ 0, its k children $\rightarrow$ $1, 2, \ldots, k$ (from left to right)
2.For each vertex v at level n with label A,
its r children $\rightarrow$ $A.1, A.2, \ldots, A.r$ (from left to right).

We can totally order the vertices using the lexicographic ordering of their labels in the universal address system.

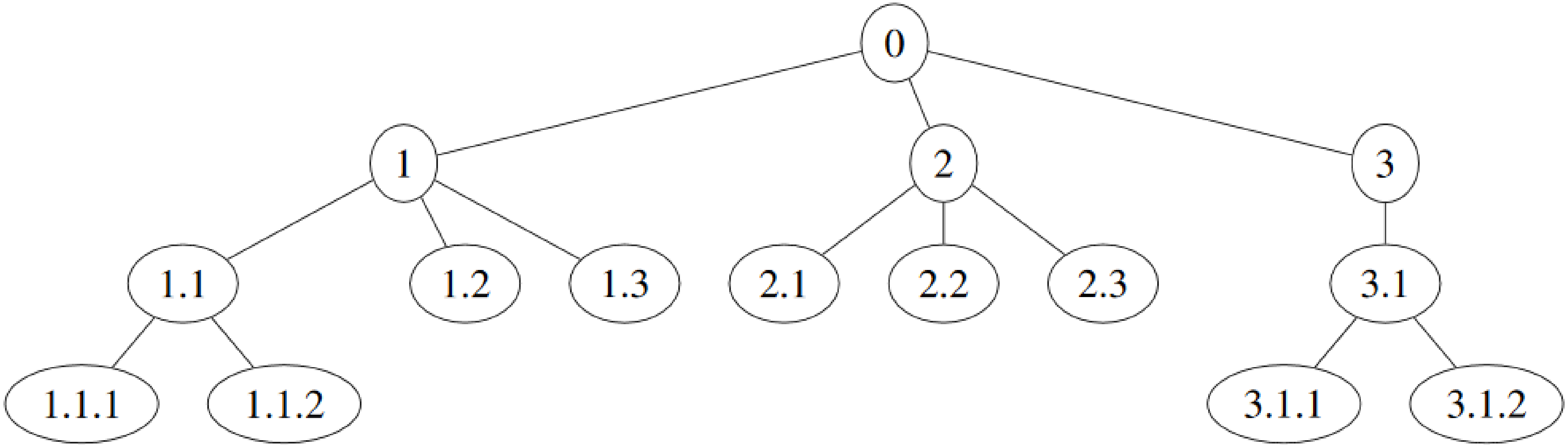
$x_1.x_2.\ldots.x_n < y_1.y_2.\ldots.y_m$
if there is an i, $0 \leq i \leq n$, with $x_1=y_1$, $x_2=y_2$, …, $x_{i-1}=y_{i-1}$, and $x_i<y_i$; or if $n<m$ and $x_i=y_i$ for $i=1, 2, \ldots, n$.



The lexicographic ordering is:

0 < 1<1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 < 3.1.2.3 < 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.1.1 < 5.2 < 5.3

Exercise



Find the lexicographic ordering of the above tree.

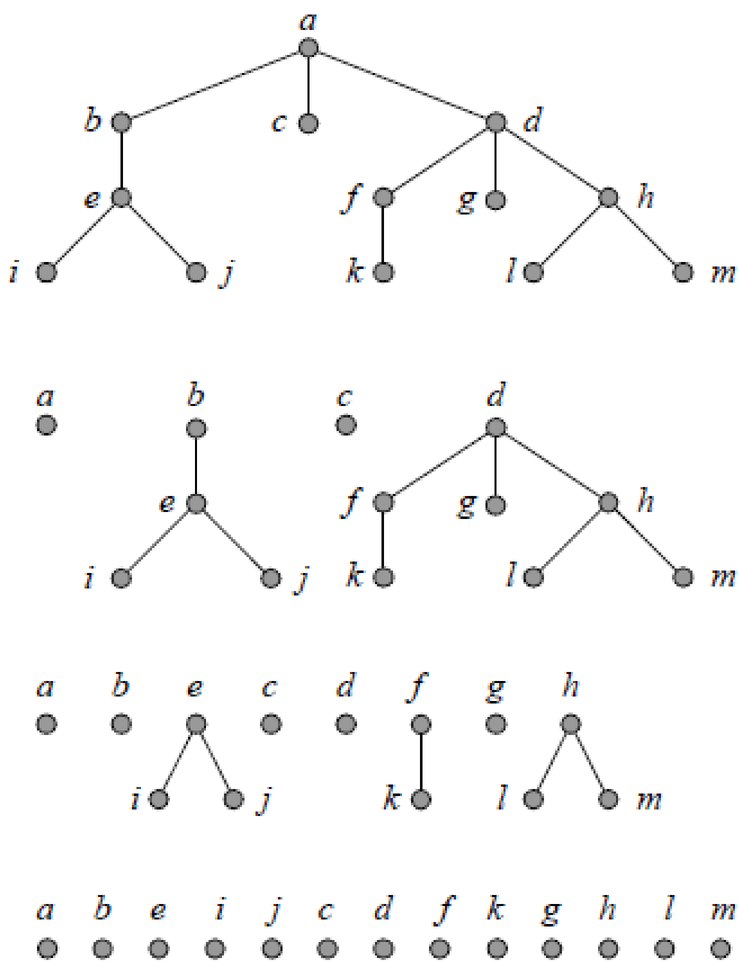
Tree Traversal

- Preorder: **root**, **left**-subtree, **right** subtree
- Inorder – **left** subtree, **root**, **right** sub-tree
- Post-order : **left** subtree, **right** sub-tree, **root**

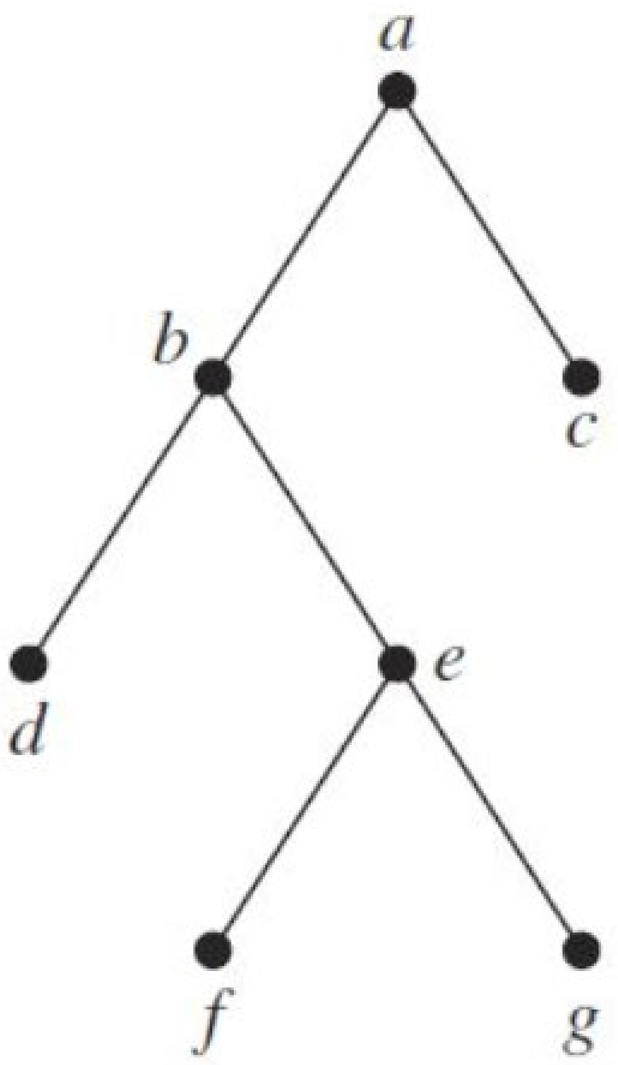
Preorder Traversal

Procedure *preorder*(T: ordered rooted tree)
r := root of T
list r
for each child c of r from left to right
begin
    $T(c)$:= subtree with c as its root
    preorder($T(c)$)
end

Preorder Traversal - Example



Exercise

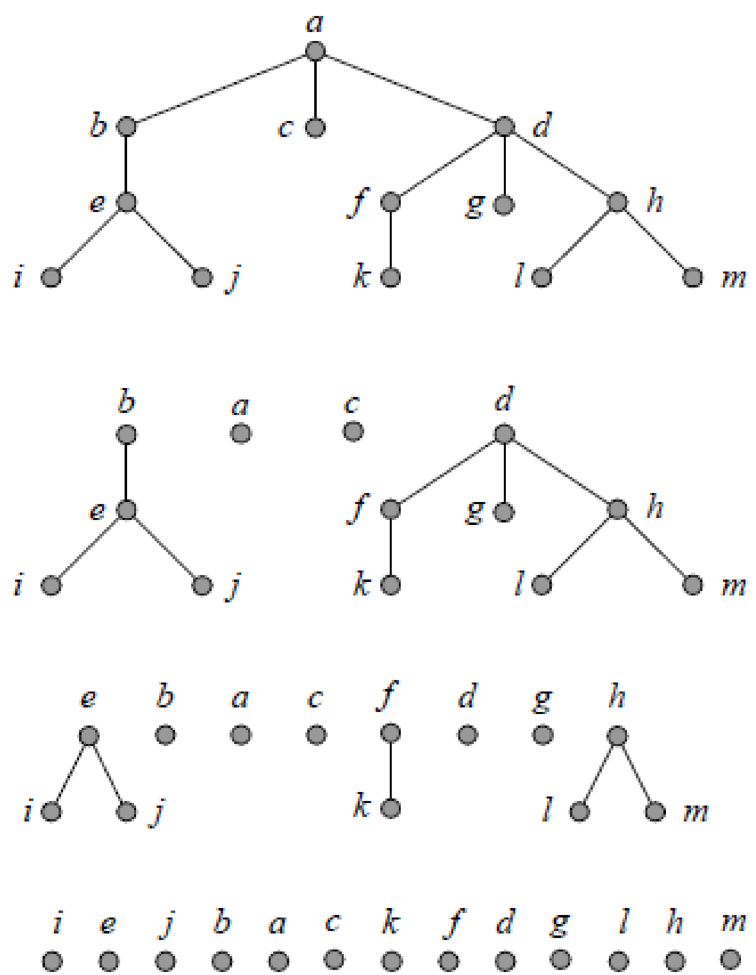


Determine the order in which a preorder traversal visits the vertices of the given ordered rooted tree.

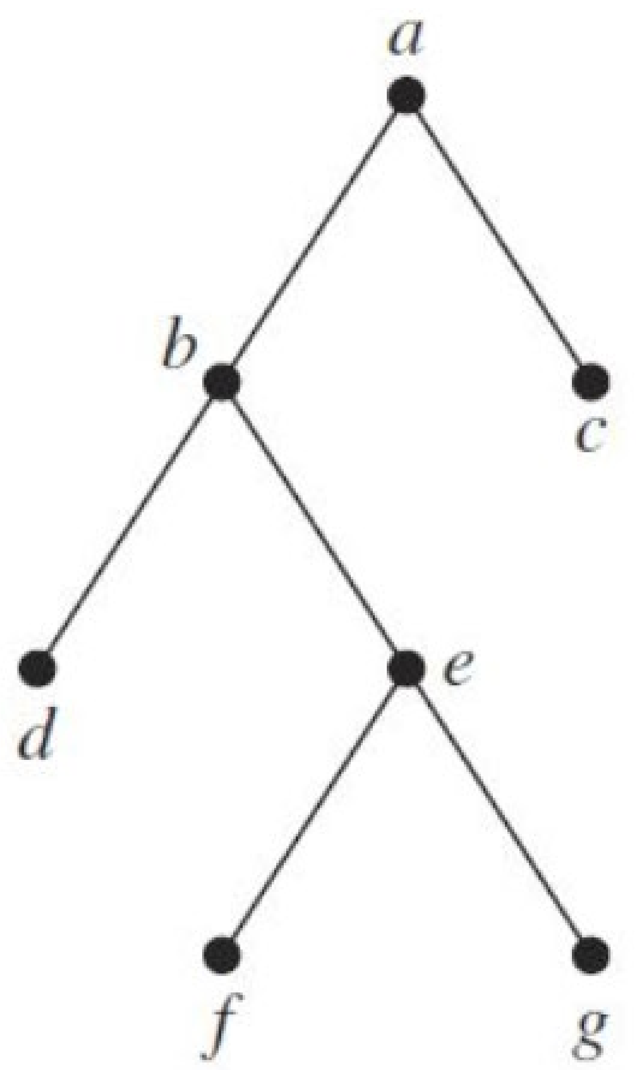
Inorder Traversal

Procedure *inorder*(T: ordered rooted tree)
r := root of T
If r is a leaf **then** list r
else
begin
    l := first child of r from left to right
    $T(l)$:= subtree with l as its root
    inorder($T(l)$)
    list r
    for each child c of r except for l from left to right
        $T(c)$:= subtree with c as its root
        inorder($T(c)$)
end

Inorder Traversal -Example



Exercise

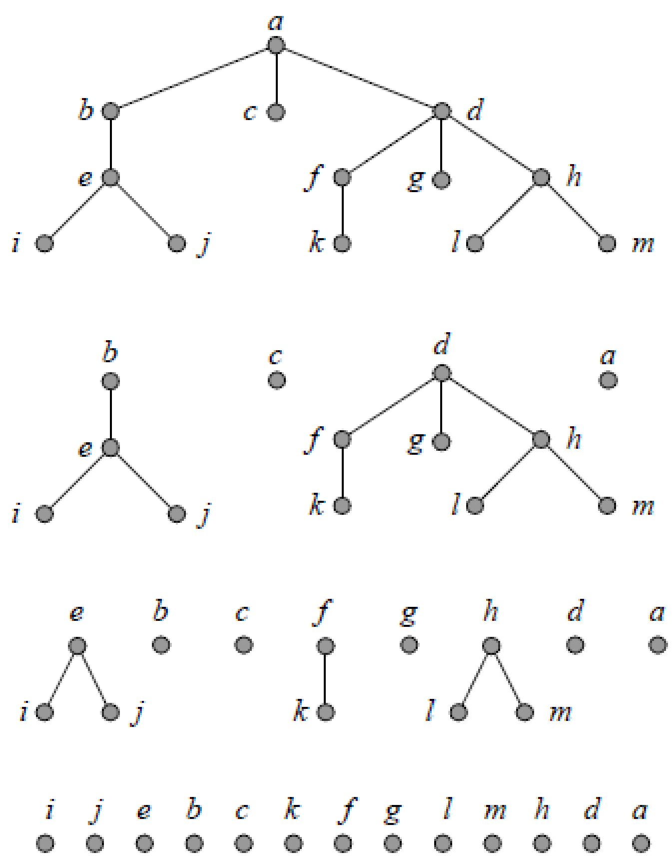


Determine the order in which a inorder traversal visits the vertices of the given ordered rooted tree.

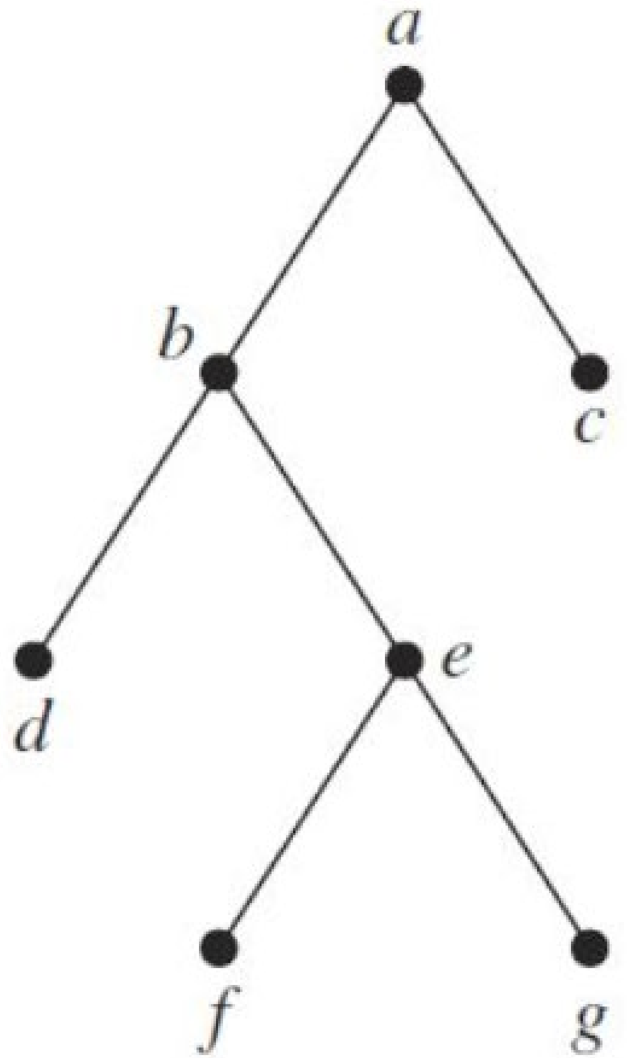
Postorder Traversal

Procedure *postorder*(T: ordered rooted tree)
r := root of T
for each child c of r from left to right
begin
    $T(c)$:= subtree with c as its root
    postorder($T(c)$)
end
list r

Postorder Traversal - Example

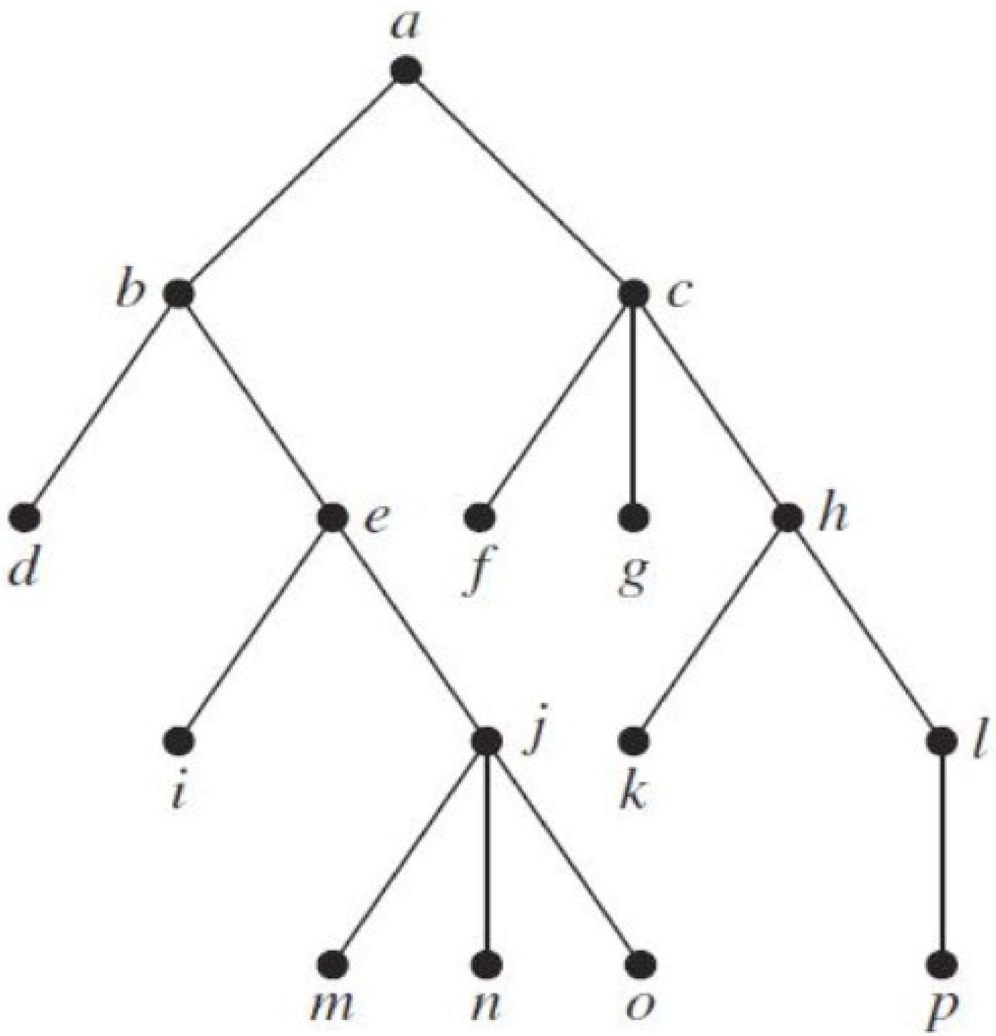


Exercise



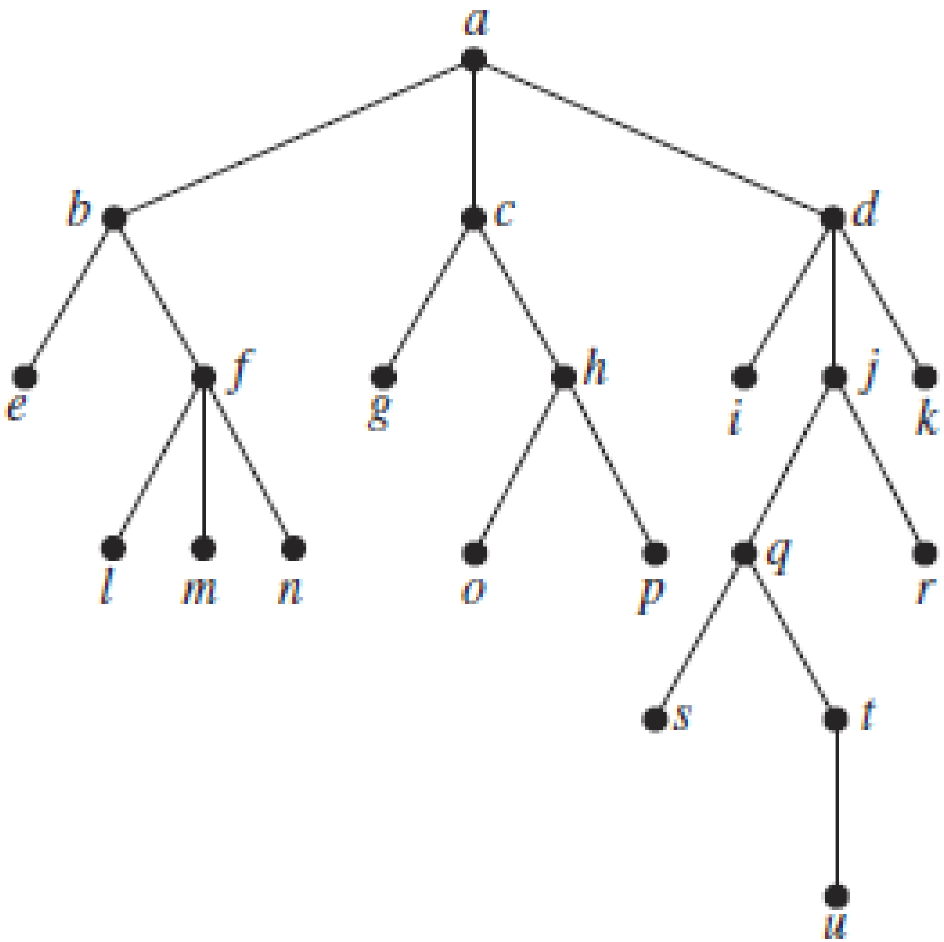
Determine the order in which a postorder traversal visits the vertices of the given ordered rooted tree.

Exercise



Determine the order of preorder, inorder and postorder of the given rooted tree.

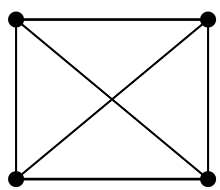
Exercise



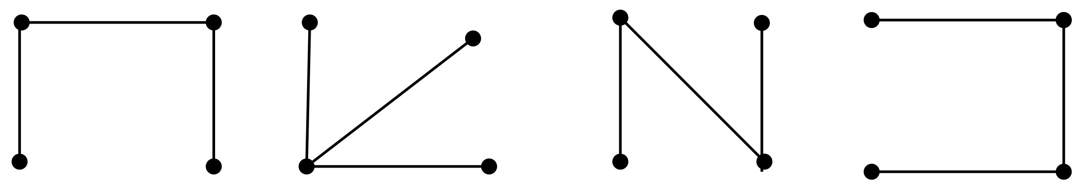
Determine the order of preorder, inorder and postorder of the given rooted tree.

Spanning Trees

- A spanning tree is a simple graph that is a subgraph of ***G*** and contains every vertex of ***G*** and is a tree.



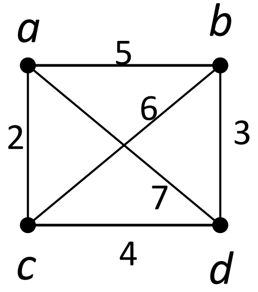
A connected undirected graph

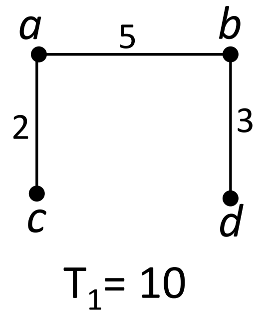


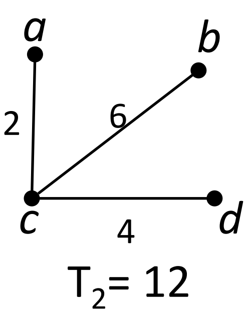
Four spanning trees of the graph

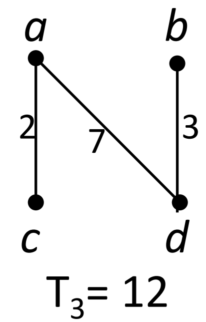
Minimum Spanning Tree (MST)

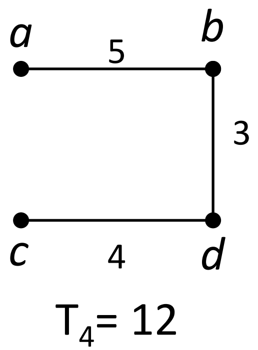
- A Minimum Spanning Tree is a spanning tree on a weighted graph that has minimum total weight.
- Example



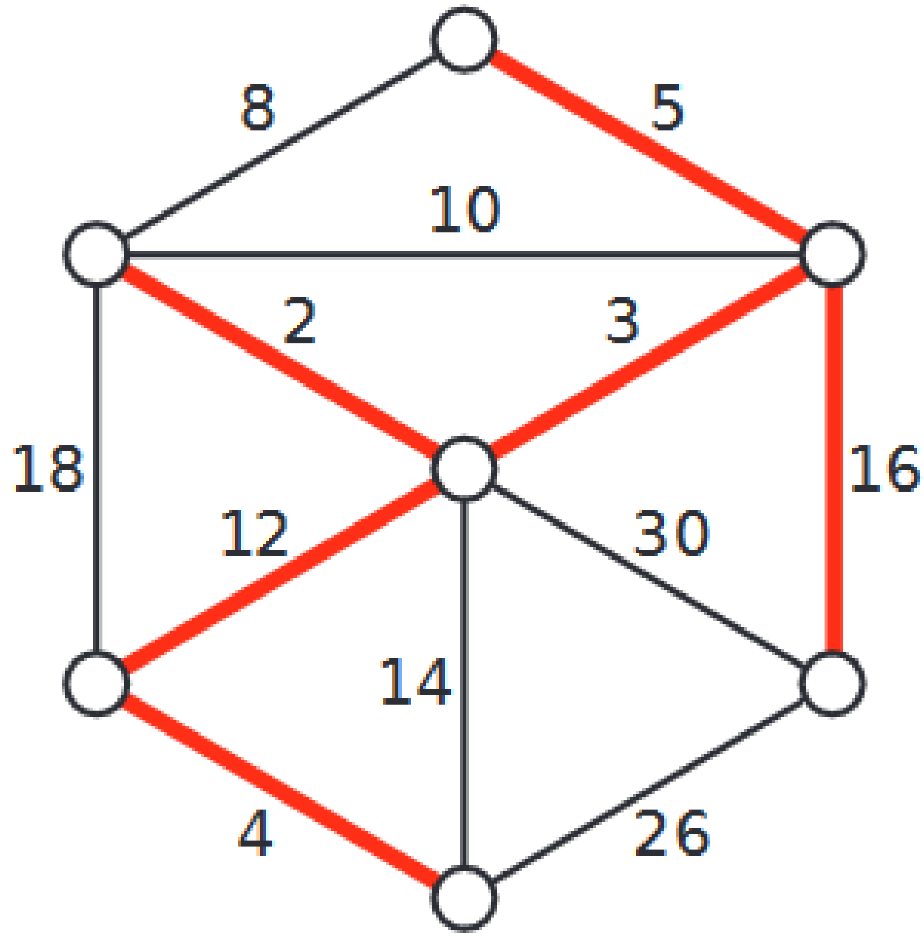








Minimum Spanning Tree (MST)



A weighted graph and its minimum spanning tree.

Muddy City Problem

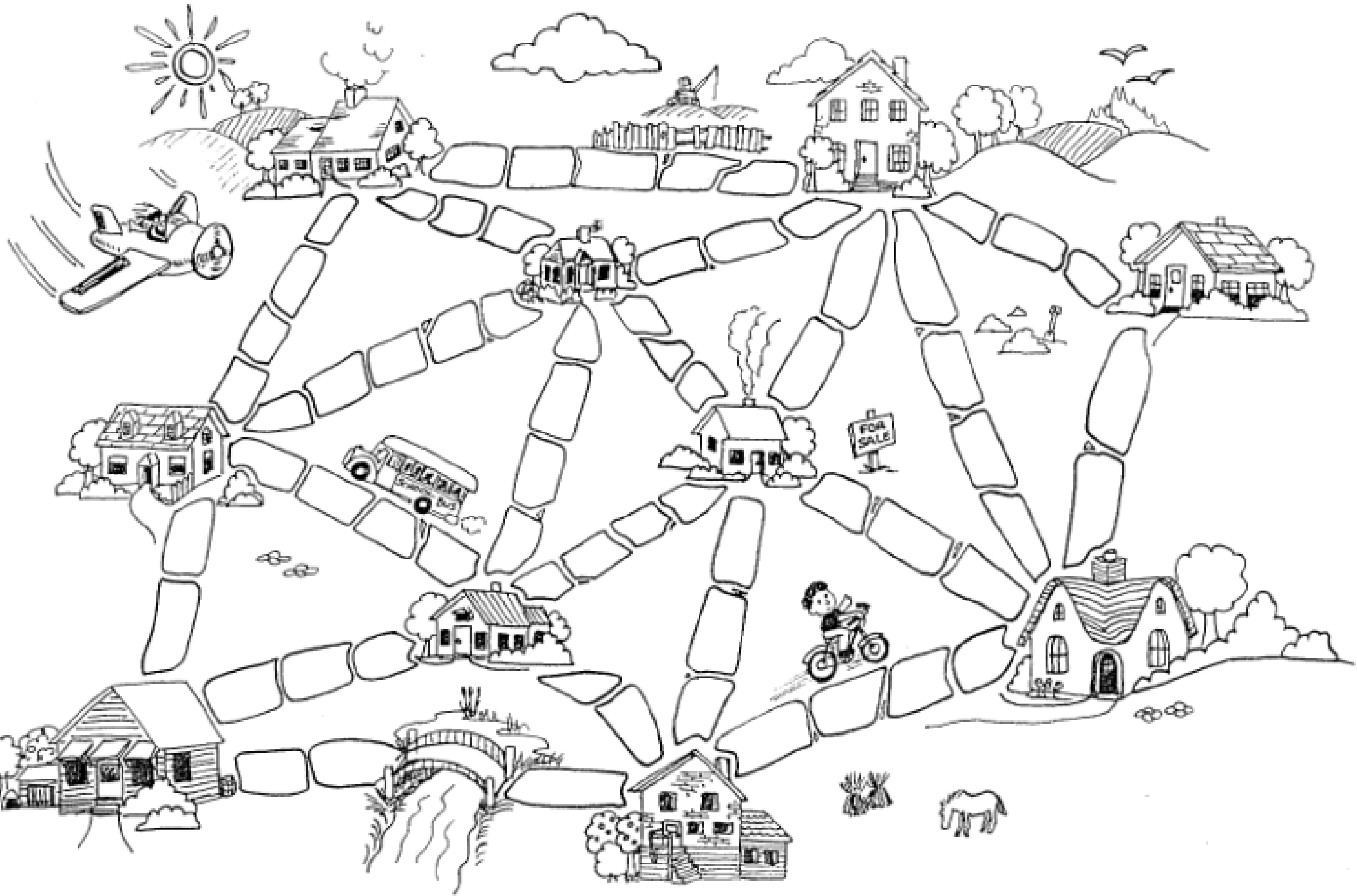
Once upon a time there was a city that had no roads. Getting around the city was particularly difficult after rainstorms because the ground became very muddy. Cars got stuck in the mud and people got their boots dirty. The mayor of the city decided that some of the streets must be paved, but didn’t want to spend more money than necessary because the city also wanted to build a swimming pool.

Muddy City Problem

The mayor therefore specified two conditions:

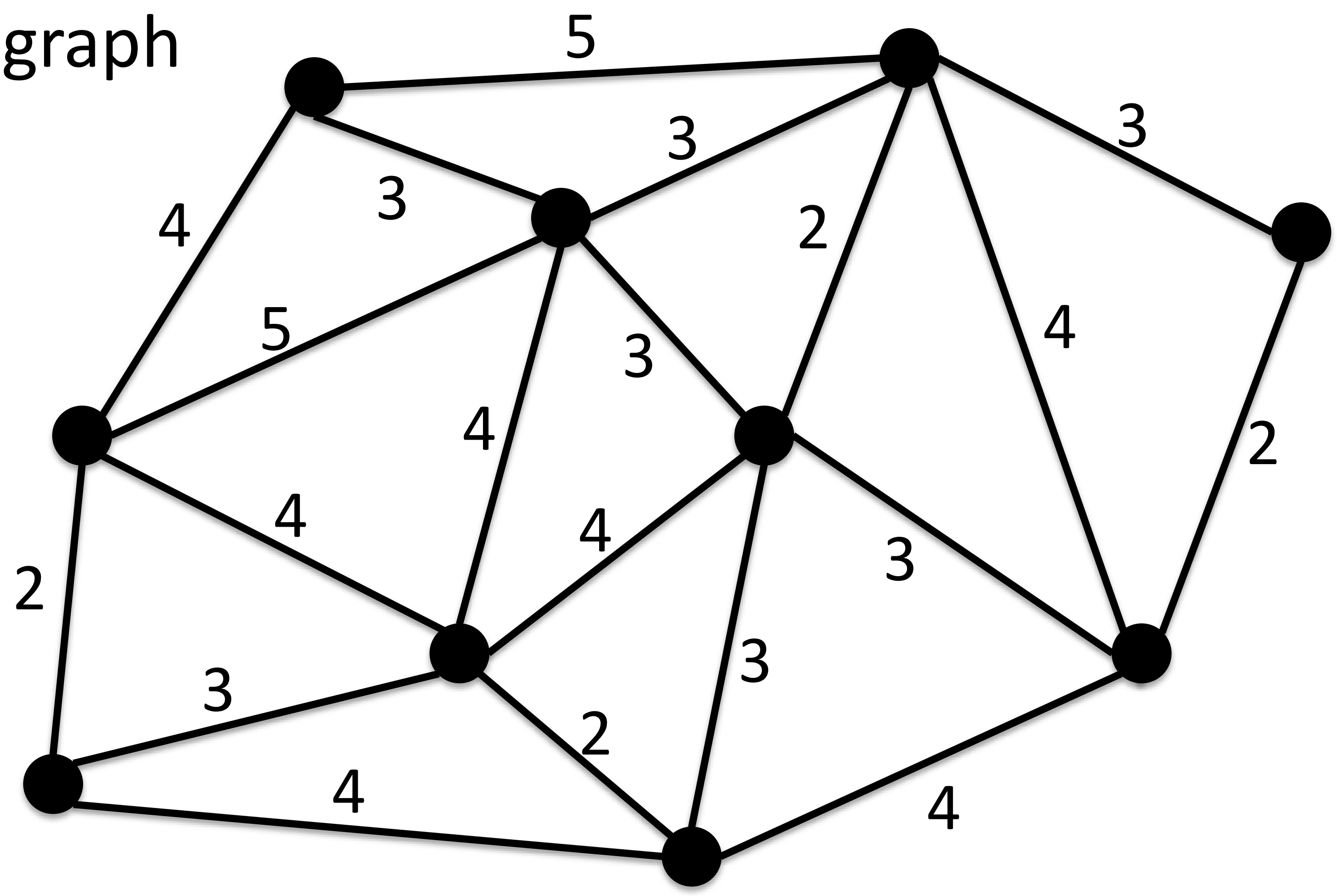
1. Enough streets must be paved so that it is possible for everyone to travel from their house to anyone else's house only along paved roads, and
2. The paving should cost as little as possible.

Here is the layout of the city. The number of paving stones between each house represents the cost of paving that route. Find the best route that connects all the houses, but uses as few counters (paving stones) as possible.


FOR SALE

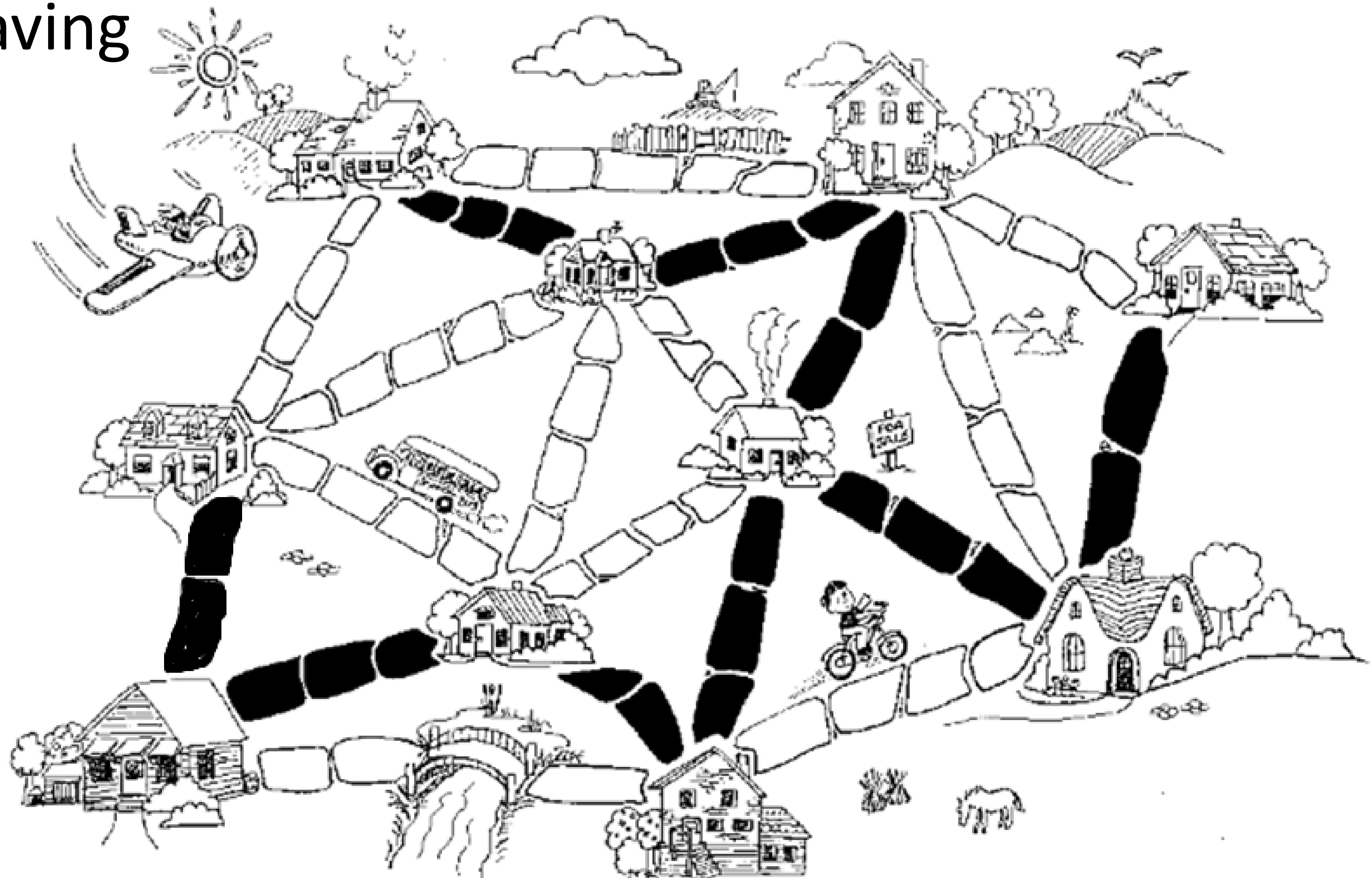
Muddy City Problem

The graph



Muddy City Problem

The paving



Application of MST: an example

- In the design of electronic circuitry, it is often necessary to make a set of pins electrically equivalent by wiring them together.
- Running cable TV to a set of houses. What's the least amount of cable needed to still connect all the houses?

Finding MST

- Kruskal's algorithm: start with no nodes or edges in the spanning tree and repeatedly add the cheapest edge that does not create a cycle

Kruskal algorithm

Procedure Kruskal (*G*: weighted connected undirected graph with *n* vertices)

T:= empty graph

for i := 1 to *n*-1

begin

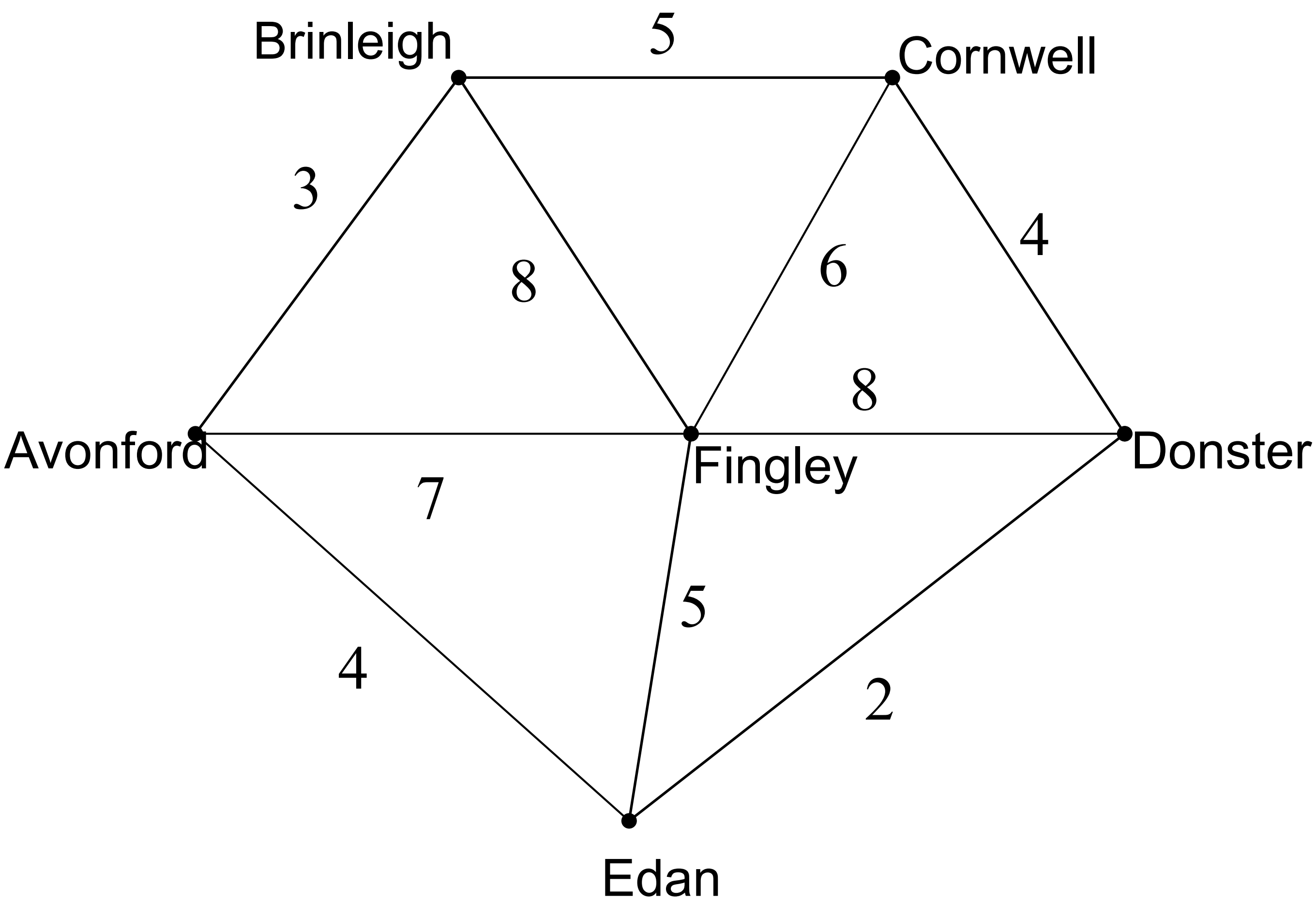
e:= any edge in *G* with smallest weight that does not form a simple circuit when added to *T*

T:= *T* with e added

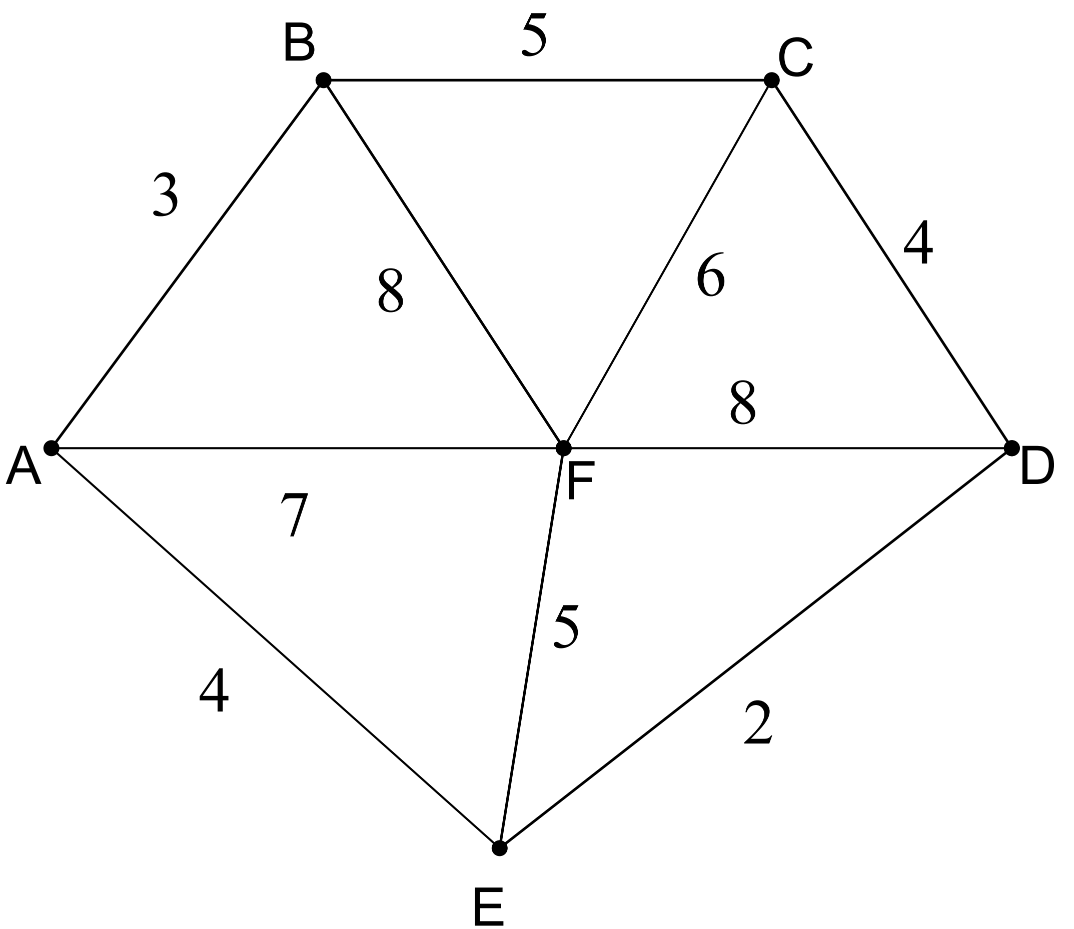
end (*T* is a minimum spanning tree of *G*)

Example

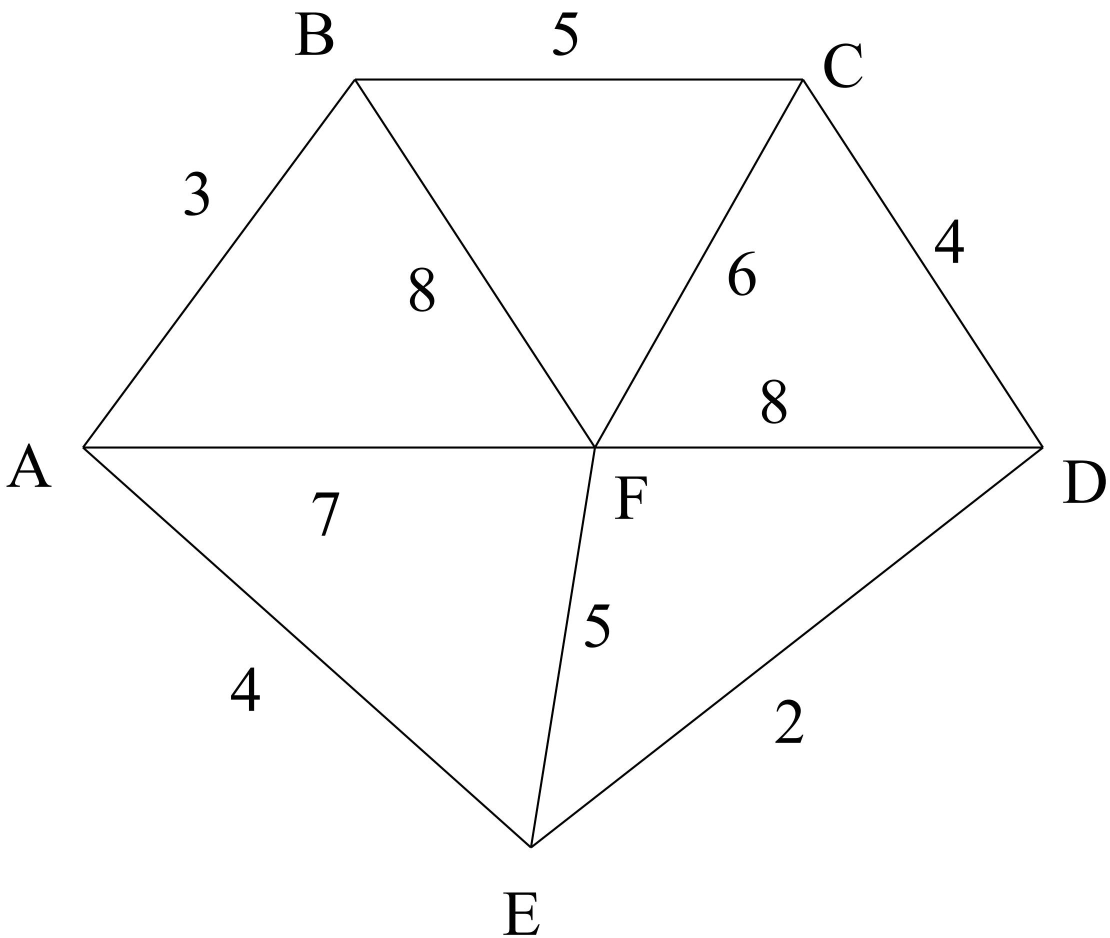
A cable company want to connect five villages to their network which currently extends to the market town of Avonford. What is the minimum length of cable needed?



We model the situation as a network, then the problem is to find the minimum connector for the network



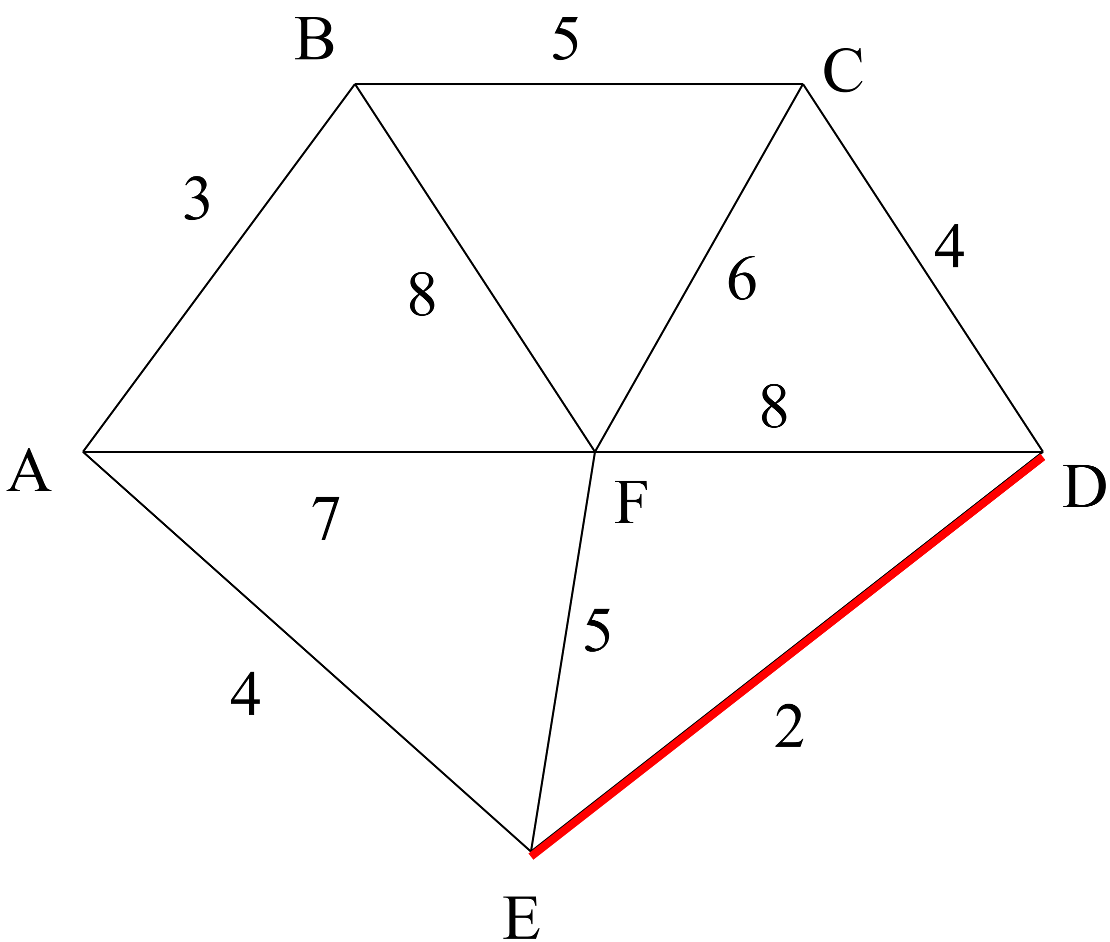
Kruskal's Algorithm



List the edges in order of size:

ED 2
AB 3
AE 4
CD 4
BC 5
EF 5
CF 6
AF 7
BF 8
CF 8

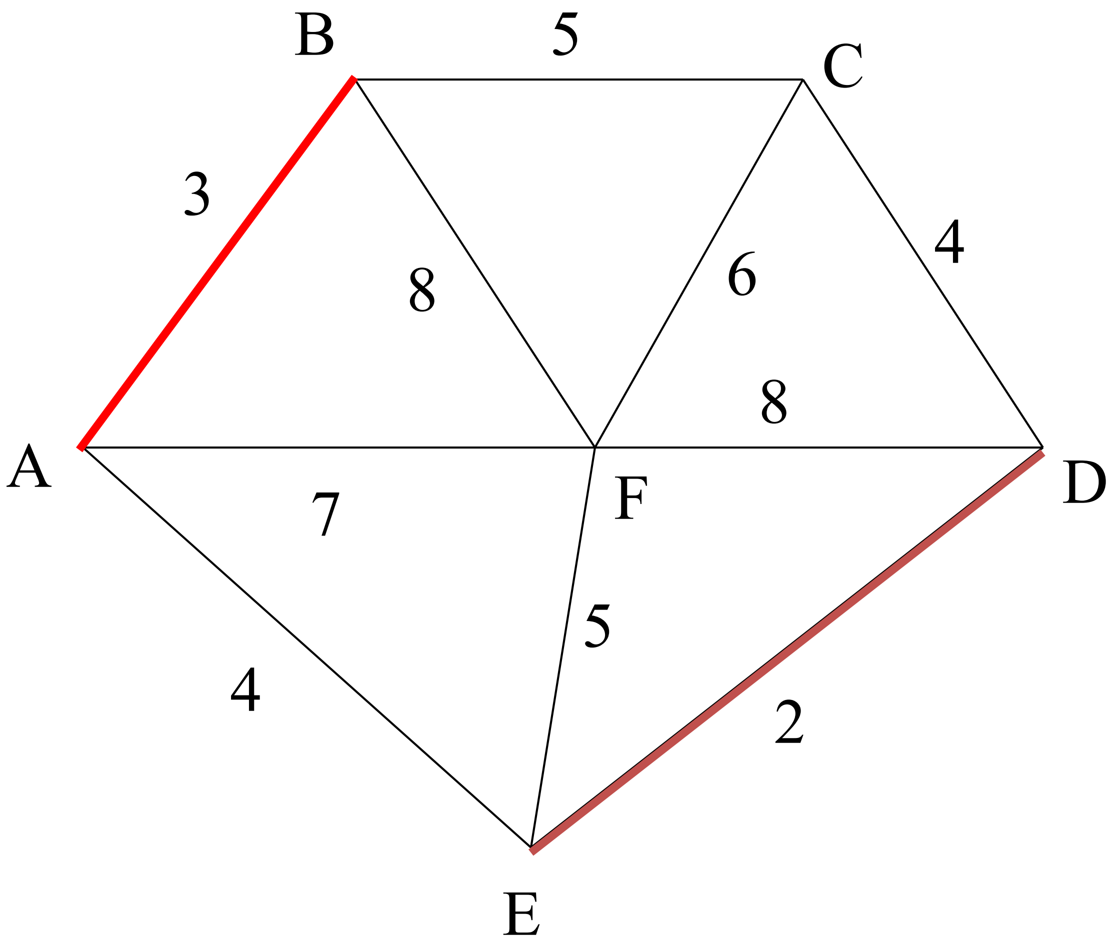
Kruskal's Algorithm



Select the shortest edge in the network

ED 2

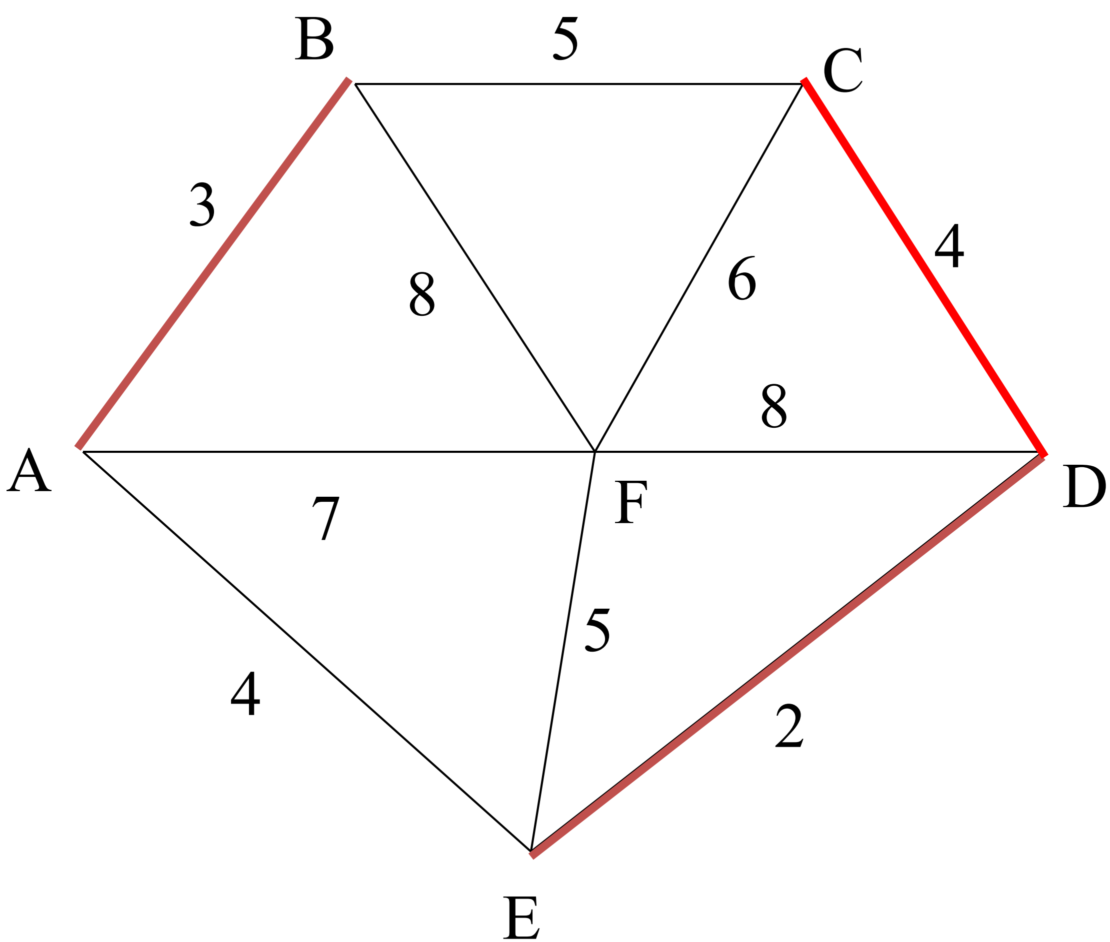
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3

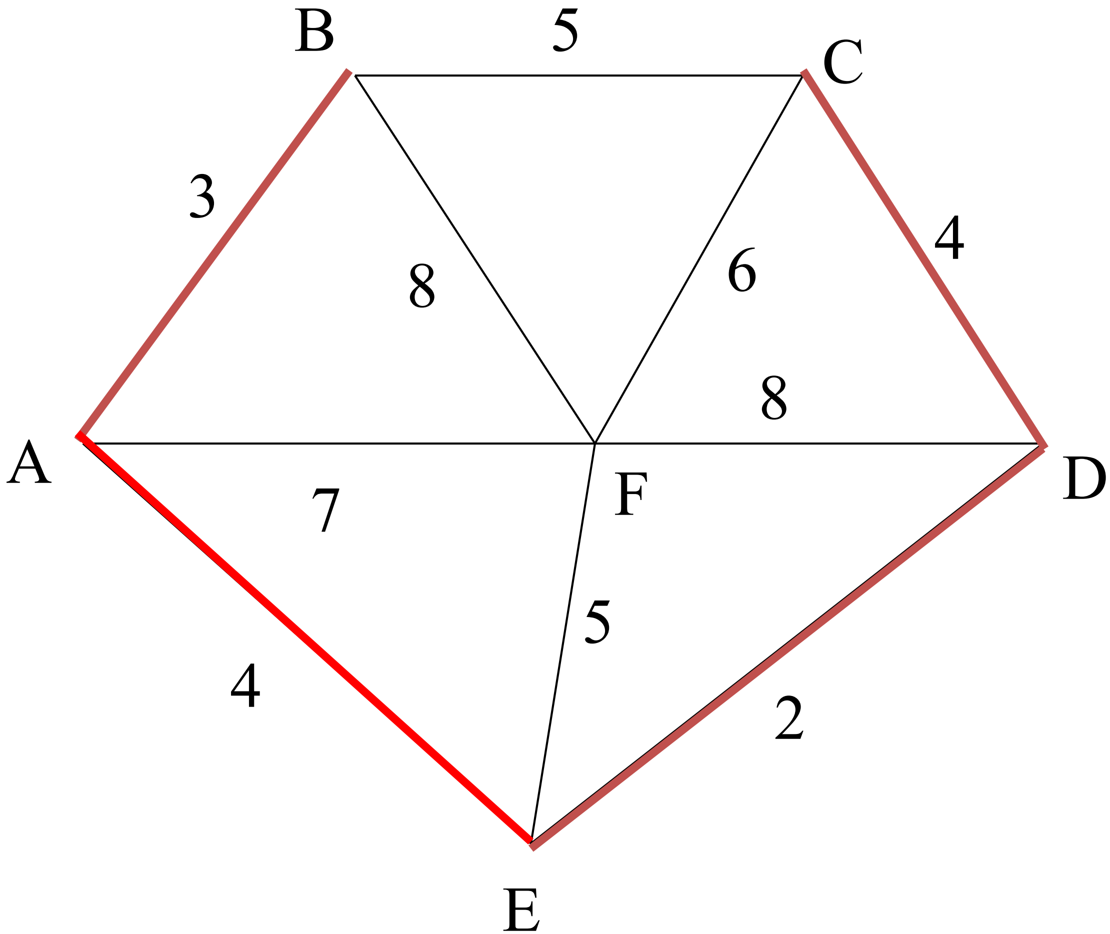
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4 (or AE 4)

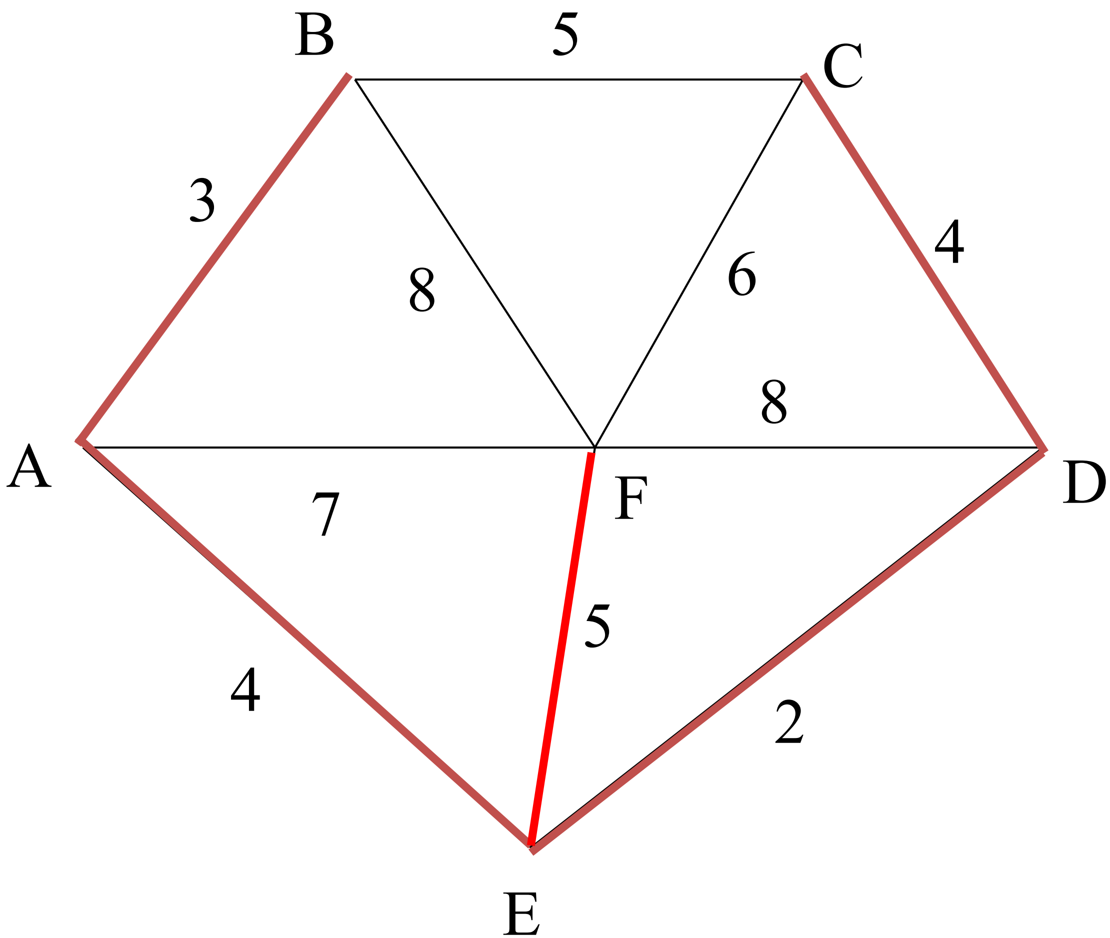
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4

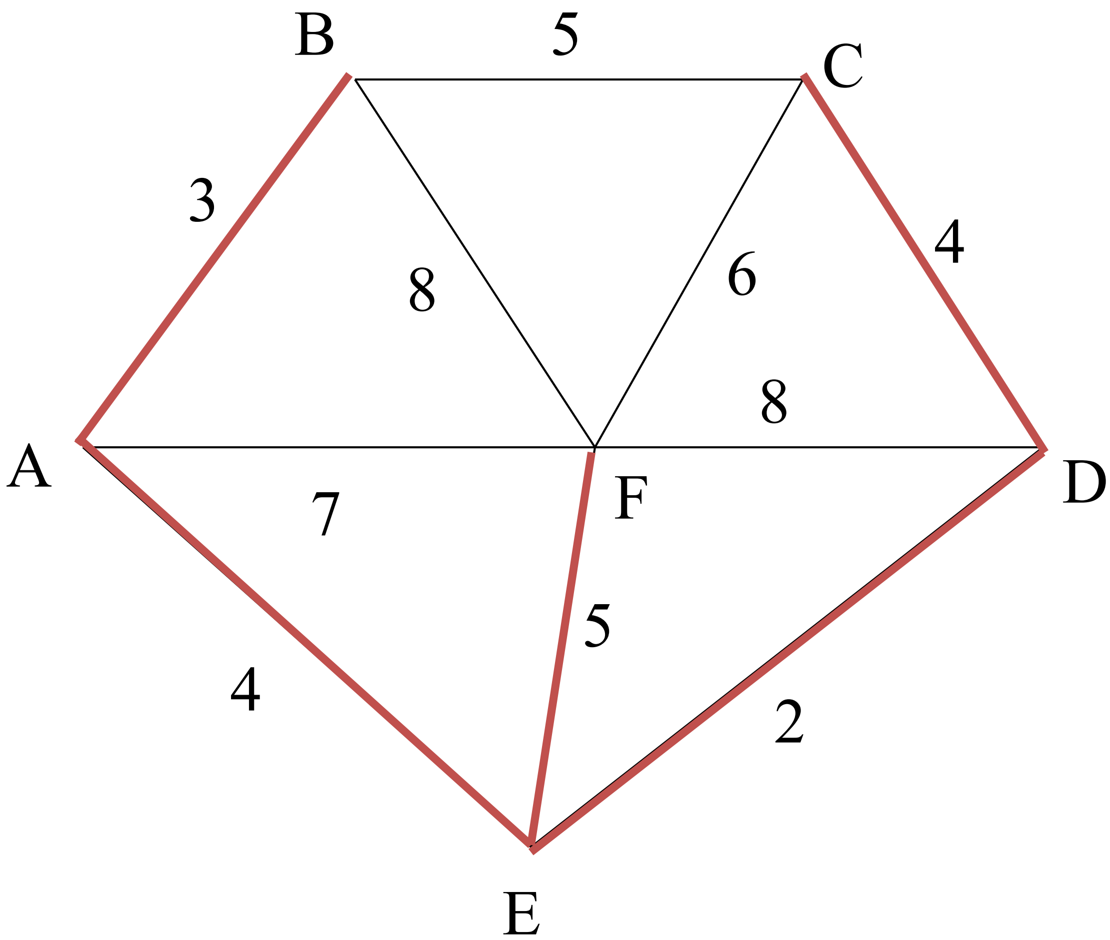
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4
BC 5 – forms a cycle
EF 5

Kruskal's Algorithm



All vertices have been connected.

The solution is

ED 2
AB 3
CD 4
AE 4
EF 5

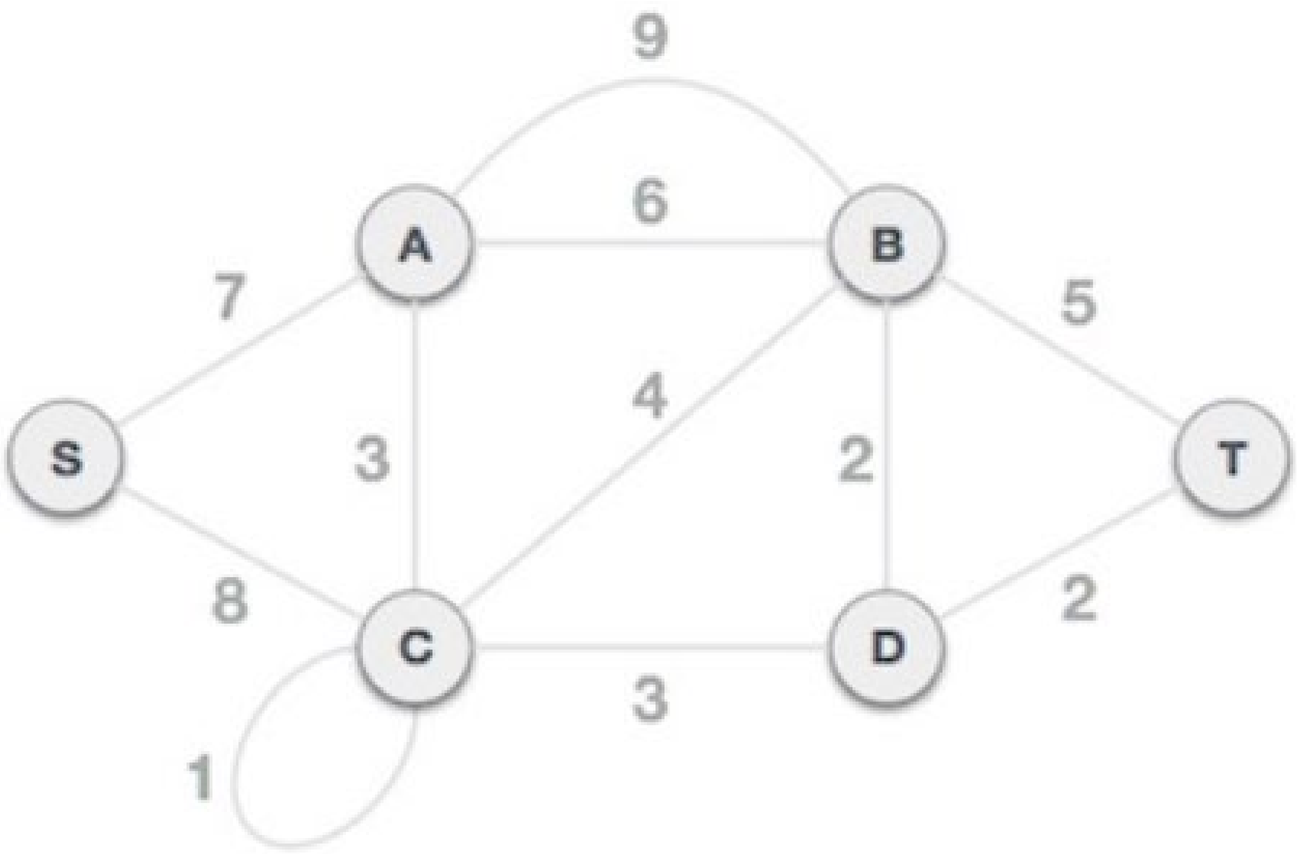
Total weight of tree: 18

Kruskal’s Algorithm

Important notes:

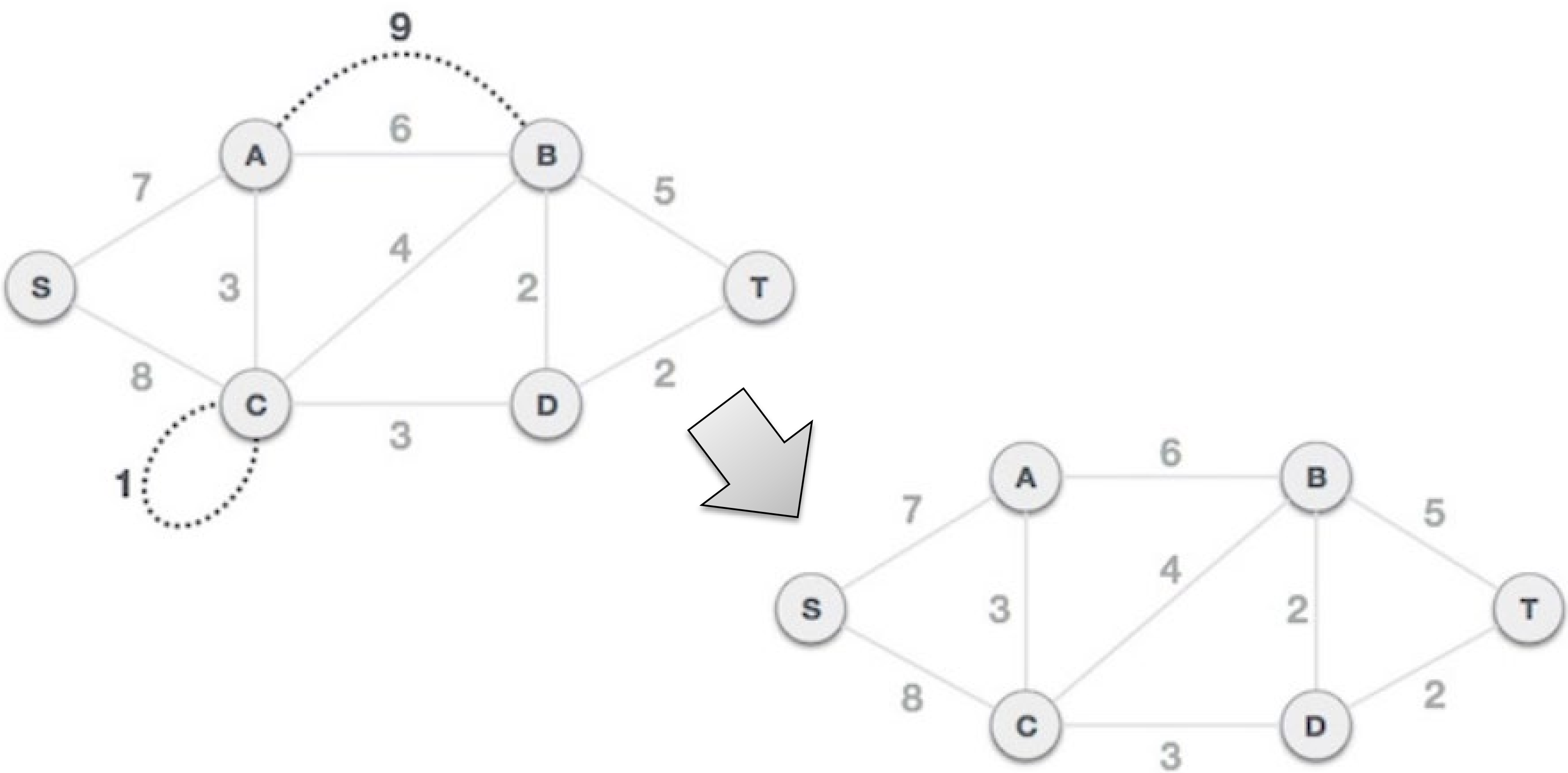
- The graph given should be a tree
  - Remove all loops (if any)
  - Remove all parallel edges
    - keep the one which has the least weight associated and remove all others

Example



Example

- Remove all loops and parallel edges



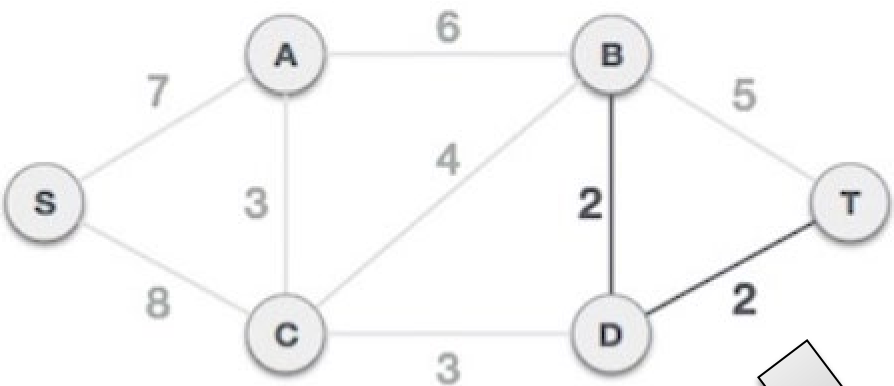
Example

- Arrange all edges in their increasing order of weight

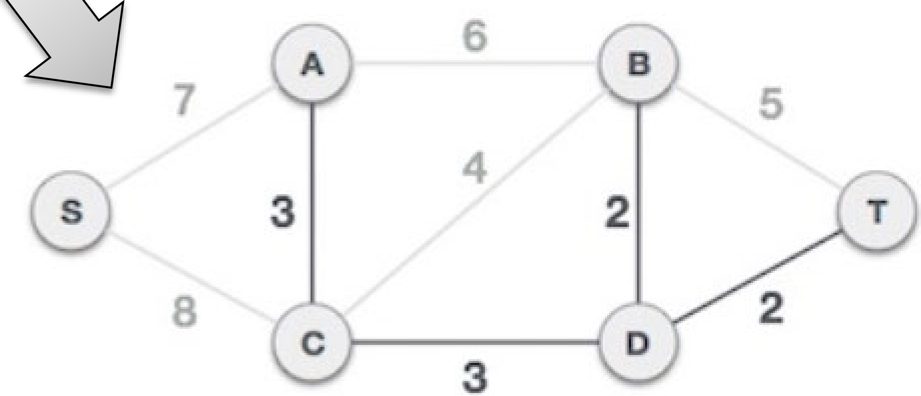
BD	DT	AC	CD	CB	BT	AB	SA	SC
2	2	3	3	4	5	6	7	8

Example

- Add the edge which has the least weightage

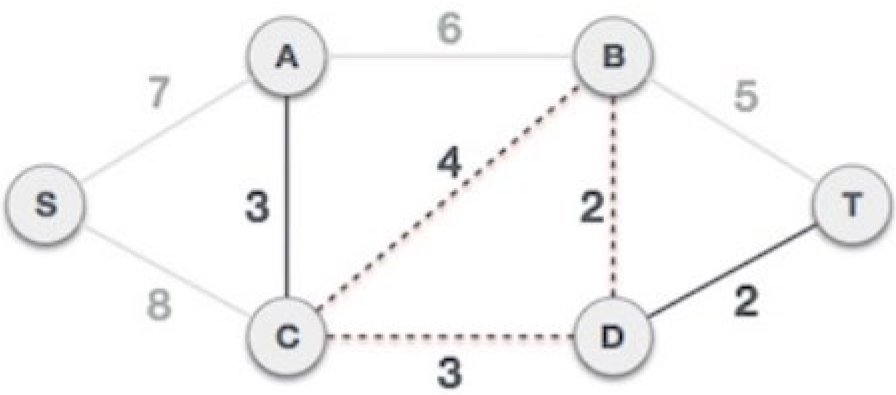


The least weight is 2 and edges involved are BD and DT



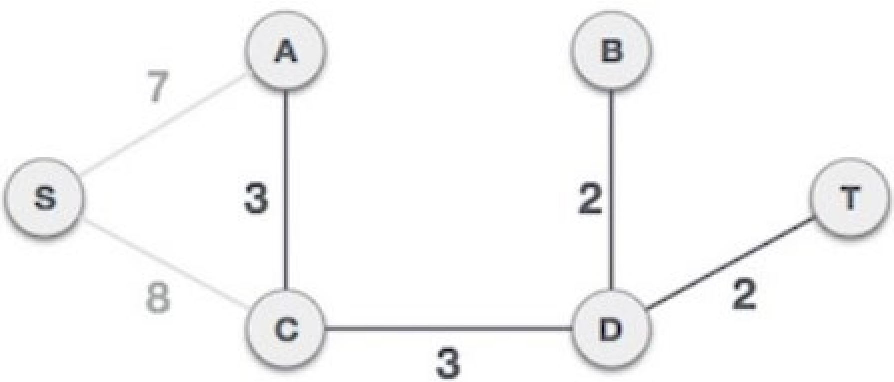
Next weight is 3, and associated edges are AC and CD

Example



Next weight is 4, and we observe that adding it will create a circuit in the graph. Thus, we ignore it.

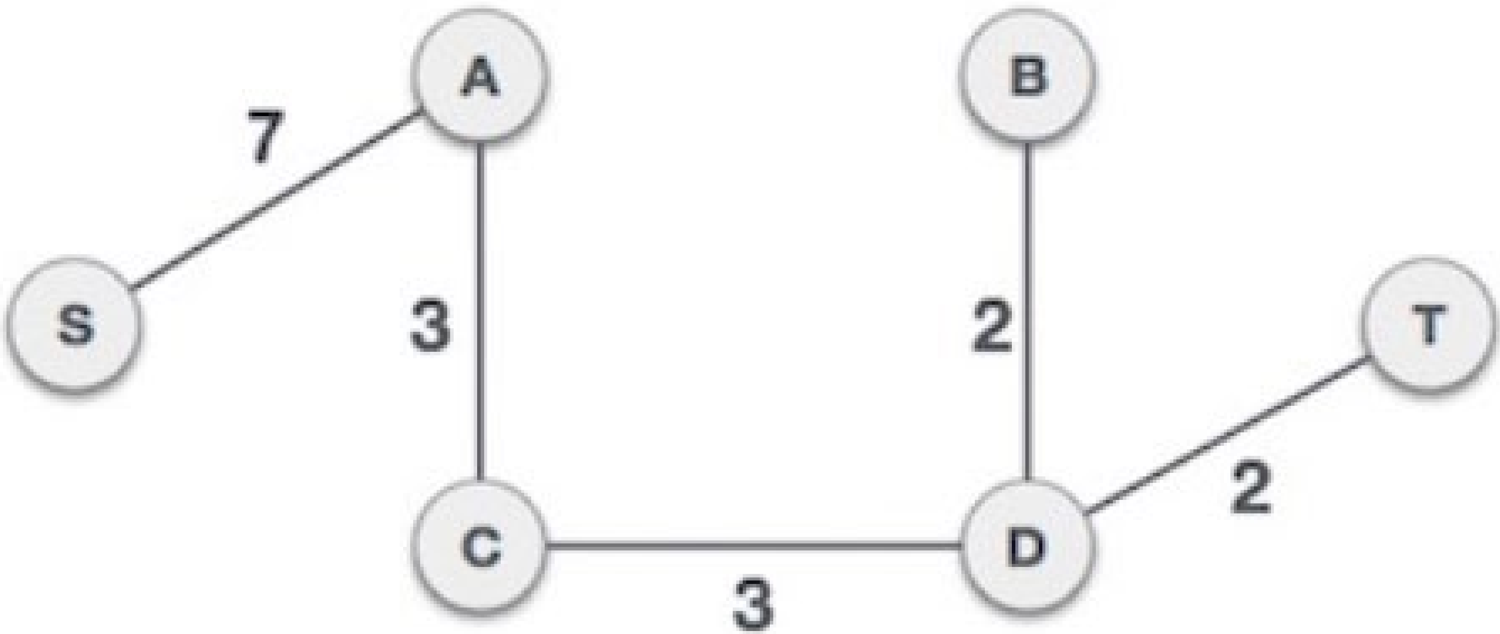
We observe that edges with weight 5 and 6 also create circuits. We ignore them and move on.



Now we are left with only one node to be added. Between the two least weighted edges available 7 and 8, we shall add the edge with weight 7.

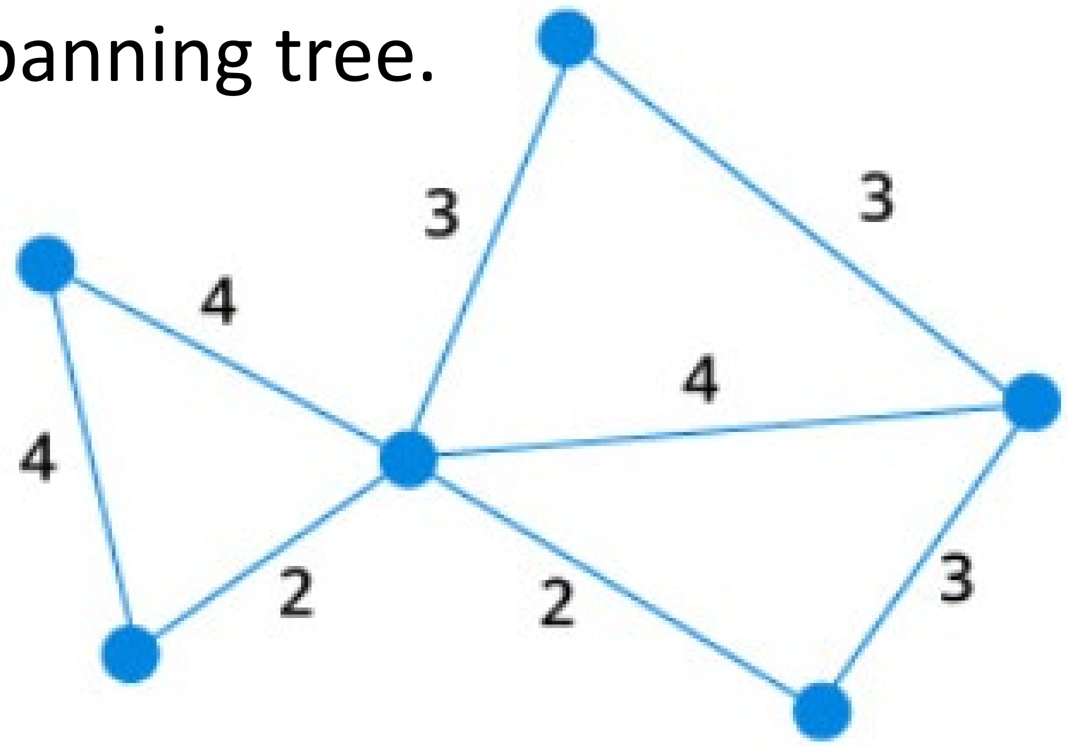
Example

Now we have minimum spanning tree with total weight is 17.



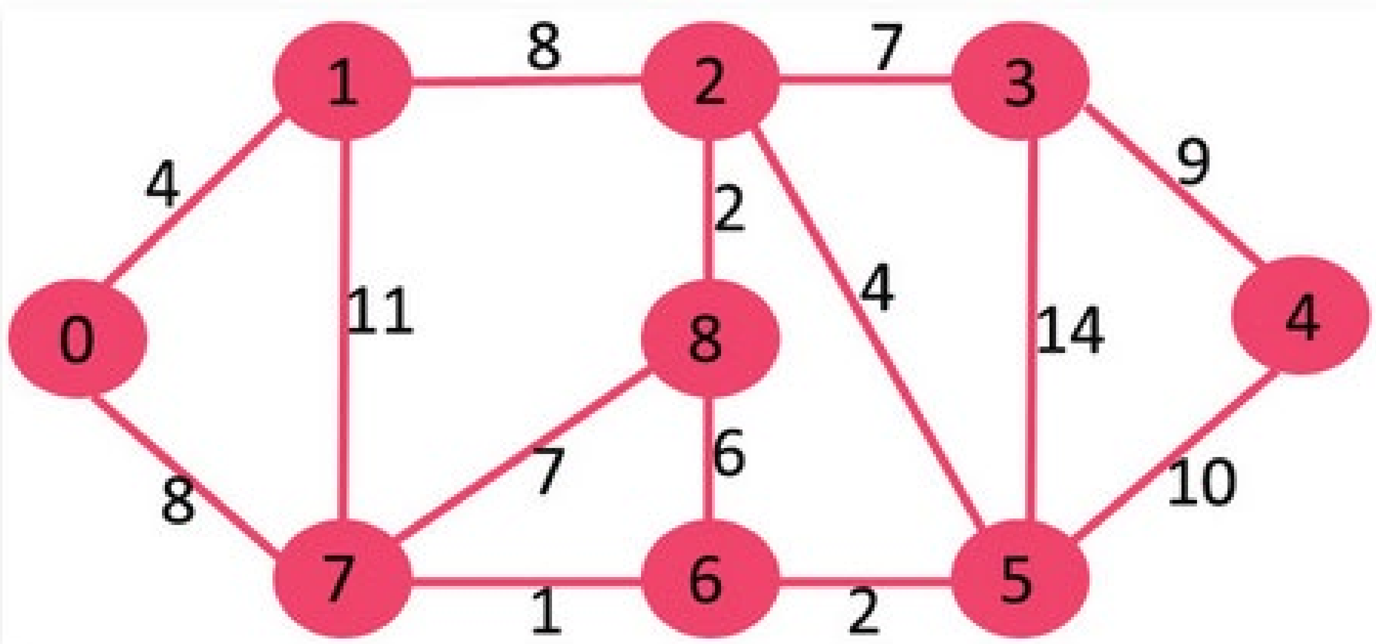
Exercise

Find the minimum spanning tree using Kraskal's Algorithm and give the total weight for the minimum spanning tree.



Exercise

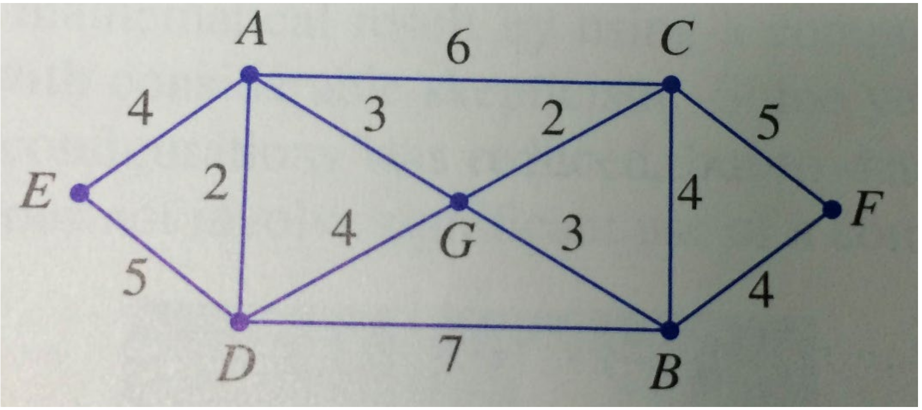
Find the minimum spanning tree using Kraskal's Algorithm and give the total weight for the minimum spanning tree.



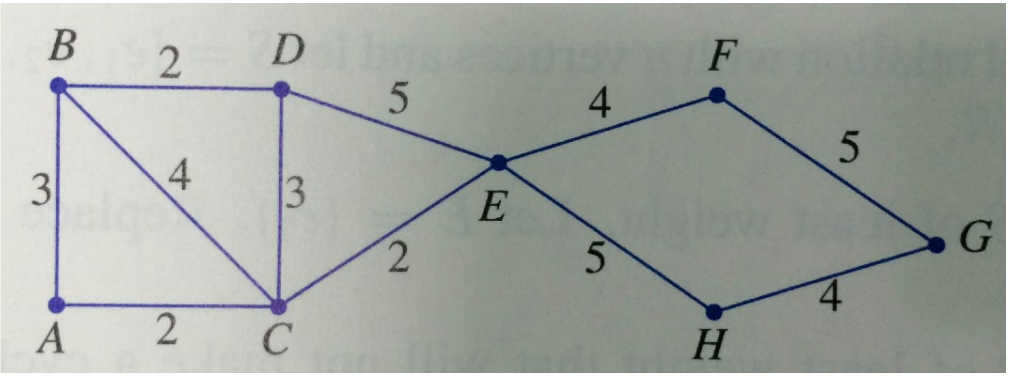
Exercise

Use Kruskal's algorithm to find a minimal spanning tree for the following graphs.

a)



b)

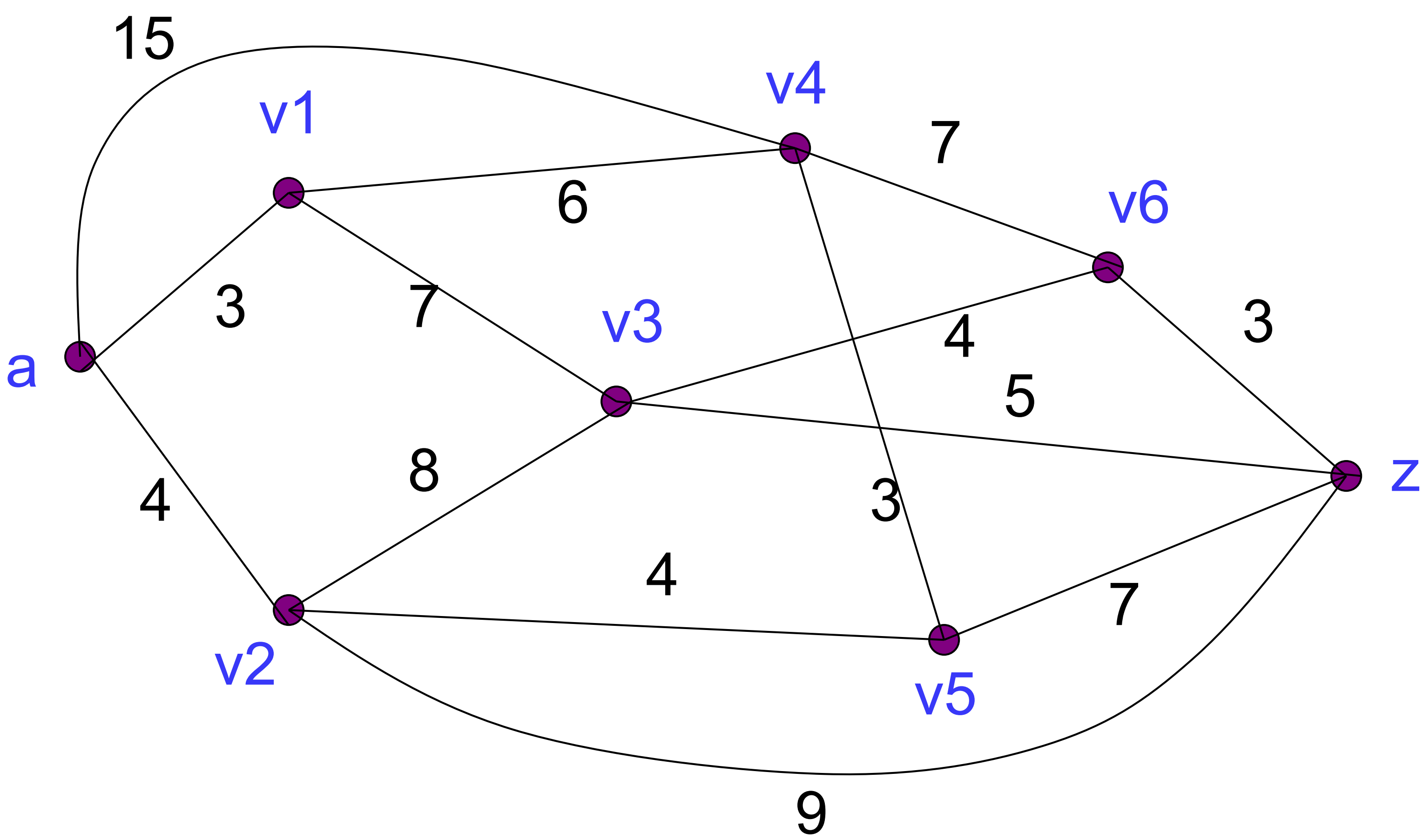


Shortest Path Problem

Shortest Path Problems

- Let *G* be a weighted graph.
- Let *u* and *v* be two vertices in *G*, and let *P* be a path in *G* from *u* to *v*.
- The length of path *P*, written *L*(*P*), is the sum of the weights of all the edges on path *P*.
- A shortest path from a vertex to another vertex is a path with the shortest length between the vertices.

example



Dijkstra's Shortest Path Algorithm

1. $S := \emptyset$
2. $N := V$
3. For all vertices, $u \in V$, $u \neq a$, $L(u) := \infty$
4. $L(a) := 0$

Dijkstra's Shortest Path Algorithm

5. While $z \notin S$ do,

   5.a Let $v \in N$ be such that

   $L(v) = \min\{L(u) \mid u \in N\}$

   5.b $S := S \cup \{v\}$

   5.c $N := N - \{v\}$

Dijkstra's Shortest Path Algorithm

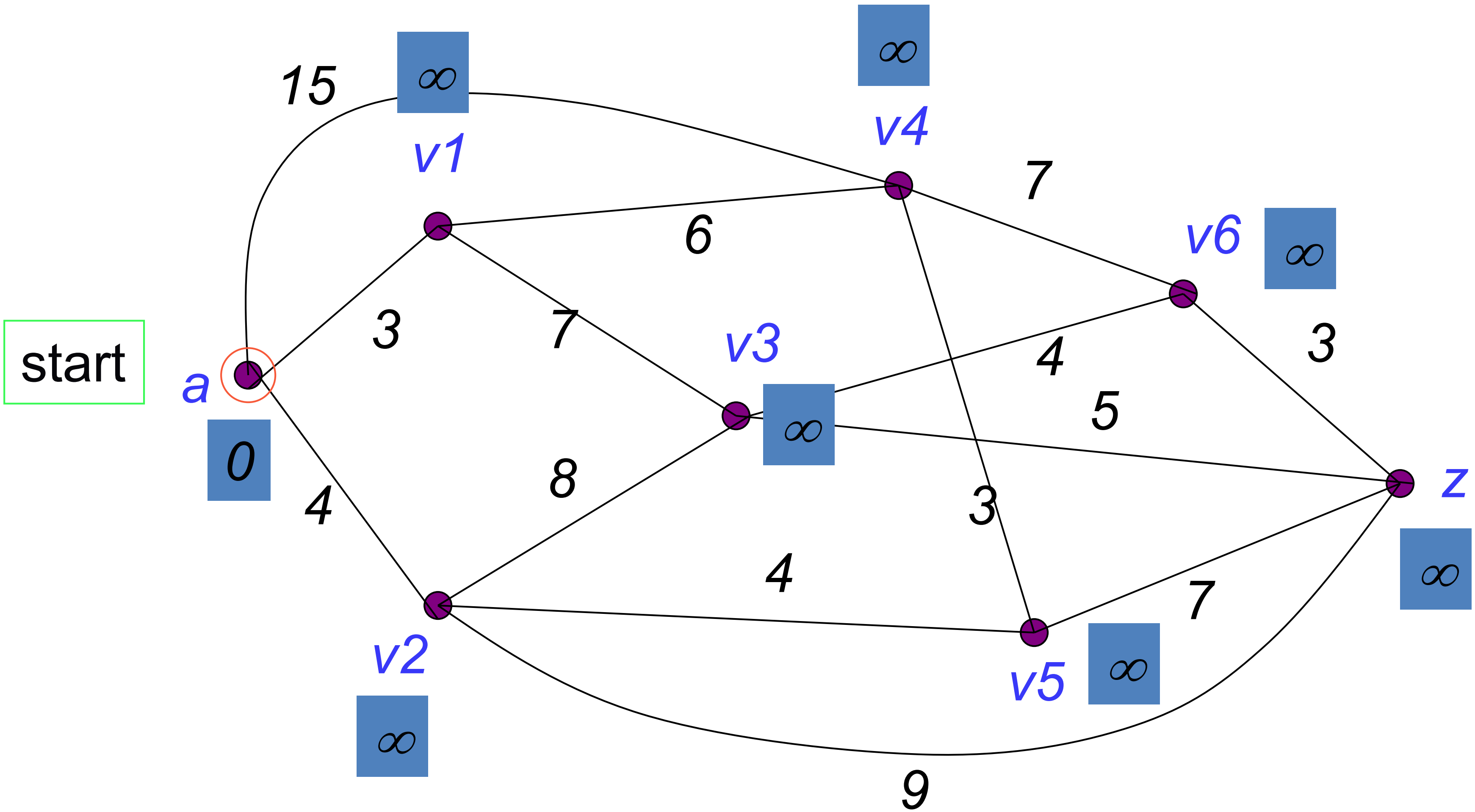
5.d For all $w \in N$ such that there is an edge from v to w

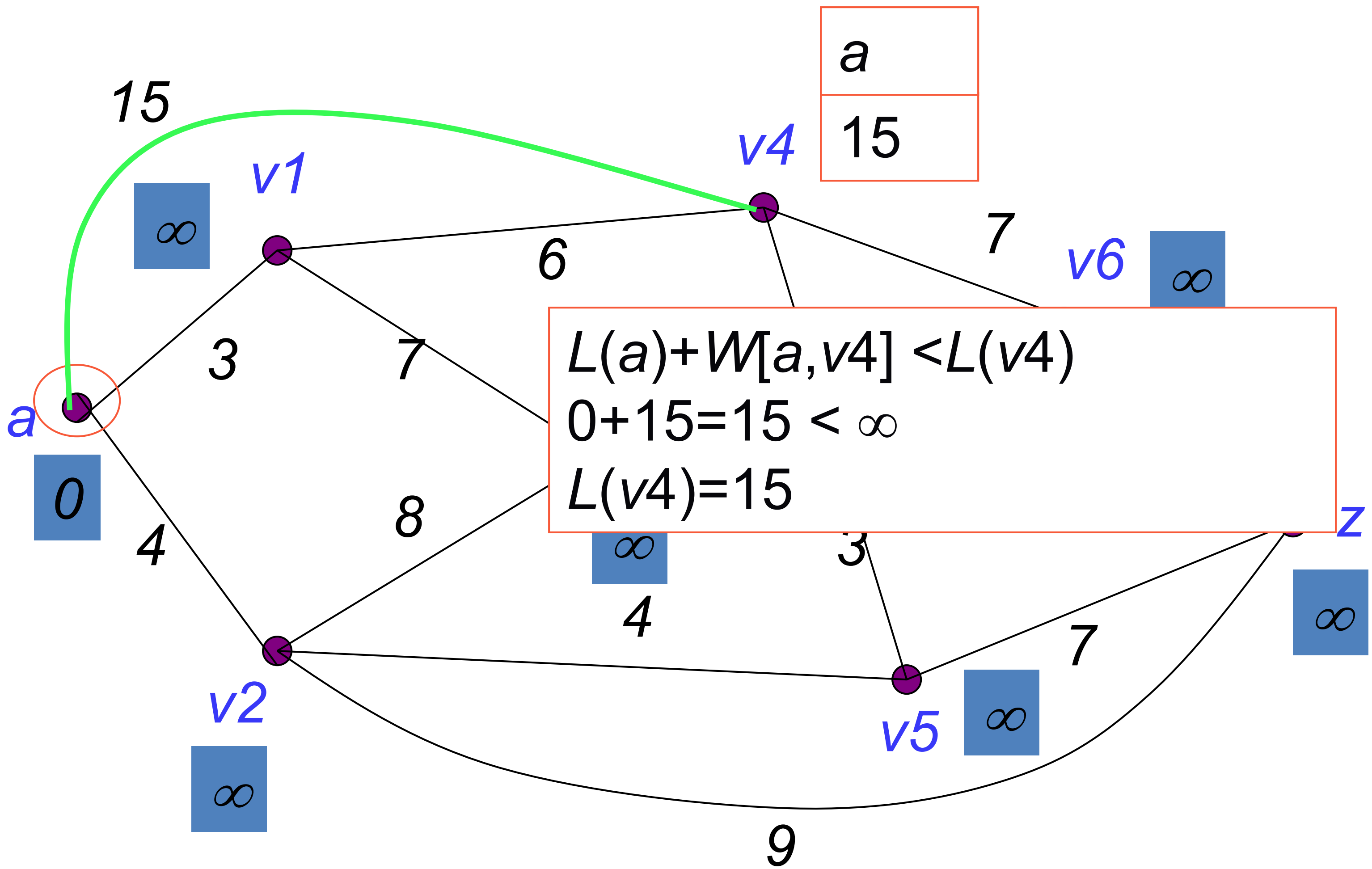
5.d.1 if $L(v)+W[v,w] < L(w)$ then

$L(w)=L(v)+W[v,w]$

$S=\varnothing$

$N=\{a,v1,v2,v3,v4,v5,v6,z\}$

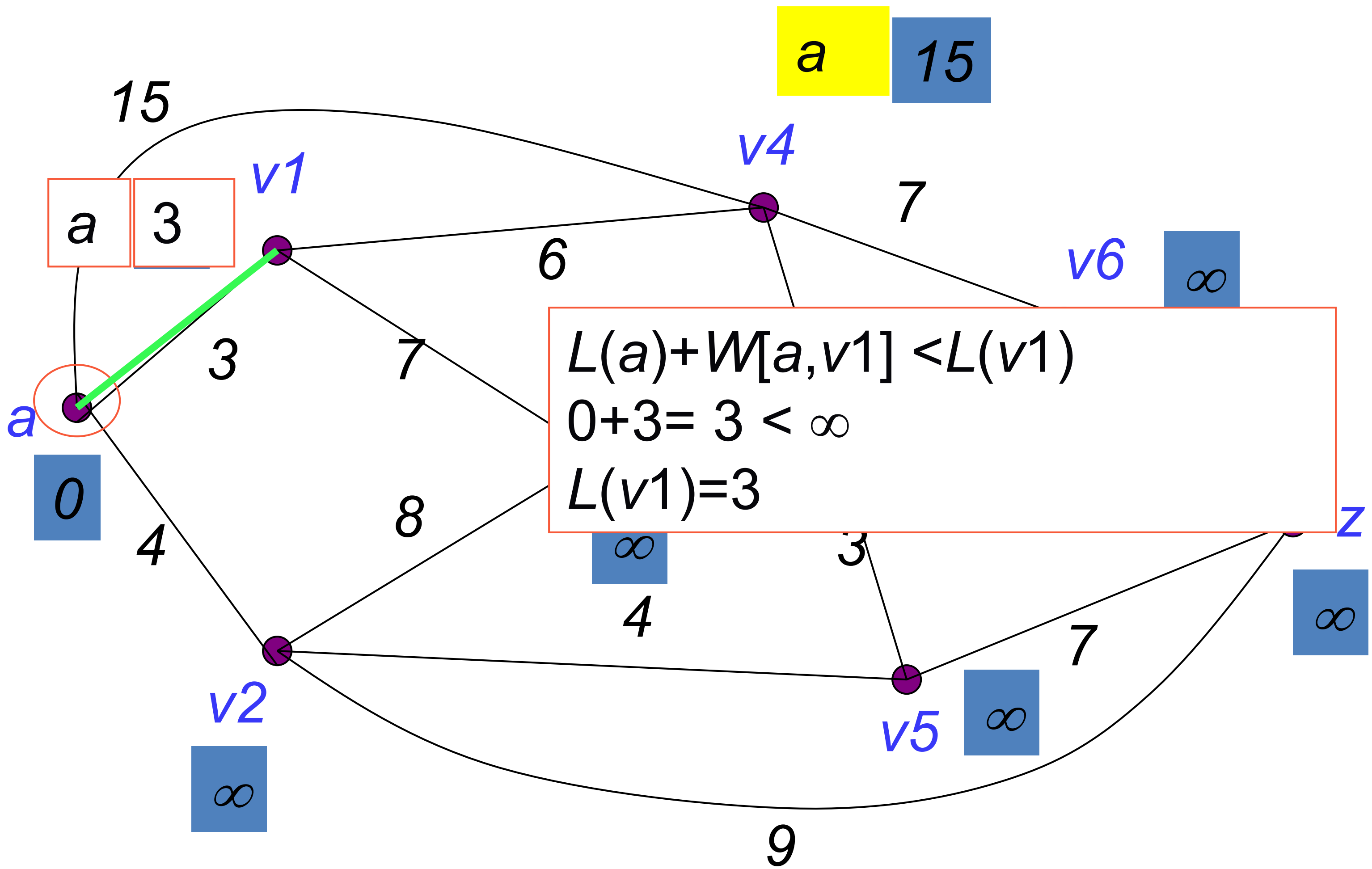




$L(a)+W[a,v4] < L(v4)$

$0+15=15 < \infty$

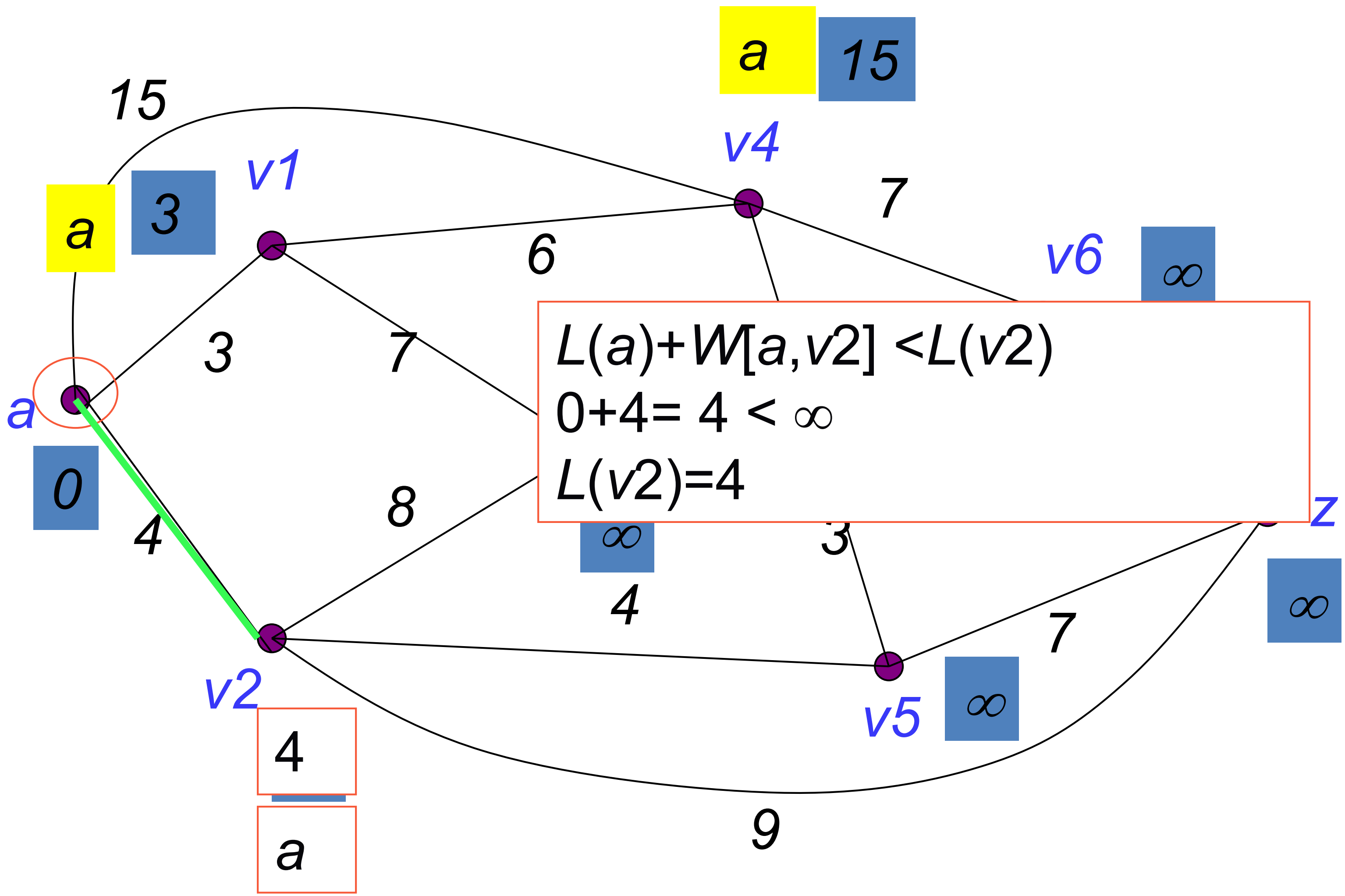
$L(v4)=15$



$L(a)+W[a,v1] < L(v1)$

$0+3= 3 < \infty$

$L(v1)=3$



$L(a)+W[a,v2] < L(v2)$

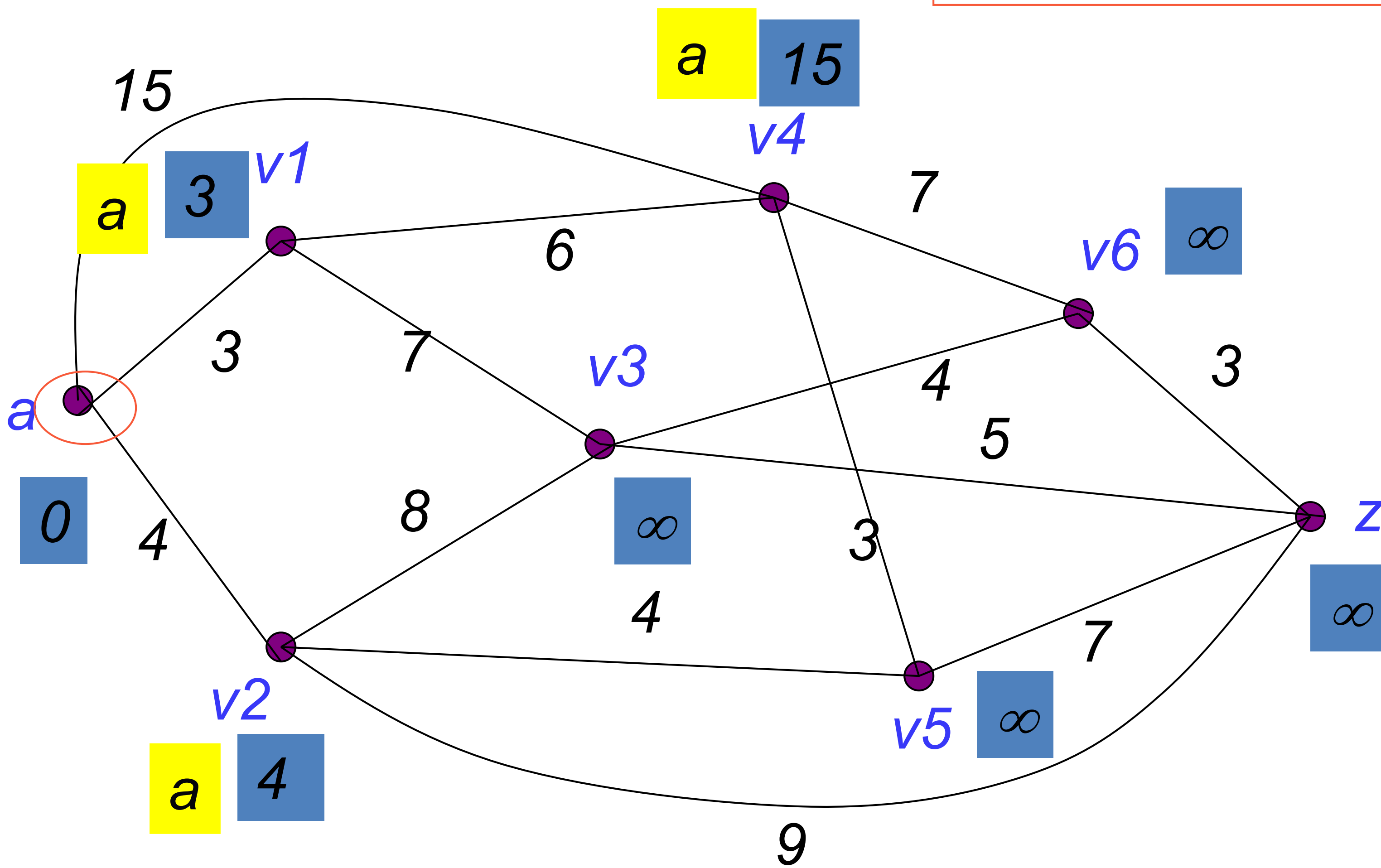
$0+4= 4 < \infty$

$L(v2)=4$

$S=\{a\}$

$N=\{v1,v2,v3,v4,v5$

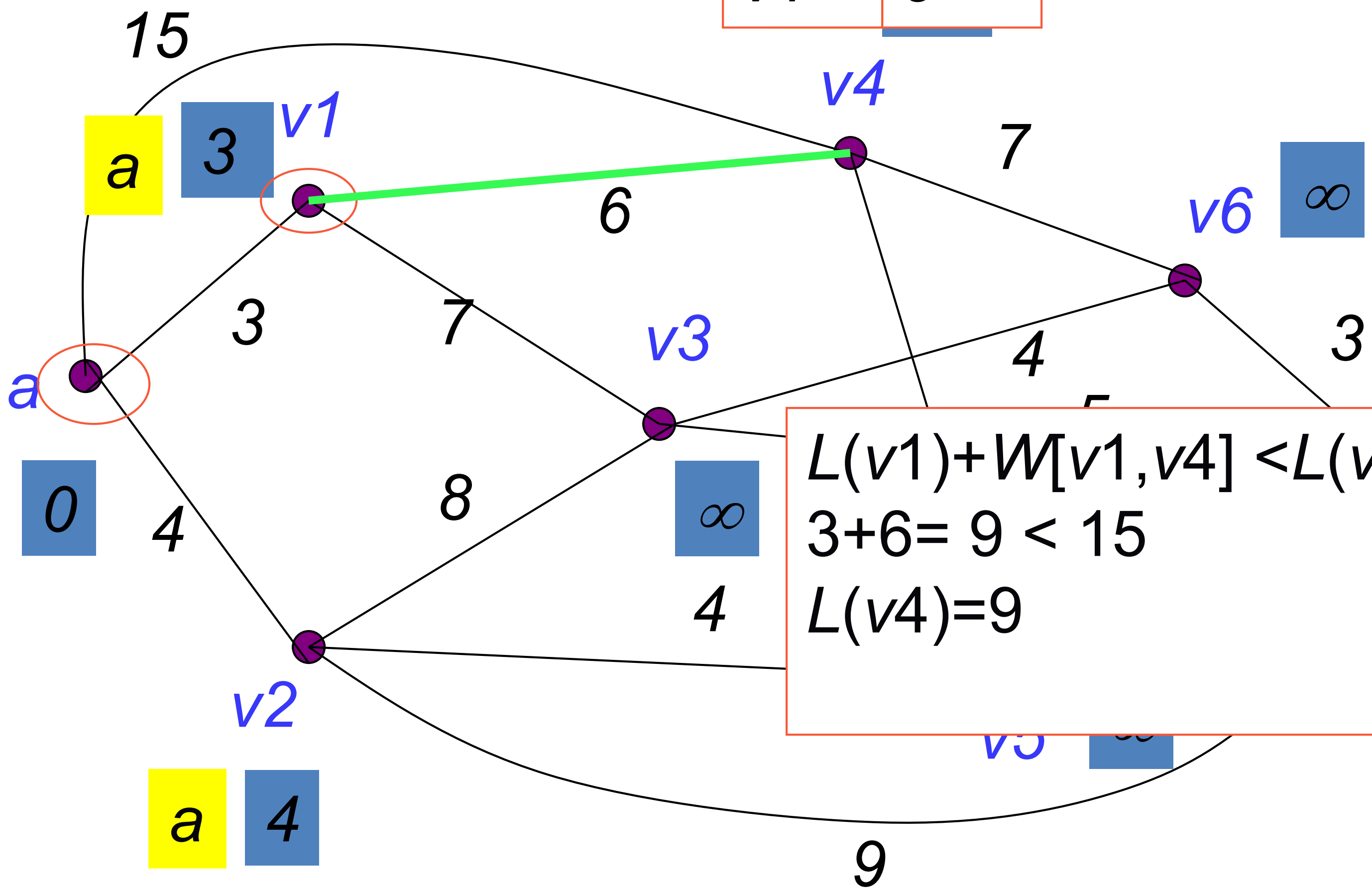
choose $v1$
because $L(v1)=3$
$= \min\{L(u) | u \in N\}$



$S=\{a,v1\}$

$N=\{v2,v3,v4,v5,v6,z\}$

v1	9



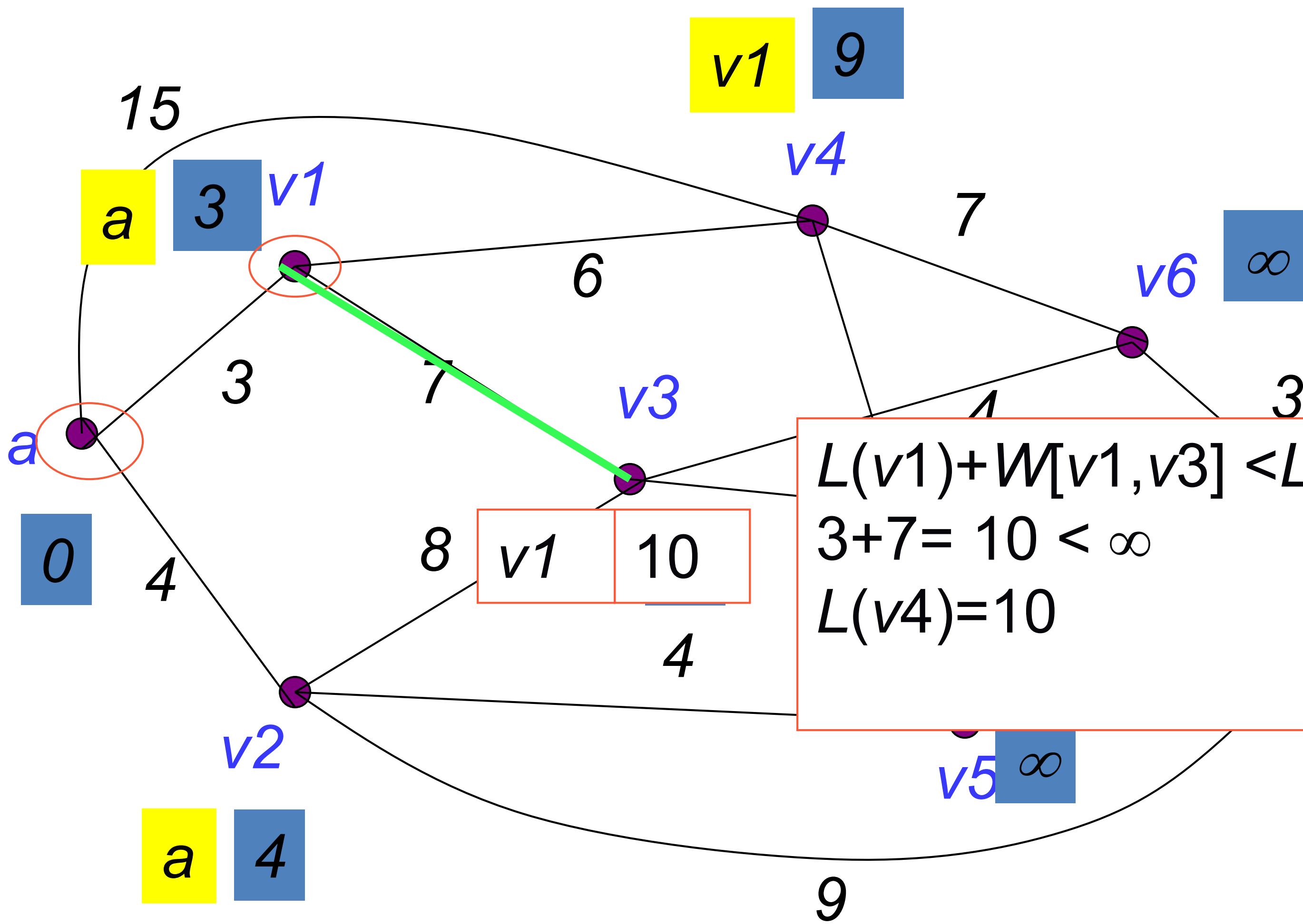
$L(v1)+W[v1,v4] < L(v4)$

$3+6= 9 < 15$

$L(v4)=9$

$S=\{a,v1\}$

$N=\{v2,v3,v4,v5,v6,z\}$



$L(v1)+W[v1,v3] < L(v3)$

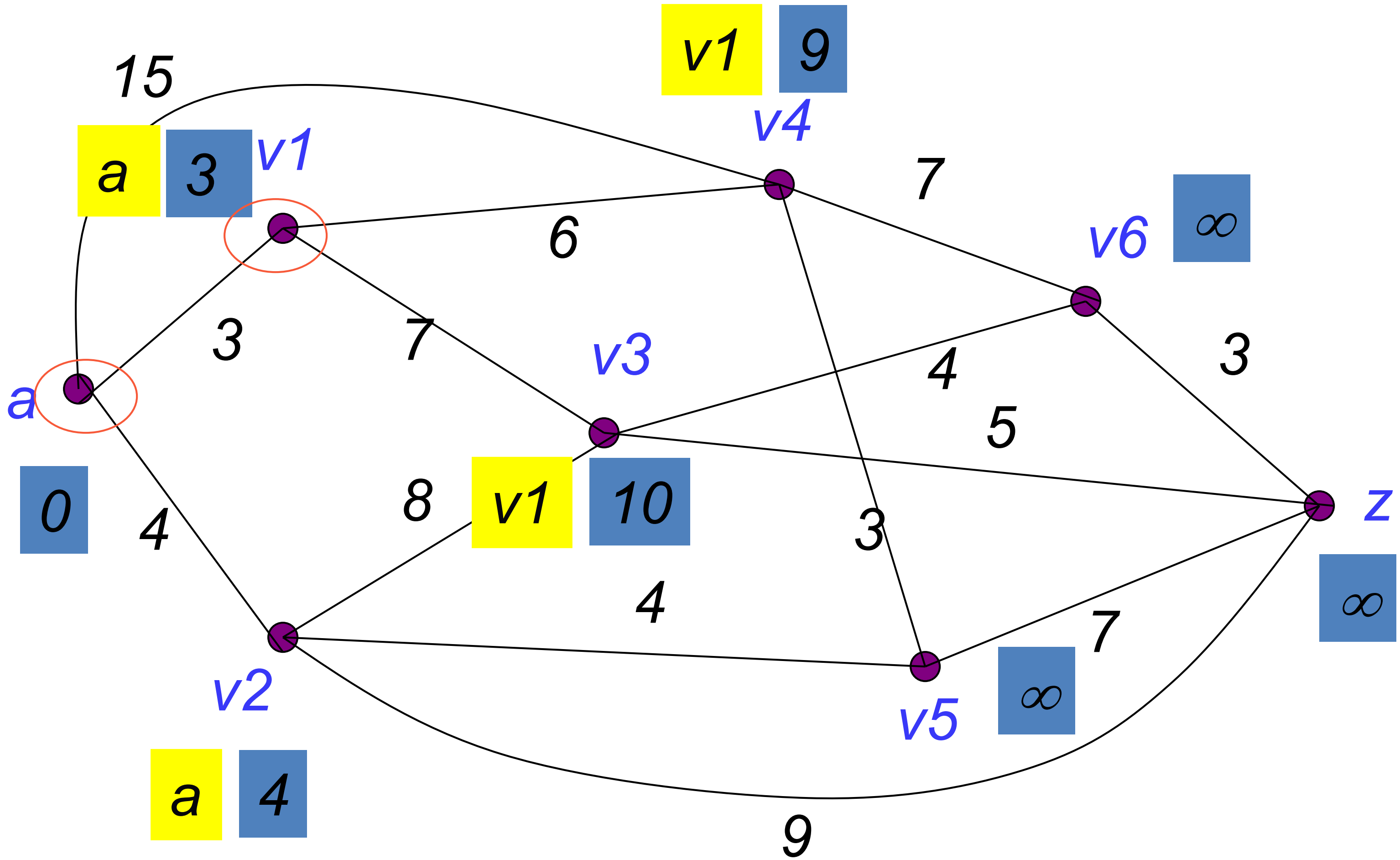
$3+7=10 < \infty$

$L(v4)=10$

$S=\{a, v1\}$

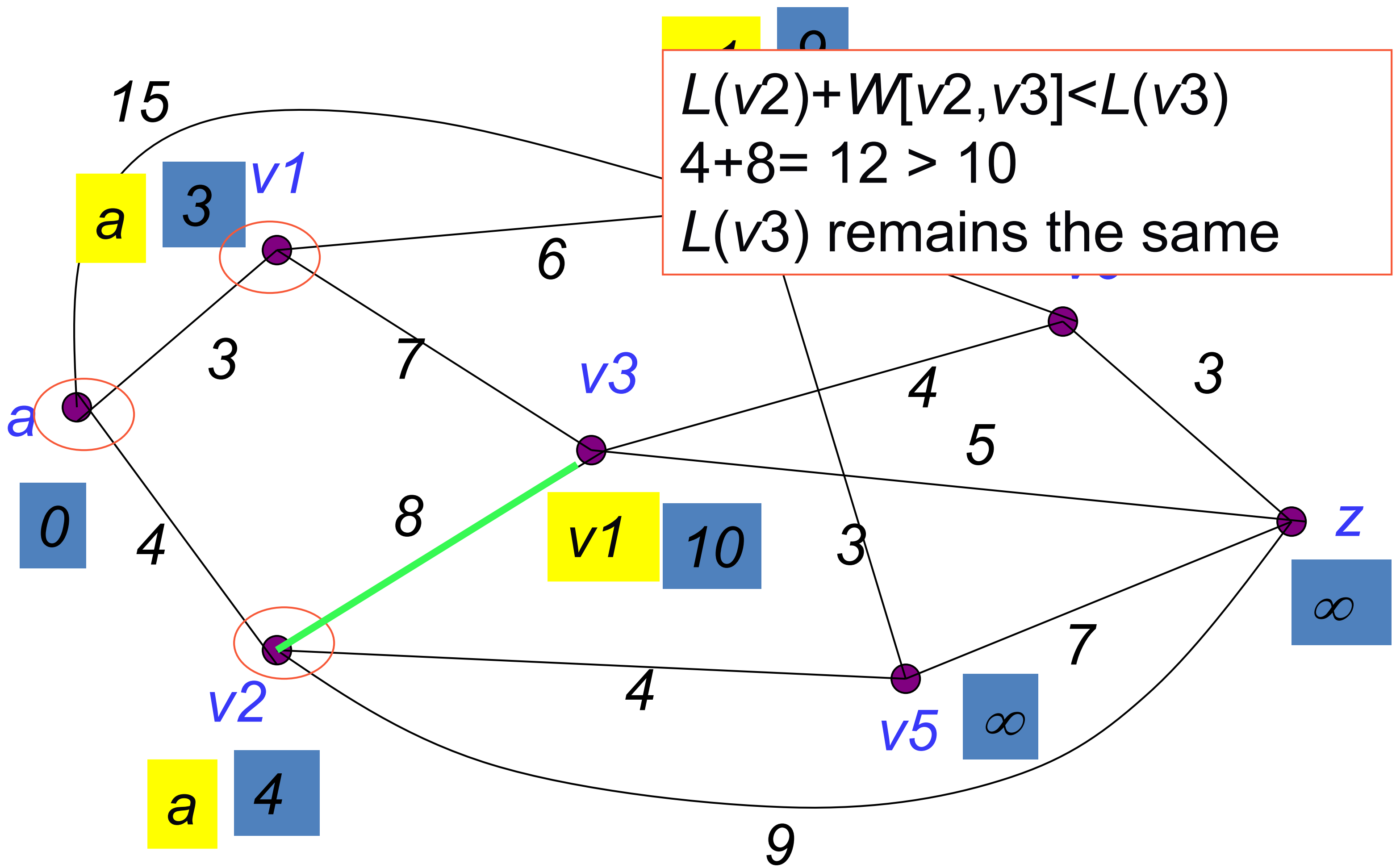
$N= \{v2,v3,v4,v5,v6,z\}$

choose *v2* because $L(v2)=4 = \min\{L(u)|u \in N\}$



$S=\{a, v1, v2\}$

$N= \{v3,v4,v5,v6,z\}$



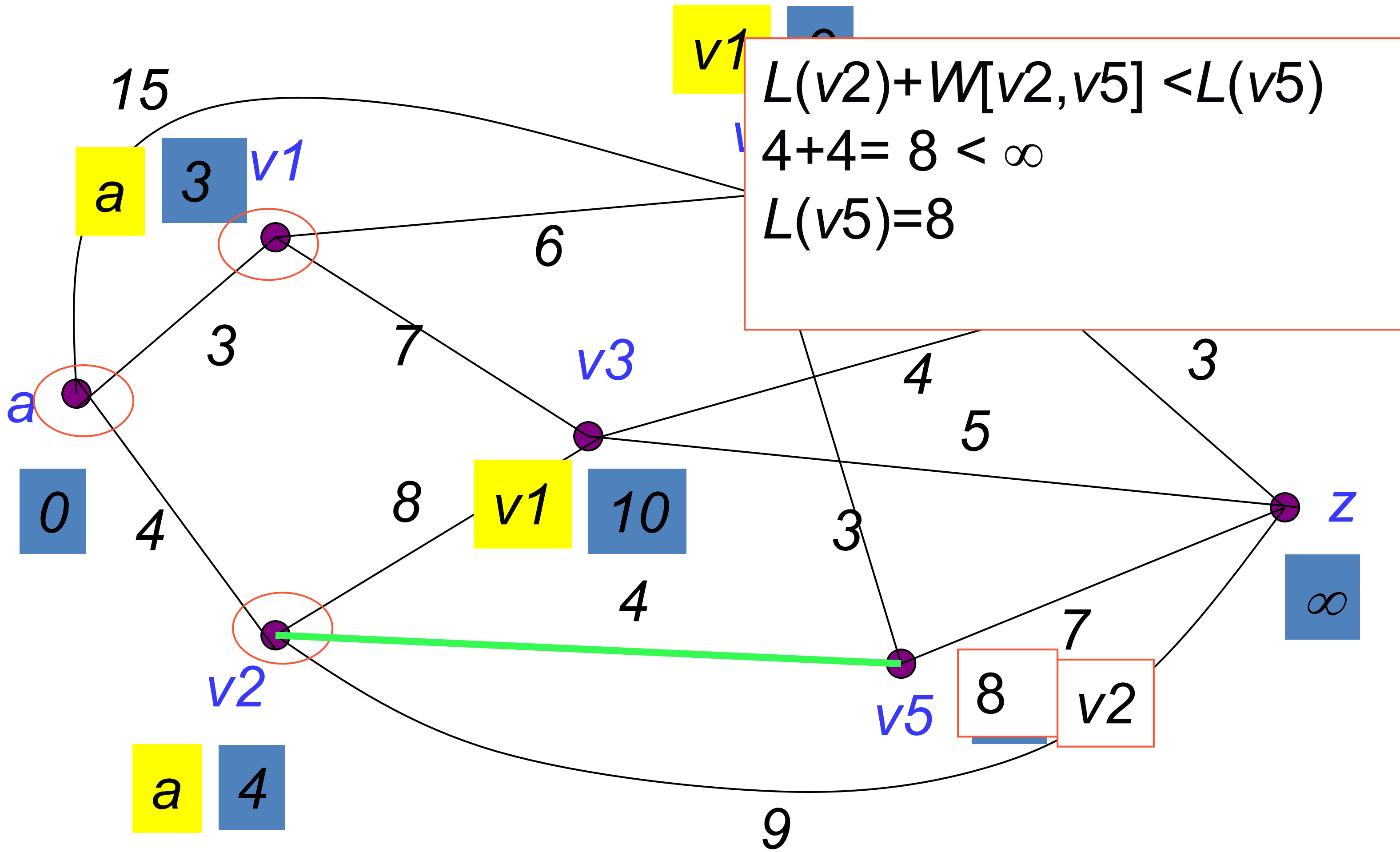
$L(v2)+W[v2,v3]<L(v3)$

$4+8= 12 > 10$

$L(v3)$ remains the same

$S=\{a, v1, v2\}$

$N= \{v3,v4,v5,v6,z\}$



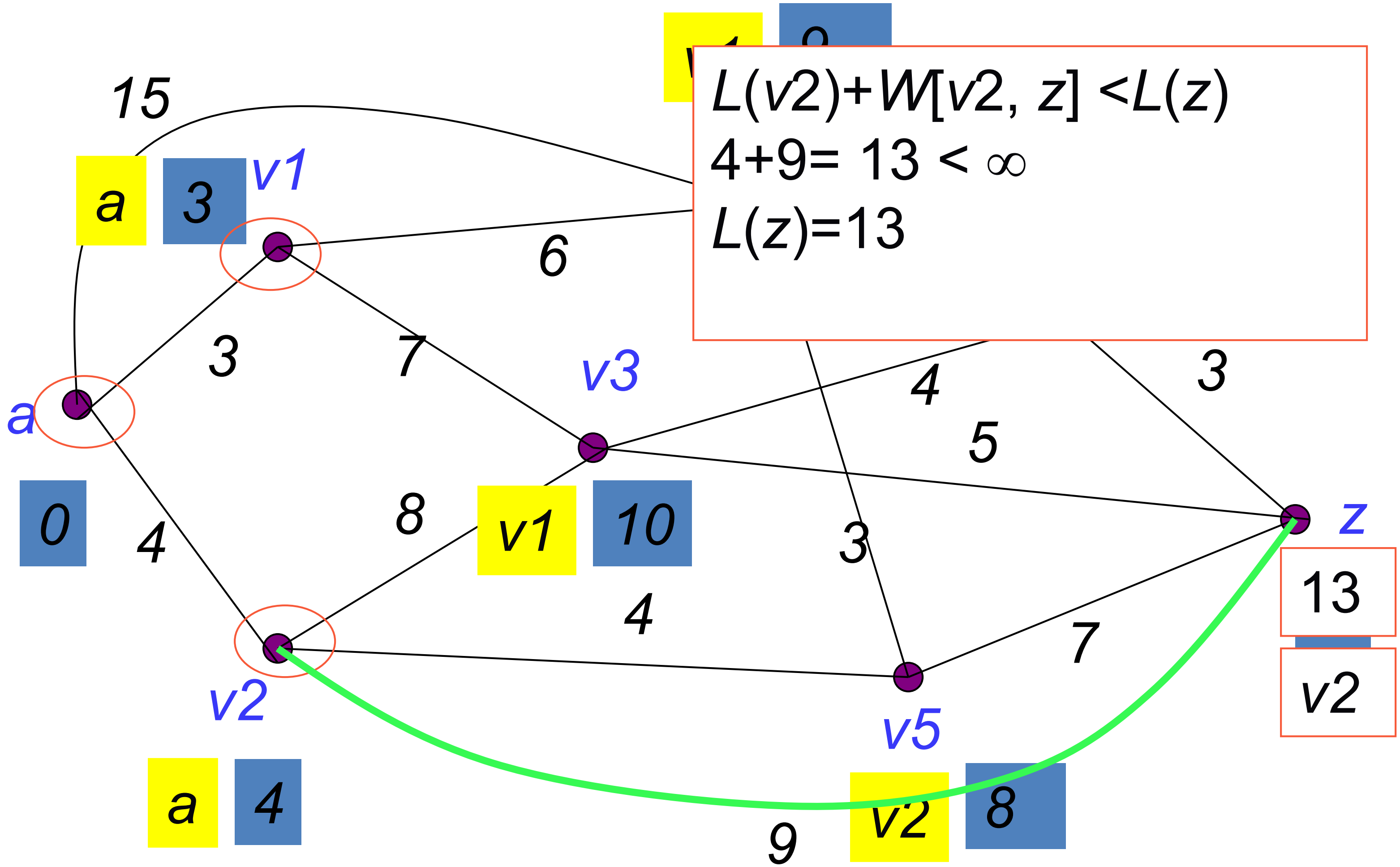
$L(v2)+W[v2,v5] < L(v5)$

$4+4= 8 < \infty$

$L(v5)=8$

$S=\{a, v1, v2\}$

$N= \{v3,v4,v5,v6,z\}$



$L(v2)+W[v2, z] < L(z)$

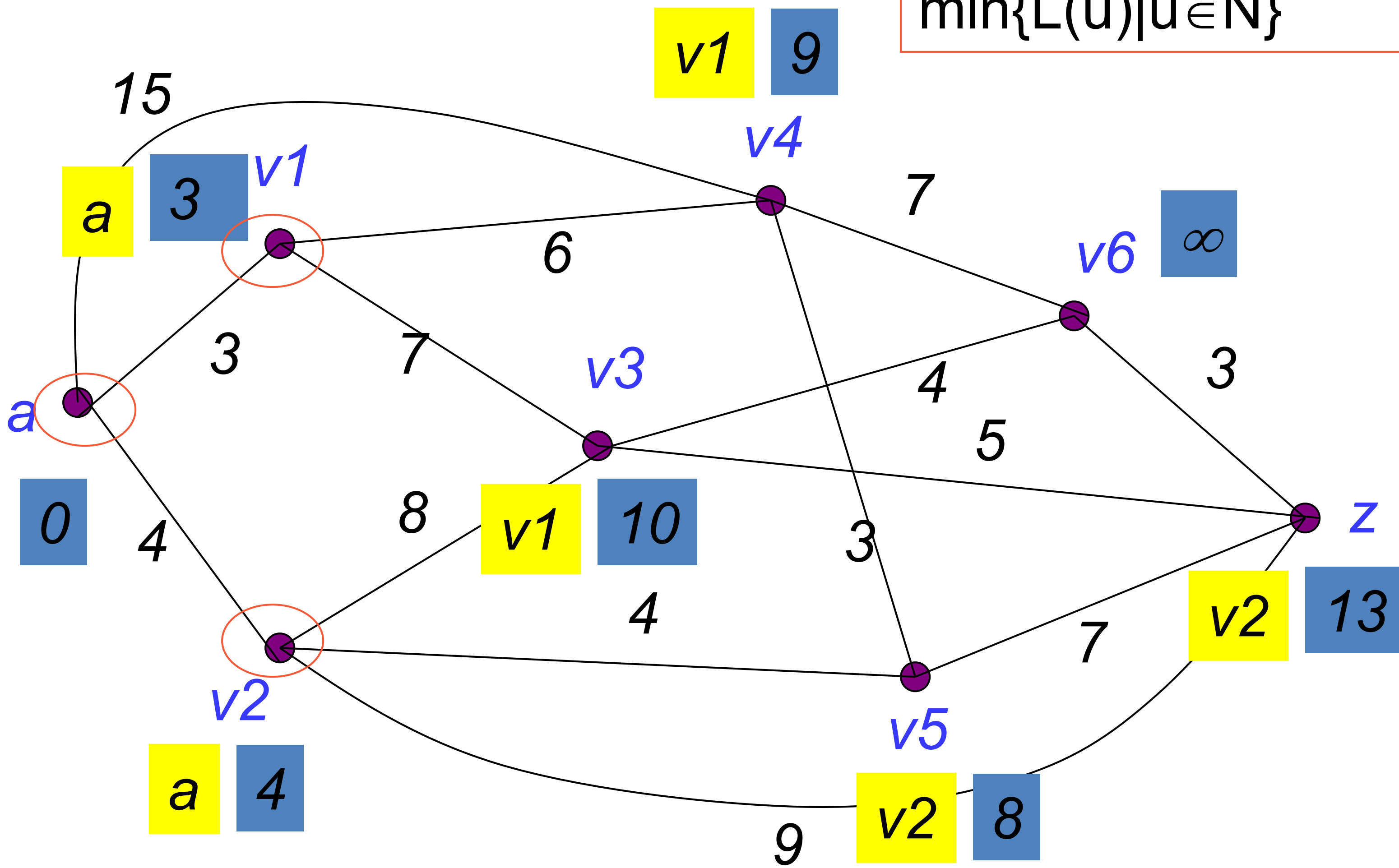
$4+9= 13 < \infty$

$L(z)=13$

$S=\{a,v1,v2\}$

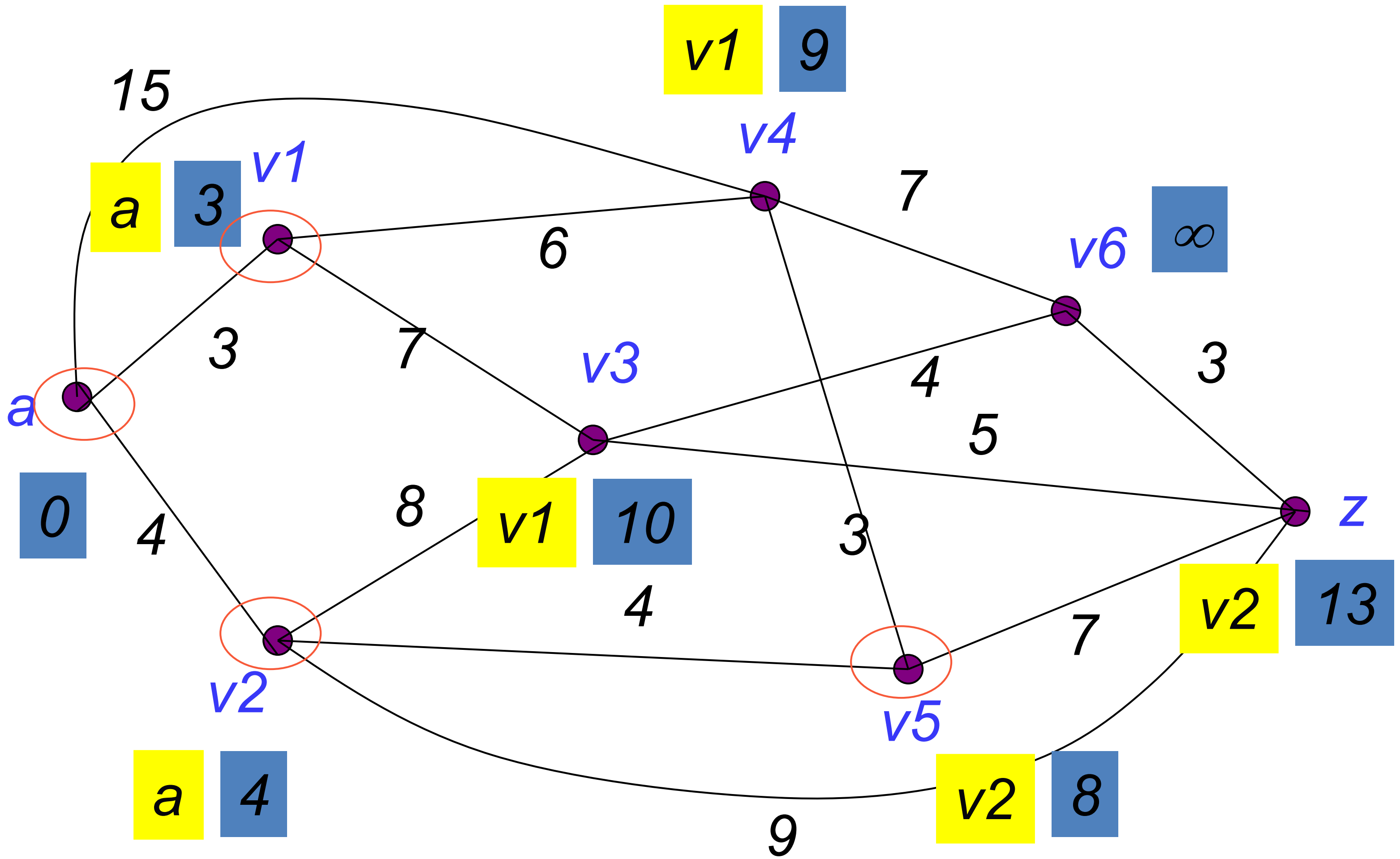
$N=\{v3,v4,v5,v6,z\}$

choose $v5$ because $L(v5)=8=\min\{L(u)|u\in N\}$



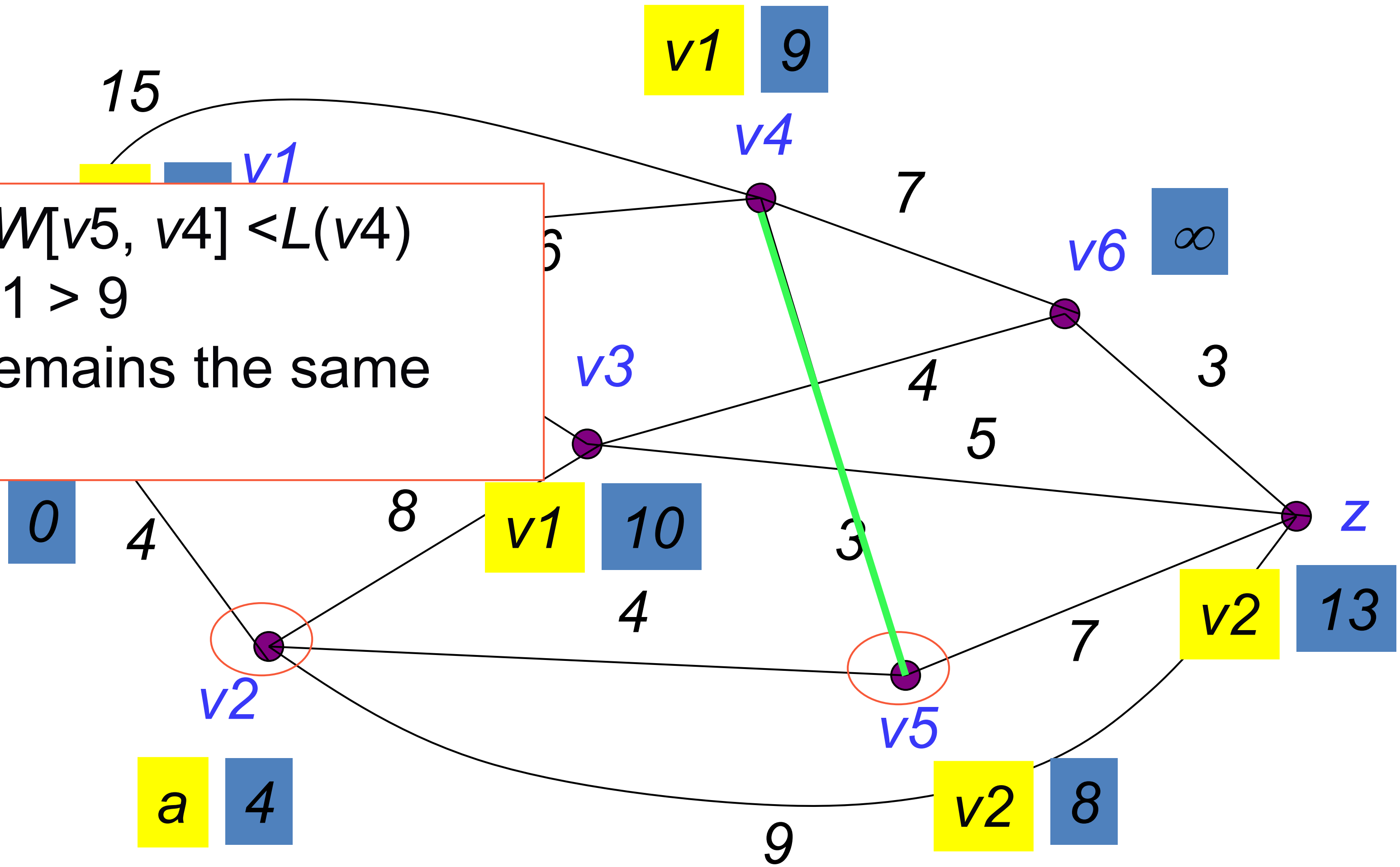
$S=\{a,v1,v2,v5\}$

$N=\{v3,v4,v6,z\}$



$S=\{a,v1,v2,v5\}$

$N=\{v3,v4,v6,z\}$



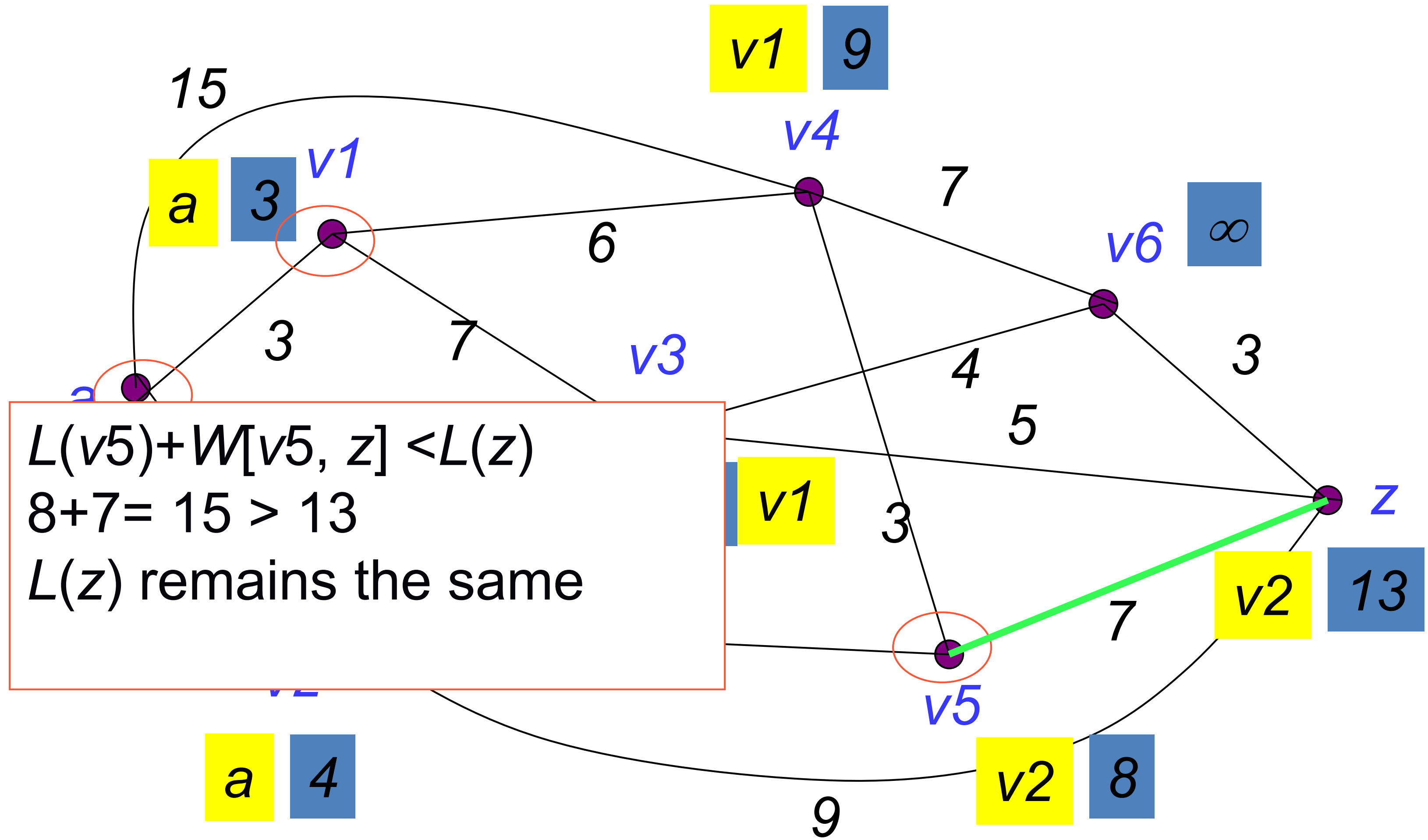
$L(v5)+W[v5, v4] < L(v4)$

$8+3= 11 > 9$

$L(v4)$ remains the same

$S=\{a,v1,v2,v5\}$

$N=\{v3,v4,v6,z\}$



$L(v5)+W[v5, z] < L(z)$

$8+7= 15 > 13$

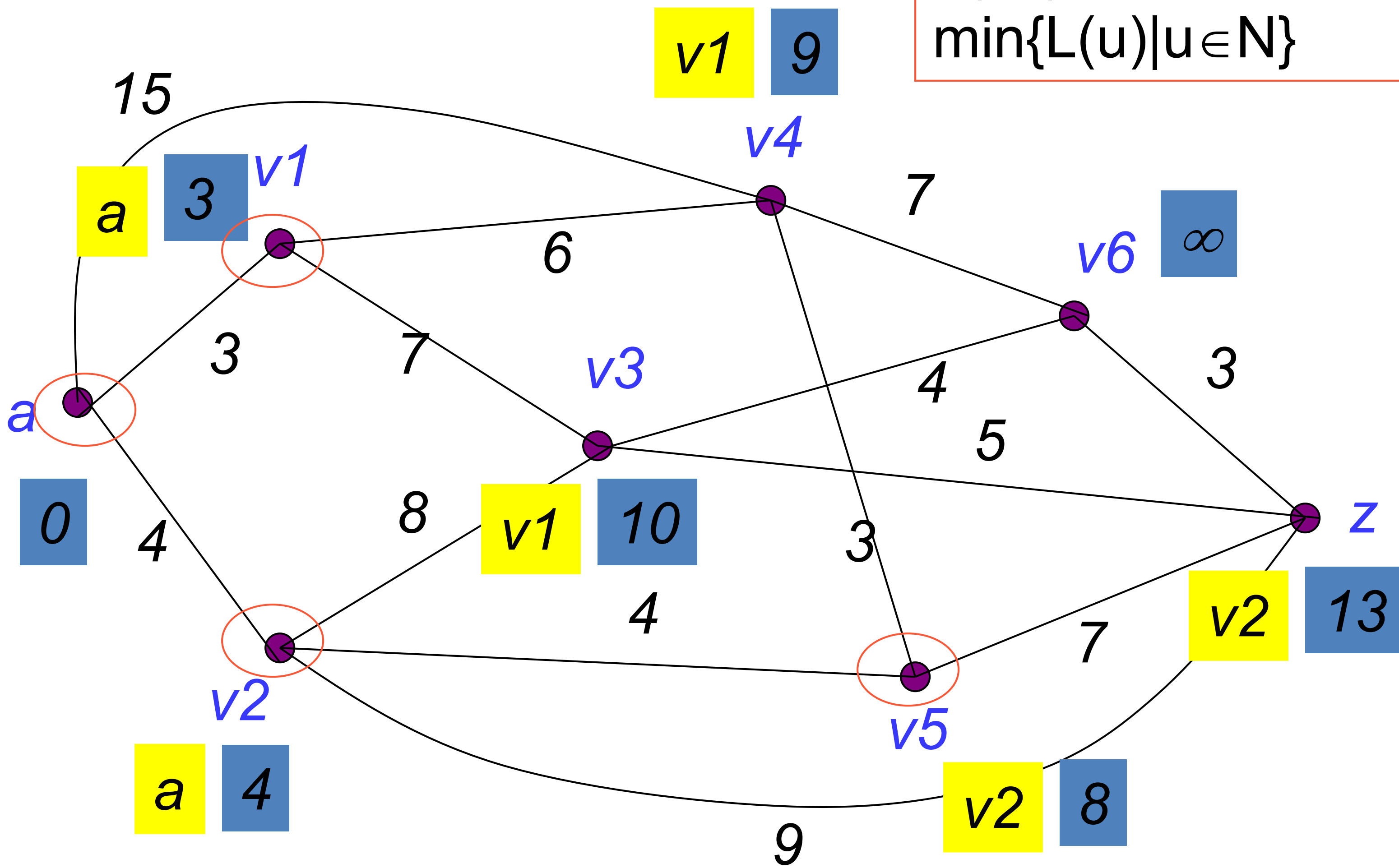
$L(z)$ remains the same

UTM UNIVERSITI TEKNOLOGI MALAYSIA

S={*a*,*v*1,*v*2,*v*5}

N= {*v*3,*v*4,*v*6,*z*}

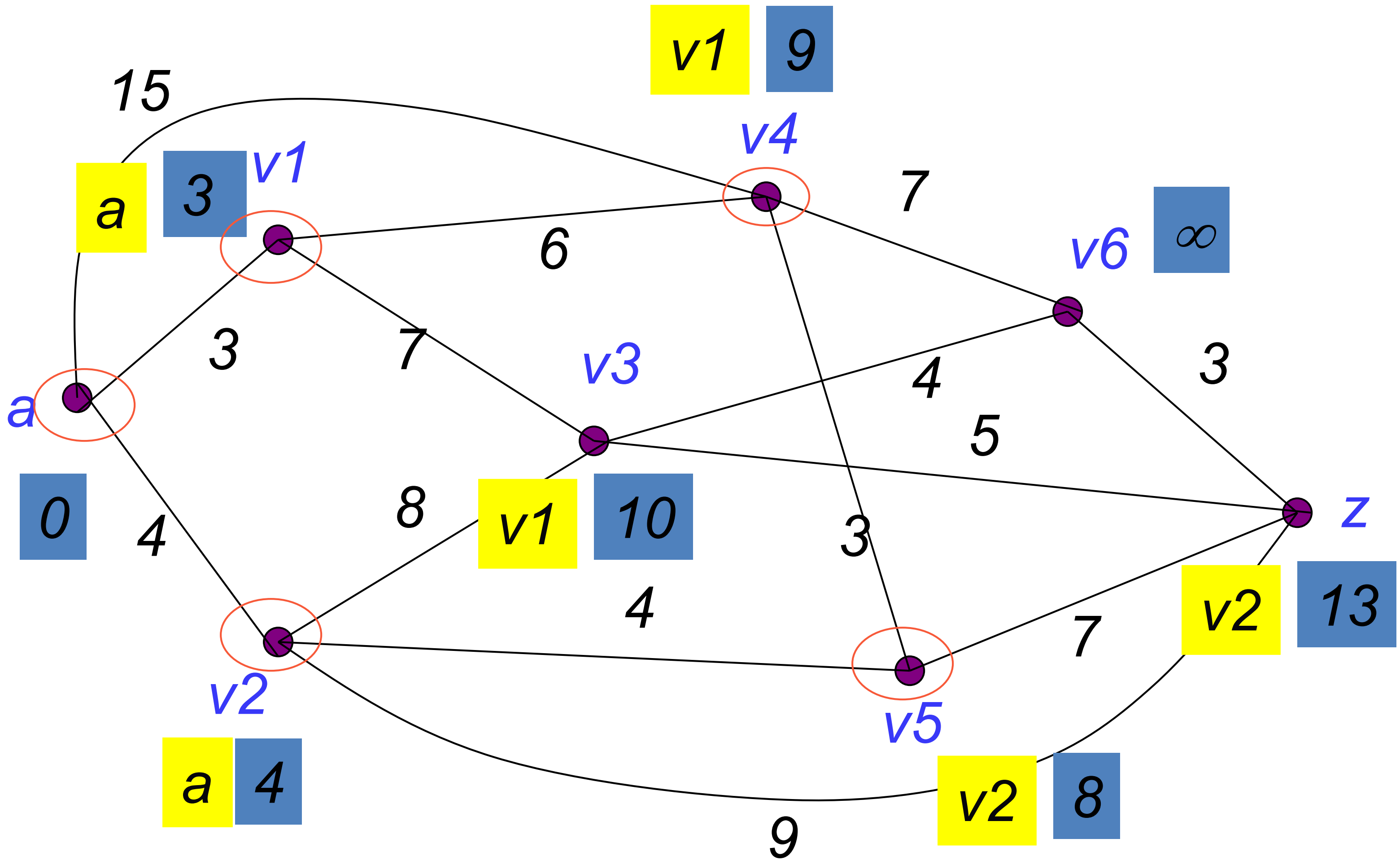
choose *v*4 because *L*(*v*4)=9 = min{L(u)|u$\in$N}



UTM UNIVERSITI TEKNOLOGI MALAYSIA

$S = \{a, v1, v2, v4, v5\}$

$N = \{v3, v6, z\}$



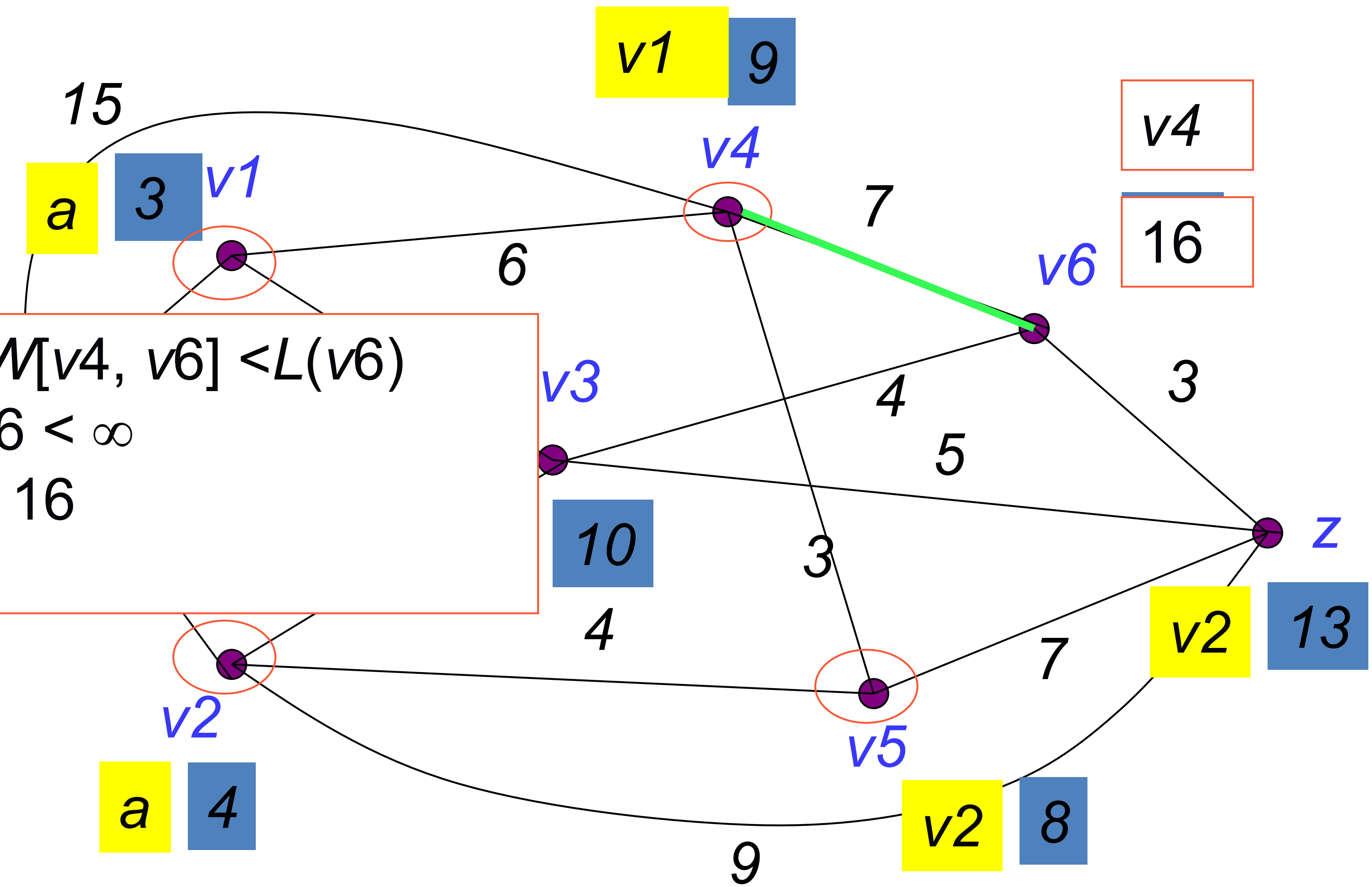
$S=\{a, v1, v2, v4, v5\}$

$N=\{v3, v6, z\}$

$L(v4)+W[v4, v6] < L(v6)$

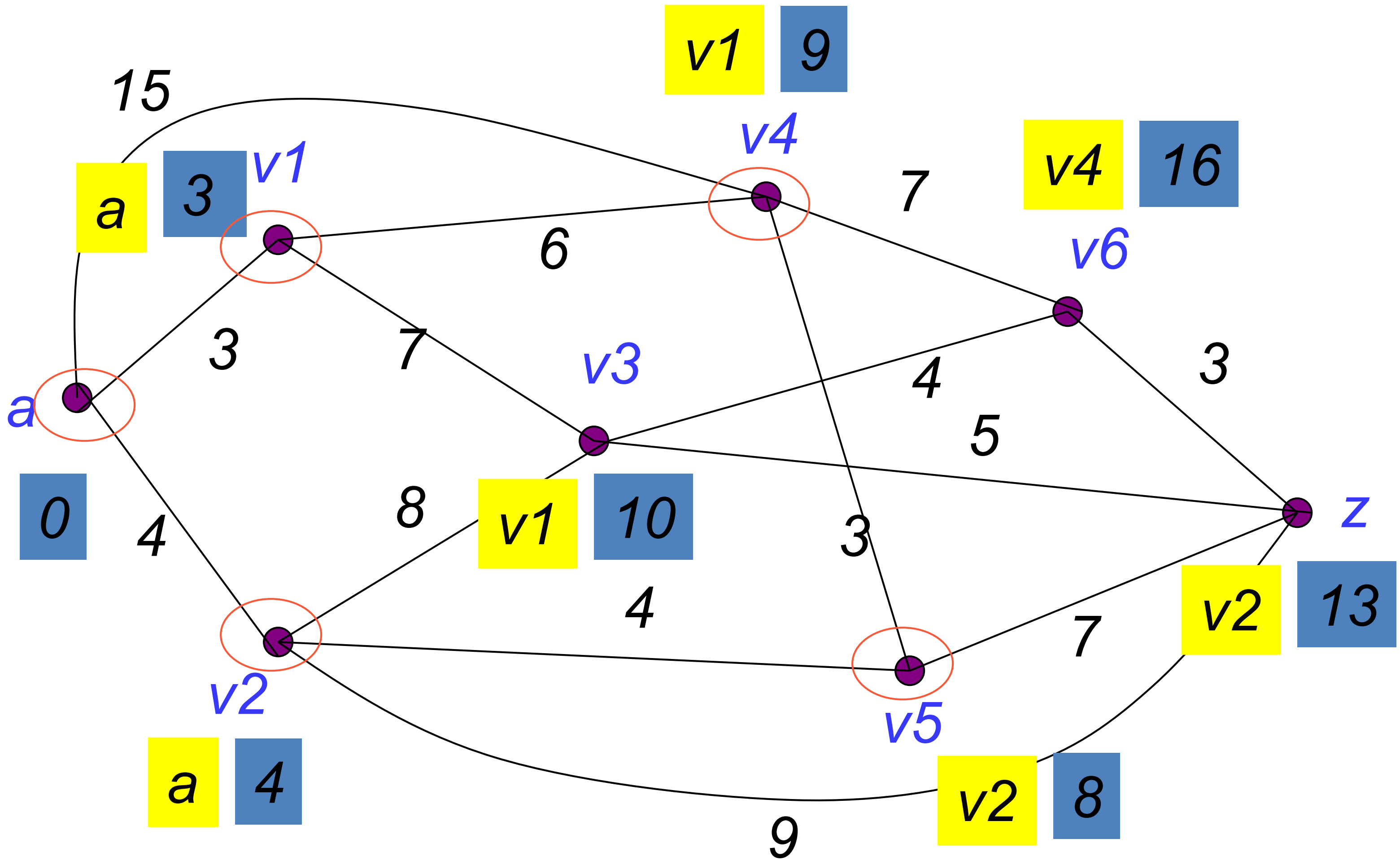
$9+7=16 < \infty$

$L(v6) = 16$



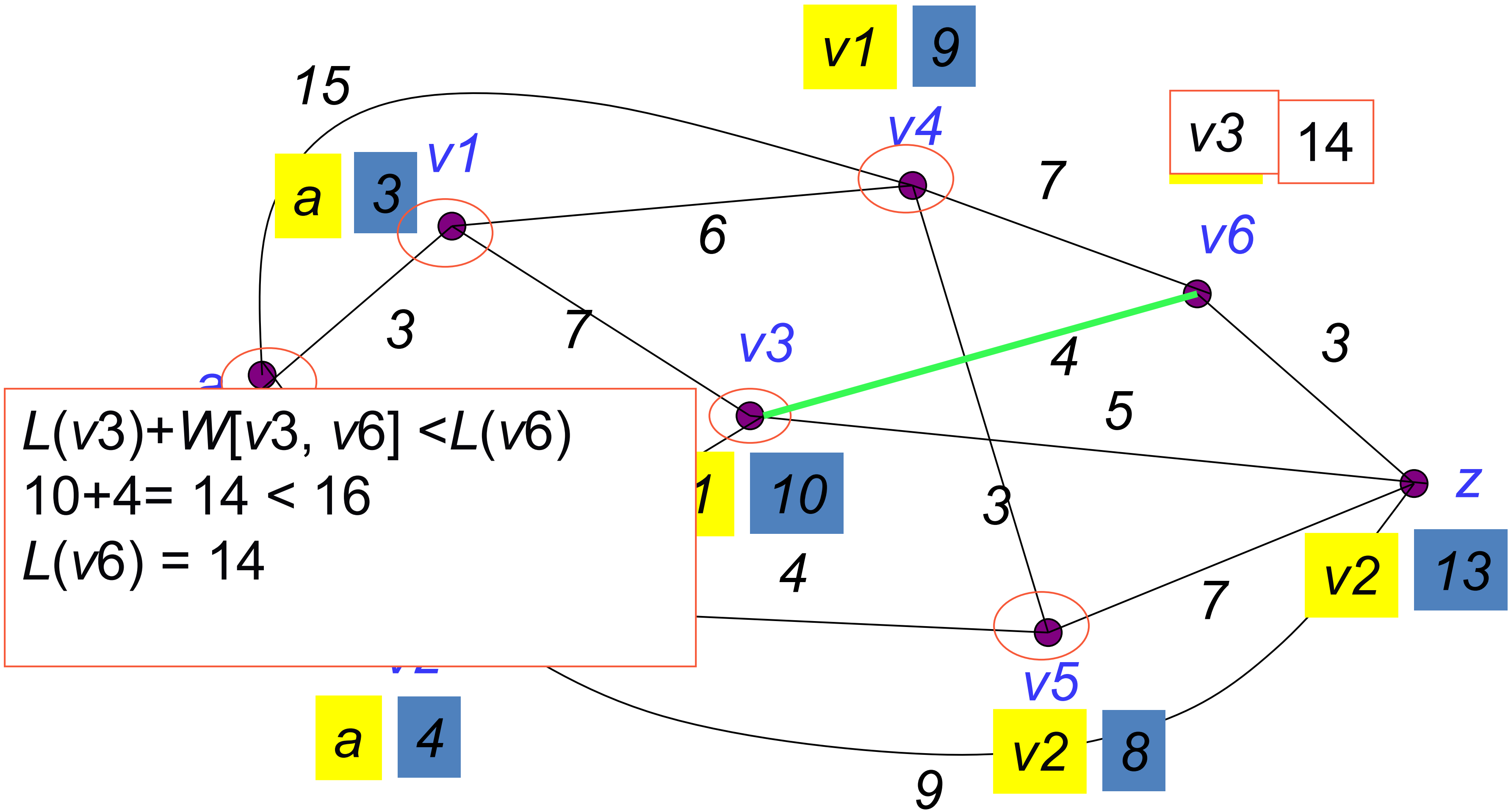
$S=\{a, v1, v2, v4, v5\}$
$N=\{v3, v6, z\}$

choose $v3$
because $L(v3)=10$
$= \min\{L(u) \mid u \in N\}$



S={a,v1,v2, v3, v4,v5}

N= {v6,z}



$L(v3)+W[v3, v6] < L(v6)$

$10+4= 14 < 16$

$L(v6) = 14$

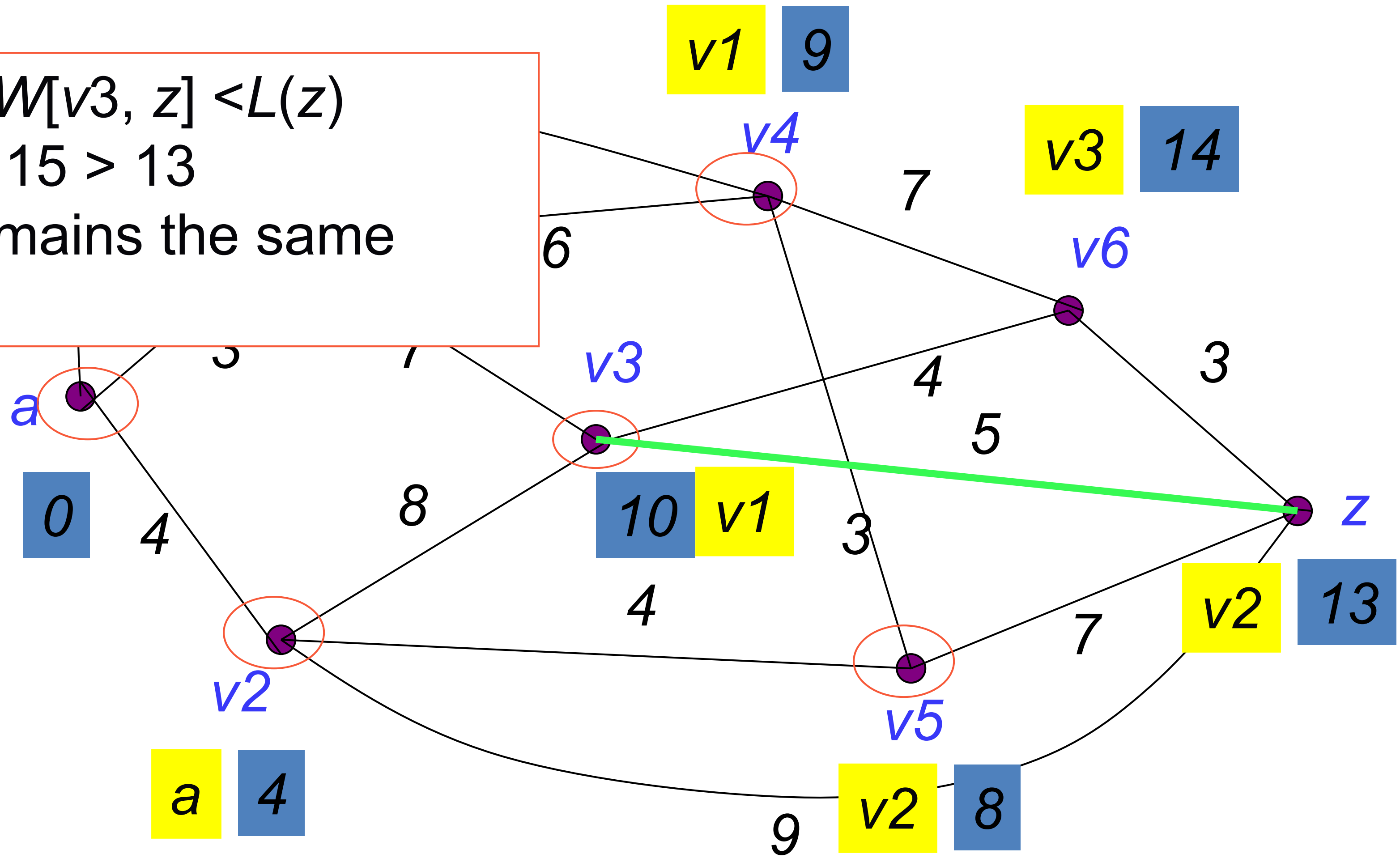
$S=\{a, v1, v2, v3, v4, v5\}$

$N=\{v6, z\}$

$L(v3)+W[v3, z] < L(z)$

$10+5=15 > 13$

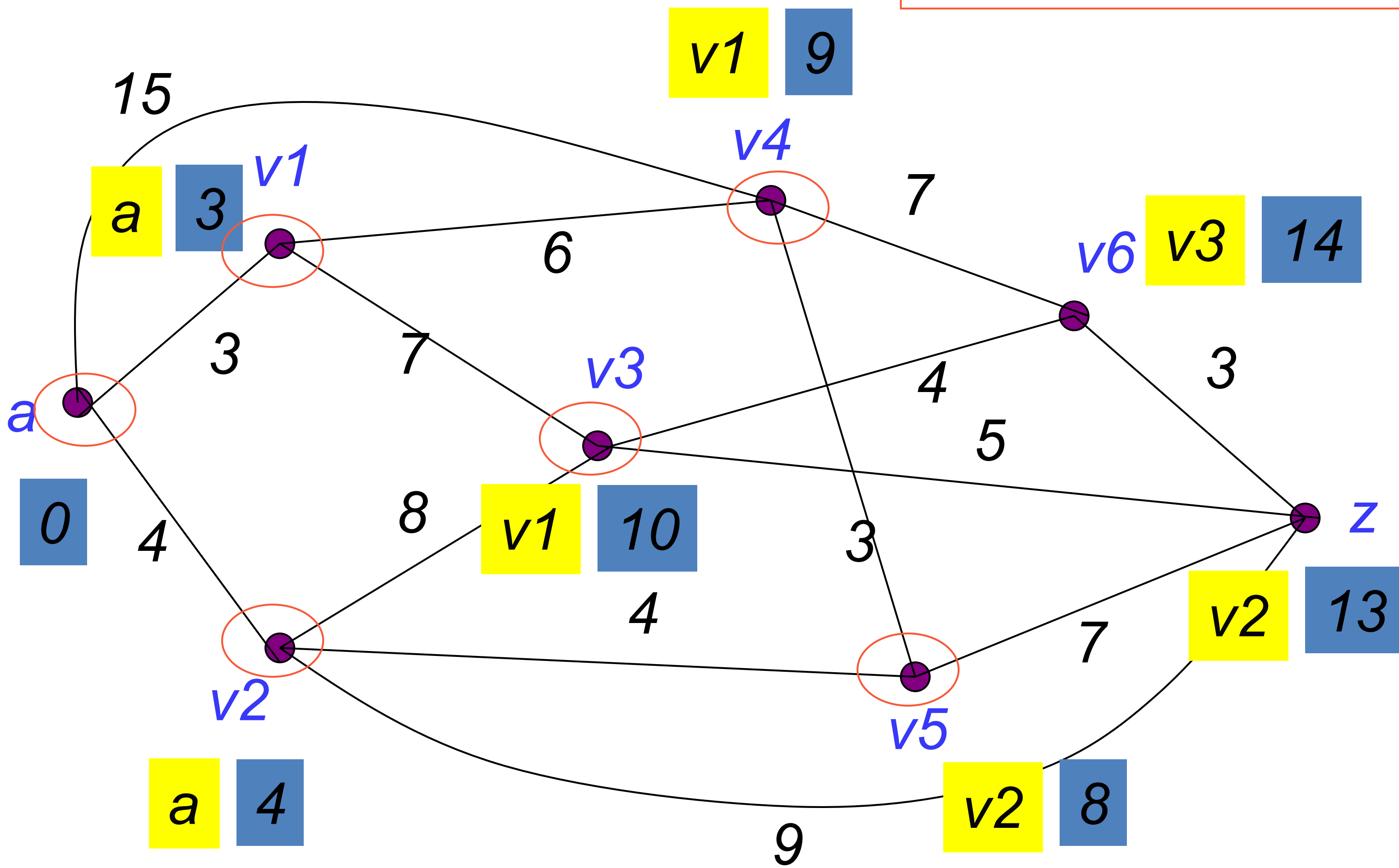
$L(z)$ remains the same



$S=\{a, v1, v2, v3, v4, v5\}$

$N= \{v6, z\}$

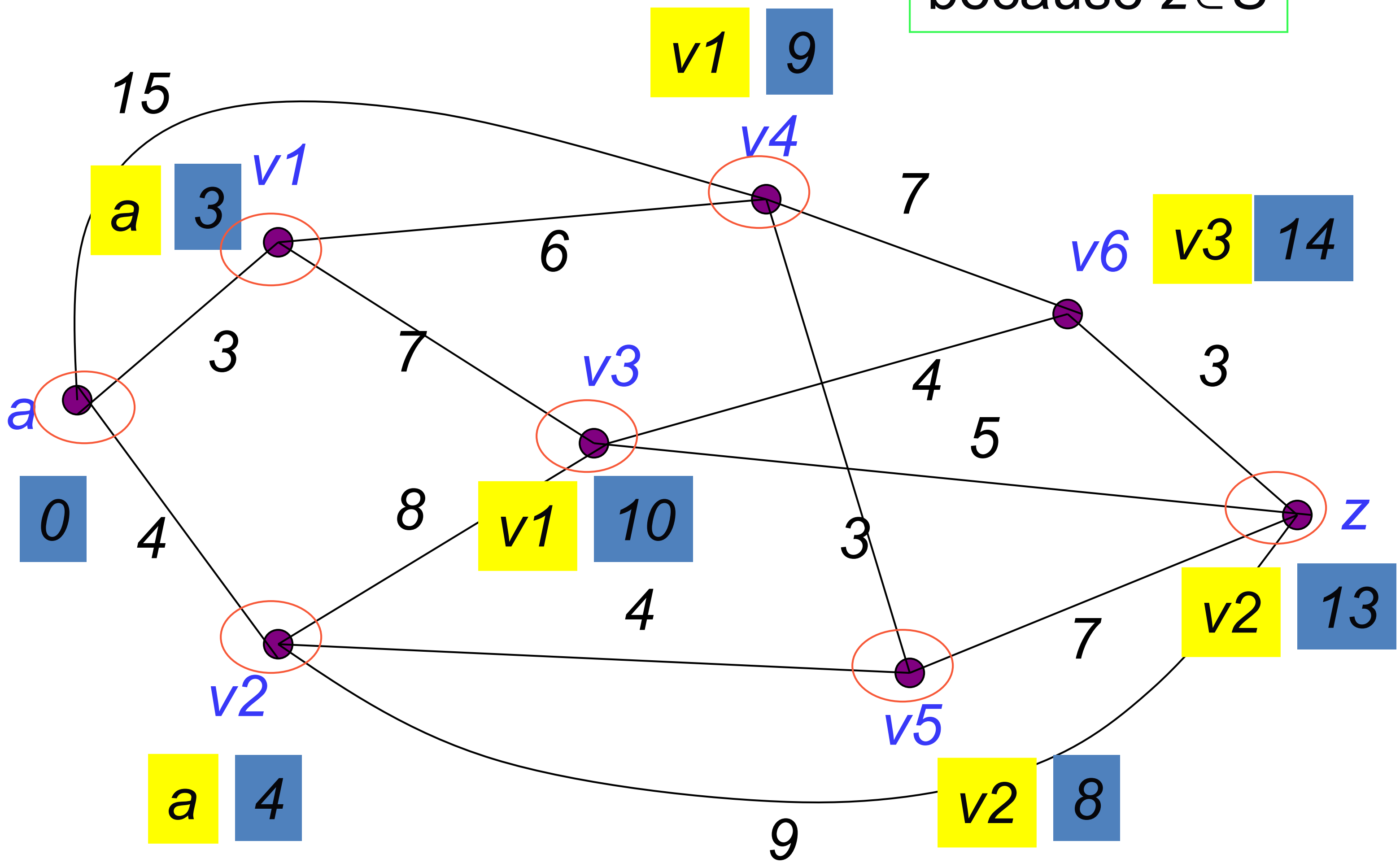
choose z because $L(z)=13 = \min\{L(u) | u \in N\}$



$S=\{a, v1, v2, v3, v4, v5, z\}$

$N=\{v6\}$

The loop terminates because $z \in S$



Shortest path from *a* to *z*

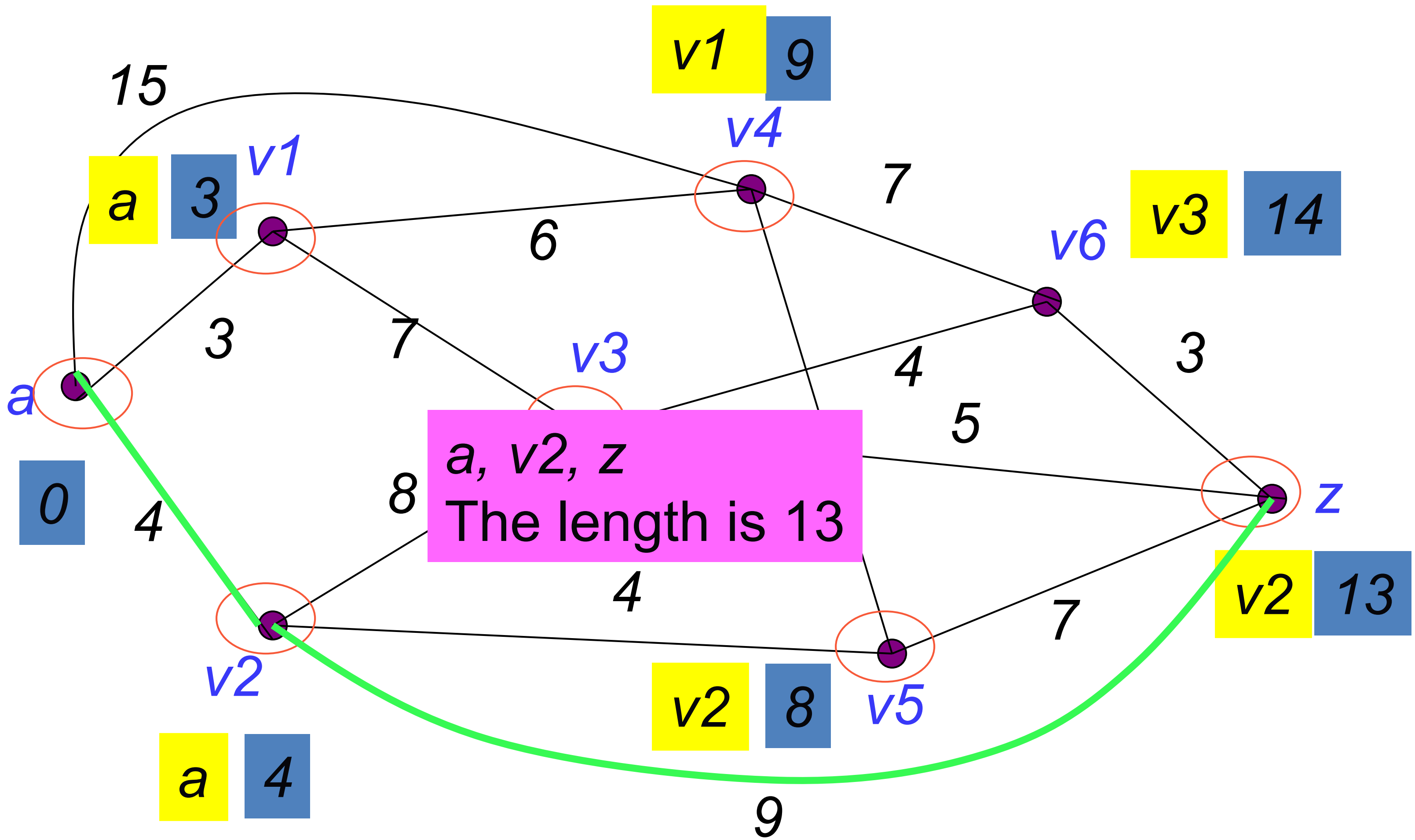
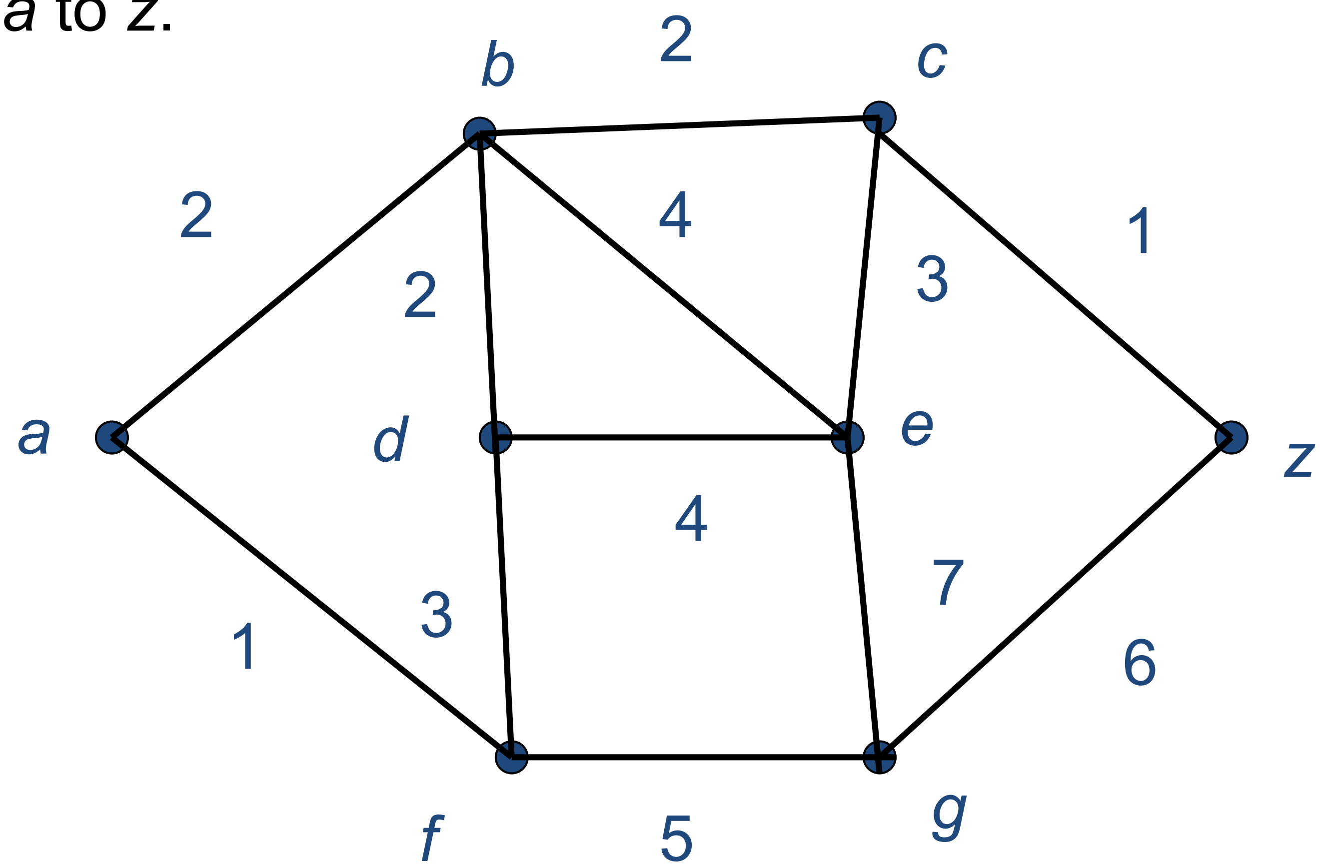


Table – Djikstra Algorithm

No.	S	N	L(a)	$L(V_1)$	$L(V_2)$	$L(V_3)$	$L(V_4)$	$L(V_5)$	$L(V_6)$	$L(z)$
0	{ }	$\{a, V_1, V_2, V_3, V_4, V_5, V_6, z\}$	0	∞	∞	∞	∞	∞	∞	∞
1	$\{a\}$	$\{V_1, V_2, V_3, V_4, V_5, V_6, z\}$		3	4	∞	15	∞	∞	∞
2	$\{a, V_1\}$	$\{V_2, V_3, V_4, V_5, V_6, z\}$		**3**	4	10	9	∞	∞	∞
3	$\{a, V_1, V_2\}$	$\{V_3, V_4, V_5, V_6, z\}$			**4**	10	9	8	∞	13
4	$\{a, V_1, V_2, V_5\}$	$\{V_3, V_4, V_6, z\}$				10	9	**8**	∞	13
5	$\{a, V_1, V_2, V_5, V_4\}$	$\{V_6, z\}$				10	**9**		16	13
6	$\{a, V_1, V_2, V_5, V_4, V_3,\}$	$\{V_6, z\}$				**10**			14	13
7	$\{a, V_1, V_2, V_5, V_4, V_3, z\}$	$\{V_6\}$							14	**13**

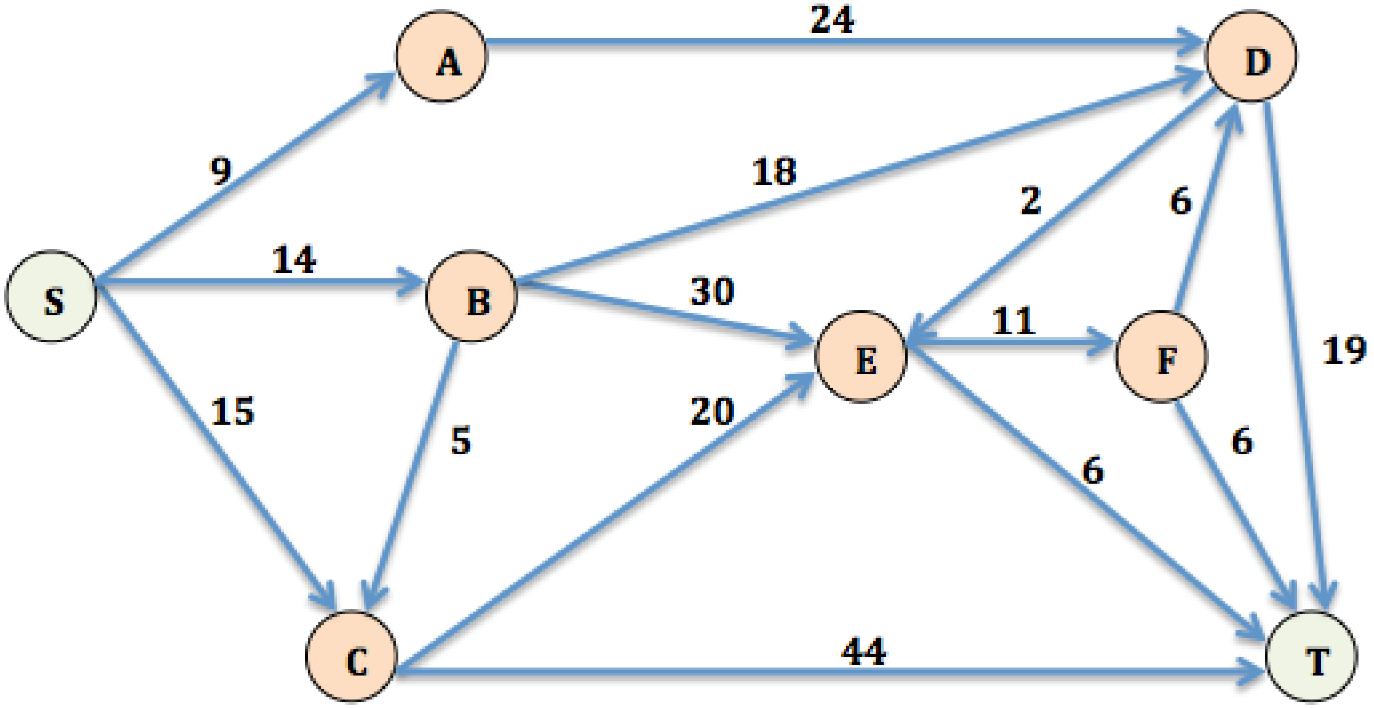
exercise

Use Dijkstra's algorithm to find the length of a shortest path from *a* to *z*.



exercise

Q: Given a weighted digraph, find the shortest path from S to T, using Djik



Note: Weights are arbitrary numbers (i.e., not necessarily distances).

Exercise Past Year 2015/2016

The network in Figure 5 gives the distances in miles between pairs of cities A, B, …, and H.

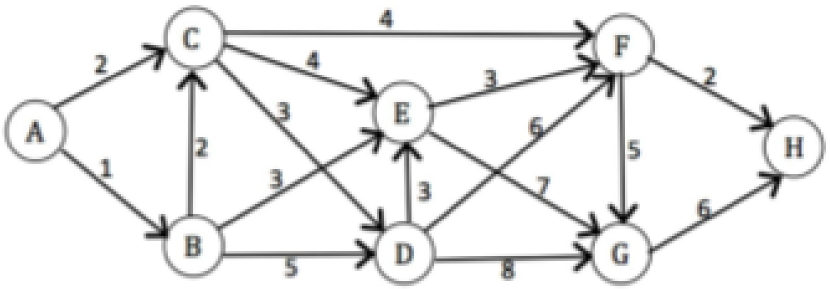


Figure 5

a) Based on Dijkstra's algorithm, complete Table 1 to find the shortest path from city A to city H. (Note: Copy Table 1 into your answer booklet).

(8 marks)

Table 1

Iteration	S	N	*L*(A)	*L*(B)	*L*(C)	*L*(D)	*L*(E)	*L*(F)	*L*(G)	*L*(H)
0										
1										
2										
3										
4										
5										
6										
7										

b) State the minimum distance and the shortest path from city 1 to city 8.

(2 marks)