Finals -Lecture Notes #4 – Kruskal, Prims, Dijskstra, & Traveling Salesman 05.10.23

REMINDER!

1) ALWAYS FOLLOW MY CLASSROOM PROCEDURES.

2) ACTIVITIES ARE ALWAYS UPDATED IN GOOGLE CLASSROOM, MYOPENMATH

3) ALWAYS SUBMIT YOUR WORK ON TIME. IF YOU MISSED SOME ACTIVITIES, BE RESPONSIBLE ENOUGH TO DO THEM.

4) THE WEEKLY HOMEWORK IS ALWAYS DUE EVERY MONDAY.

5) CHECK THE UPDATES ON GOOGLE CLASSROOM AND DO YOUR MISSING ASSIGNMENTS as the GRADES are always UPDATED.

Activities for today

MyOpenMath

1) **Bellwork#4 (Finals) – Let's Do It ALL 05.10.23.**
2) **Activity # 4 (Finals) – All About Trees 05.10.23**
3) **Finals - Weekly Homework #4 – Reviewing is Fun 05.10-05.16.23**
4) **Finals – Lecture Notes #4** More on Graph Theory & Spanning Tree 05.10.2023

At the end of the lesson, students will be able to

1) **identify graph theory.**
2) **problems associated with the graph theory and spanning tree**

Minimum Cost Spanning Tree (India)

https://www.youtube.com/watch?v=4ZlRH0eK-qQ

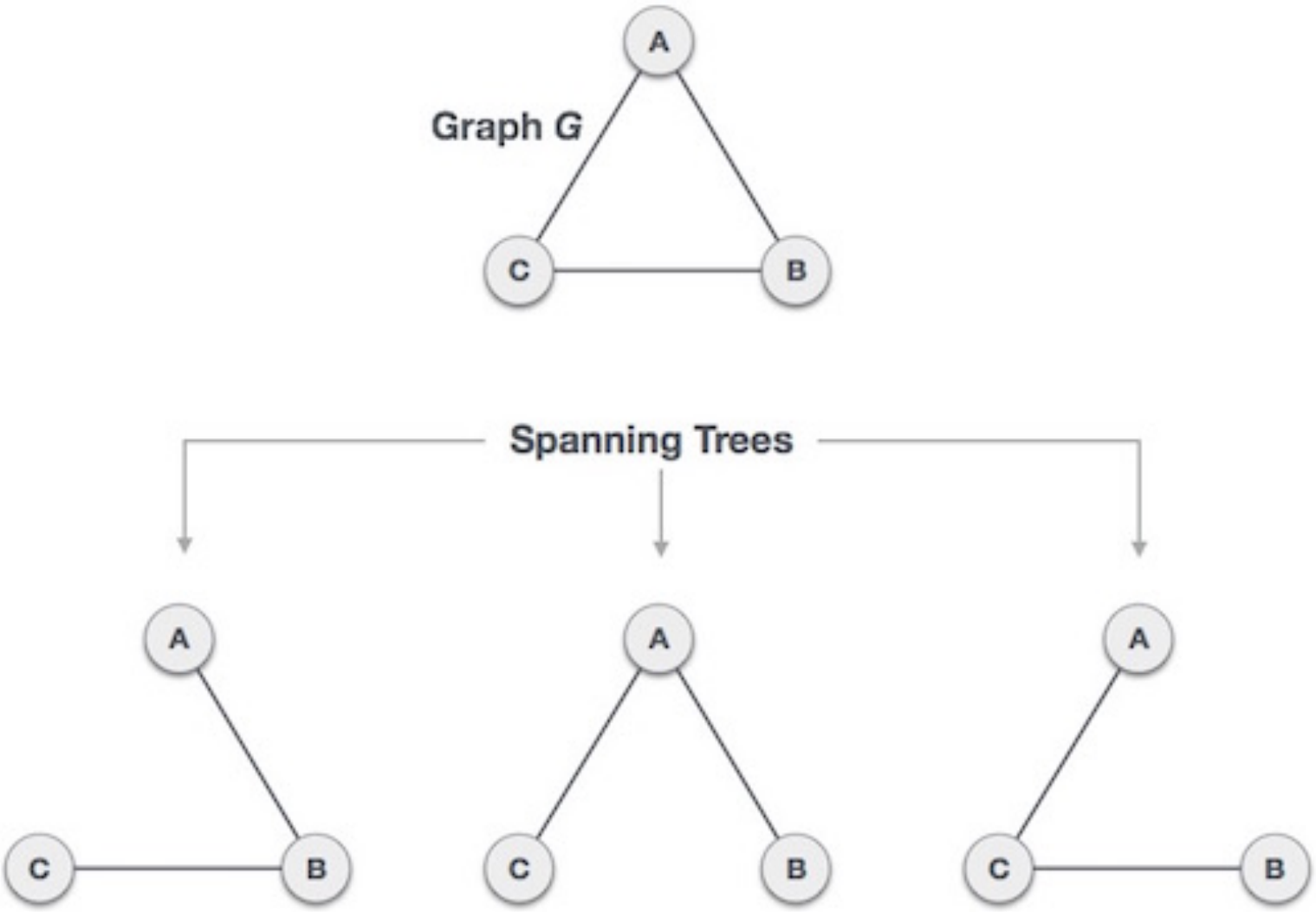
Spanning Tree

Source: https://www.tutorialspoint.com/data_structures_algorithms/spanning_tree.htm

Video: https://www.youtube.com/watch?v=0ljjRM8hWjU

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.



Note: We found three spanning trees off one complete graph. A complete undirected graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees, where **n** is the number of nodes. In the above addressed example, **n is 3,** hence $\mathbf{3^{3-2} = 3}$ spanning trees are possible.

General Properties of Spanning Tree

We now understand that one graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G –

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.

- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e., the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

Mathematical Properties of Spanning Tree

- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).
- From a complete graph, by removing maximum **e – n + 1** edges, we can construct a spanning tree.
- A complete graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees.

Thus, we can conclude that spanning trees are a subset of connected Graph G and disconnected graphs do not have spanning tree.

Application of Spanning Tree

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common applications of spanning trees are –

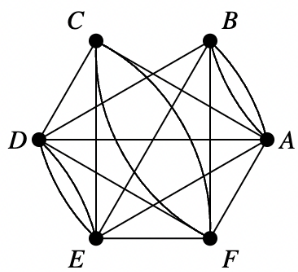
- **Civil Network Planning**
- **Computer Network Routing Protocol**
- **Cluster Analysis**

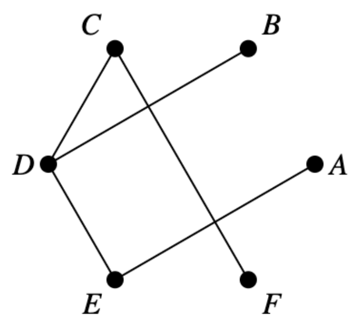
Let us understand this through a small example.

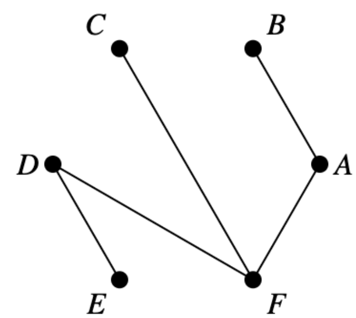
Example: Consider, city network as a huge graph and now plans to deploy telephone lines in such a way that in minimum lines we can connect to all city nodes. This is where the spanning tree comes into picture.

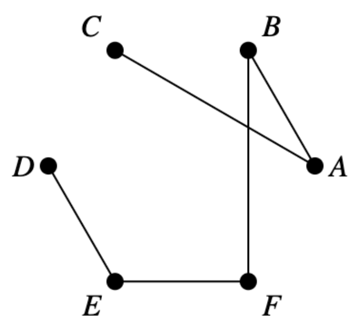
CHECK FOR UNDERSTANDING 2

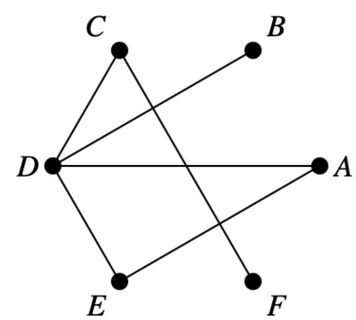
Question #1) Select the graphs that are spanning trees for the given graph.

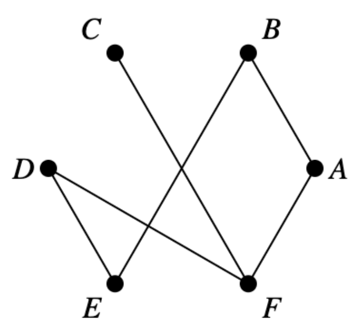


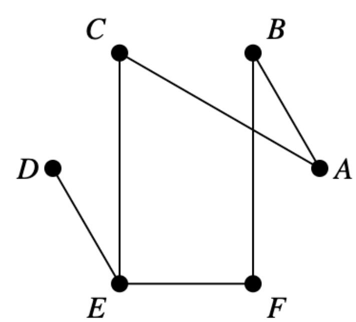
☐ 

☐ 

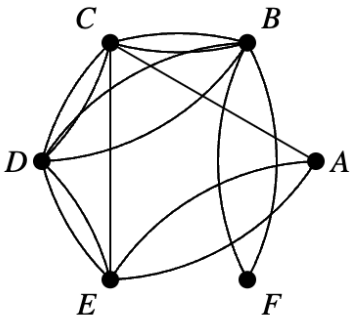
☐ 

☐ 

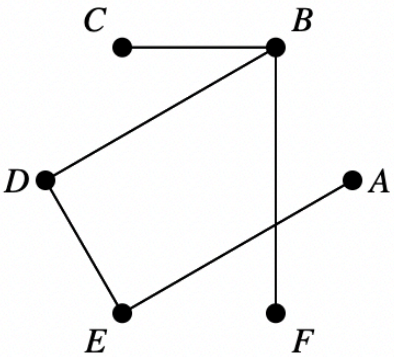
☐ 

☐ 

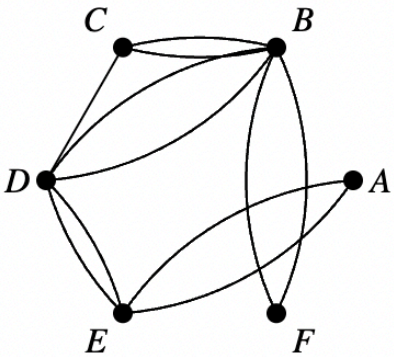
Question #2) Select the graphs that are spanning trees for the given graph.



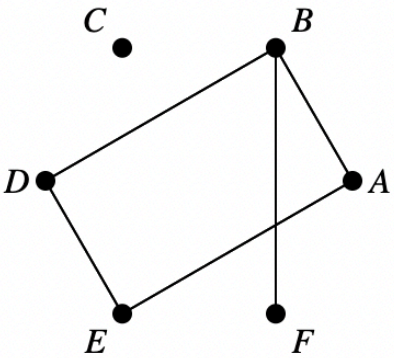
○



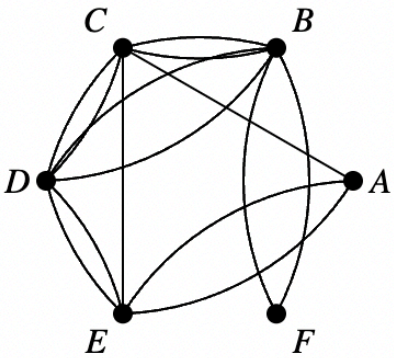
○



○



○



Minimum Spanning Tree

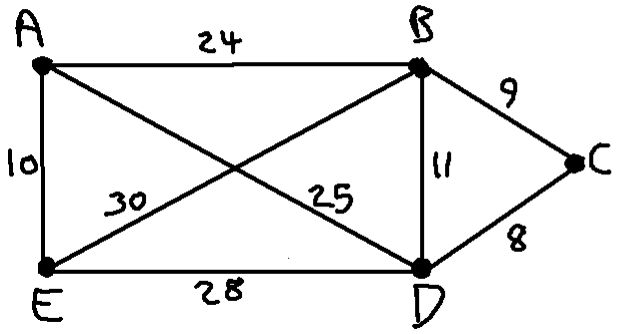
A minimum spanning tree is the tree that spans all of the vertices in a problem with the least cost (or time, or distance).

A tree that is created from a weighted graph such that the tree is of minimum possible total weight is called a/an ***minimum spanning tree.***

Key Idea. Spanning is **the amount of area or the amount of time that something encompasses**. An example of span is how long you live. An example of span is a house on three acres.

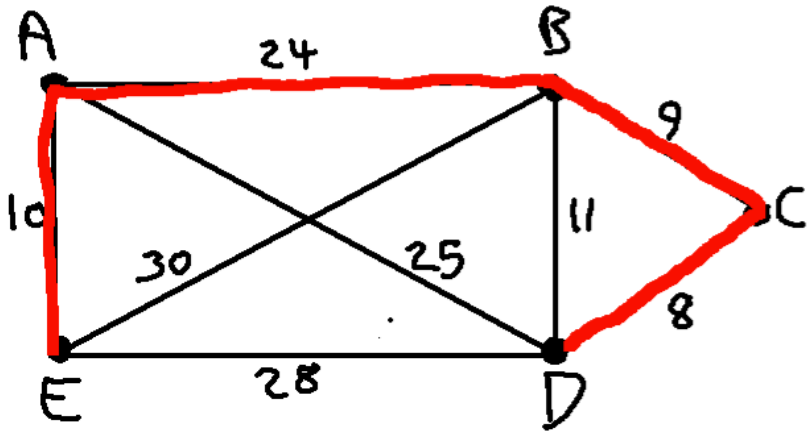
Additional Idea. In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

Example 6.2.4: Minimum Spanning Tree



The above is a weighted graph where the numbers on each edge represent the cost of each edge. We want to find the minimum spanning tree of this graph so that we can find a network that will reach all vertices for the least total cost.

Figure 6.2.6: Minimum Spanning Tree for Weighted Graph 1



This is the minimum spanning tree for the graph with a total cost of 51.

Minimum Spanning-Tree Algorithm

We shall learn about two most important spanning tree algorithms here –

- Kruskal's Algorithm
- Prim's Algorithm

Kruskal's Algorithm

Videos: https://www.youtube.com/watch?v=JZBQLXgSGfs

https://www.youtube.com/watch?v=Yo7sddEVONg

https://www.youtube.com/watch?v=d4BEgzK08JE&t=184s

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

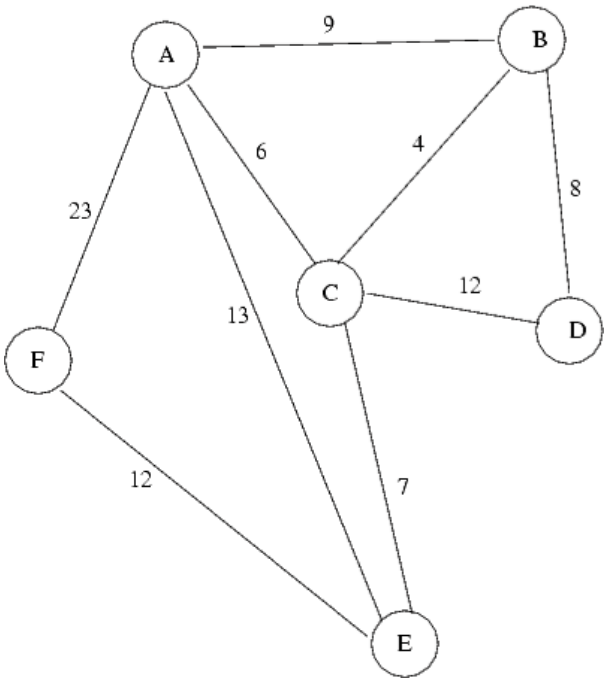
Key Idea. A procedure that produces the minimum spanning tree of a graph is called *Kruskal's* Algorithm. The algorithm specifies that one is always to choose an edge with the least possible *weight* while avoiding the creation of any *circuits*.

Kruskal's Algorithm: Since some graphs are much more complicated than the previous example, we can use Kruskal's Algorithm to always be able to find the minimum spanning tree for any graph.

1. Find the cheapest link in the graph. If there is more than one, pick one at random. Mark it in red.
2. Find the next cheapest link in the graph. If there is more than one, pick one at random. Mark it in red.
3. Continue doing this as long as the next cheapest link does not create a red circuit.
4. You are done when the red edges span every vertex of the graph without any circuits. The red edges are the MST (minimum spanning tree).

Example 6.2.5: Using Kruskal's Algorithm Figure

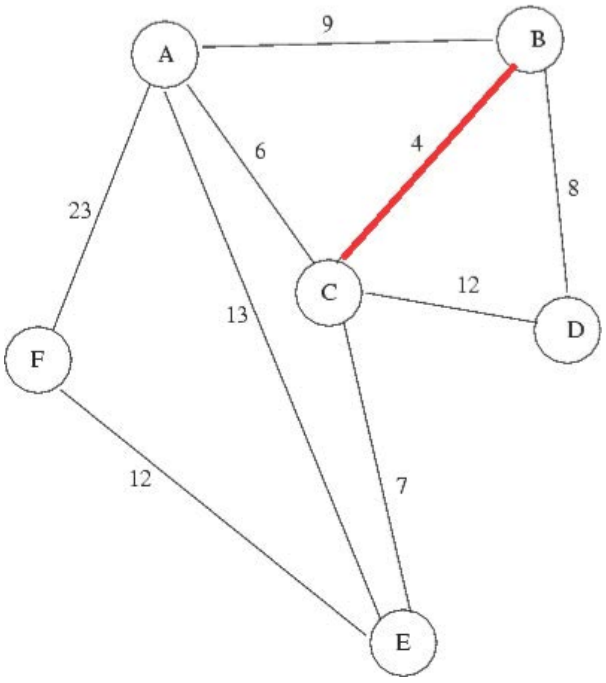
6.2.7: Weighted Graph 2



Suppose that it is desired to install a new fiber optic cable network between the six cities (A, B, C, D, E, and F) shown above for the least total cost. Also, suppose that the fiber optic cable can only be installed along the roadways shown above. The weighted graph above shows the cost (in millions of dollars) of installing the fiber optic cable along each roadway. We want to find the minimum spanning tree for this graph using Kruskal's Algorithm.

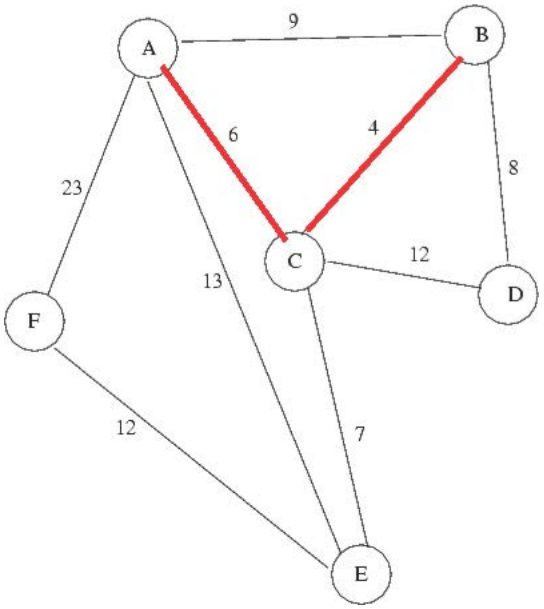
Step 1: Find the cheapest link of the whole graph and mark it in red. The cheapest link is between B and C with a cost of four million dollars.

Figure 6.2.8: Kruskal's Algorithm Step 1



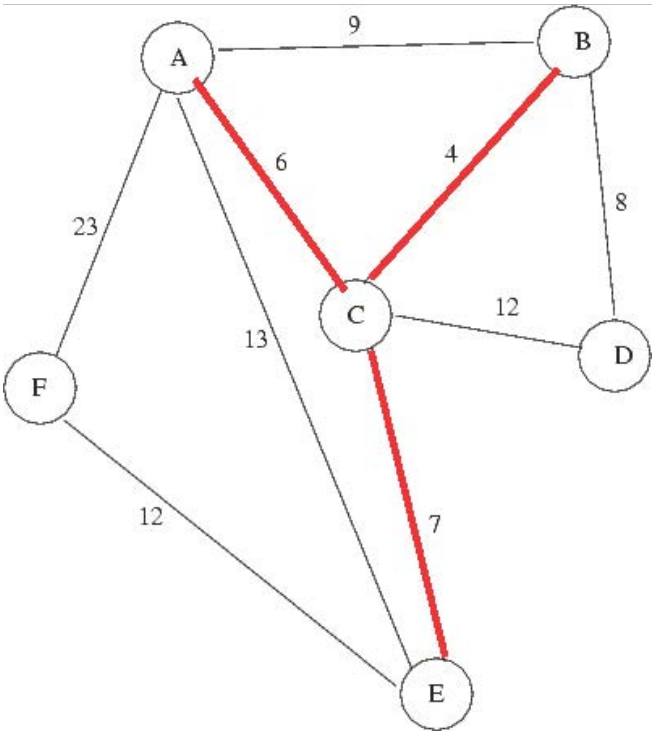
Step 2: Find the next cheapest link of the whole graph and mark it in red. The next cheapest link is between A and C with a cost of six million dollars.

Figure 6.2.9: Kruskal's Algorithm Step 2



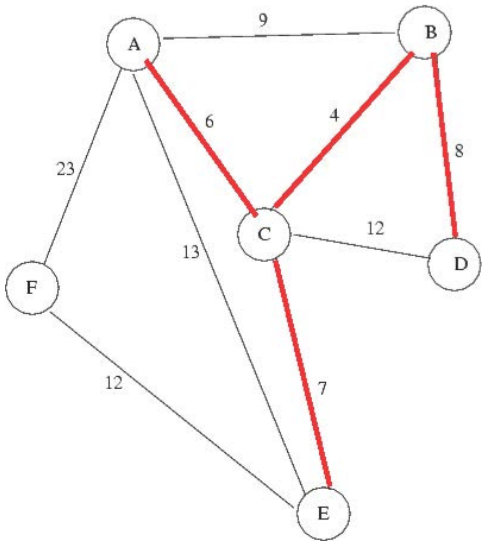
Step 3: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit. The next cheapest link is between C and E with a cost of seven million dollars.

Figure 6.2.10: Kruskal's Algorithm Step 3



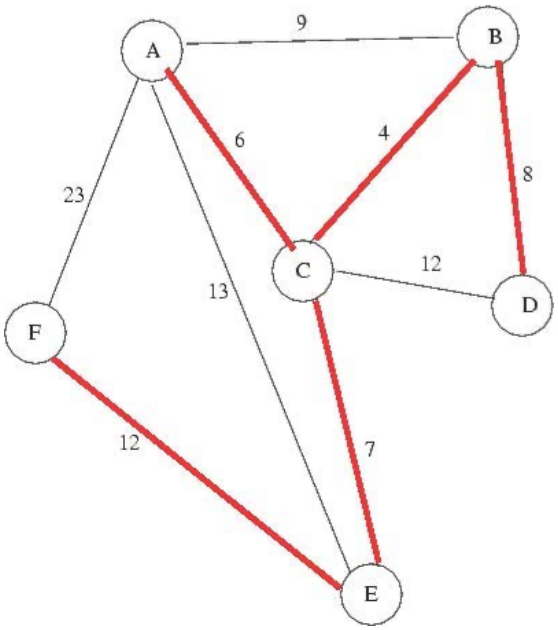
Step 4: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit. The next cheapest link is between B and D with a cost of eight million dollars.

Figure 6.2.11: Kruskal's Algorithm Step 4



Step 5: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit. The next cheapest link is between A and B with a cost of nine million dollars, but that would create a red circuit so we cannot use it. Therefore, the next cheapest link after that is between E and F with a cost of 12 million dollars, which we are able to use. We cannot use the link between C and D which also has a cost of 12 million dollars because it would create a red circuit.

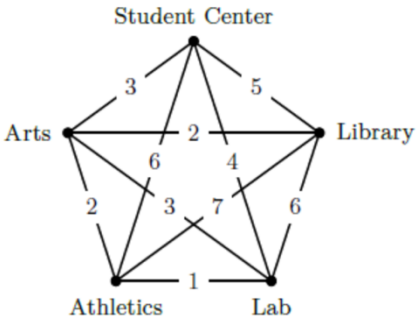
Figure 6.2.12: Kruskal's Algorithm Step 5



This was the last step, and we now have the minimum spanning tree for the weighted graph with a total cost of $37,000,000.

CHECK FOR UNDERSTANDING 3

1) A maintenance team is responsible for a group of five buildings on campus. These buildings are shown in the graph below, with the distance given between each pair of buildings. After a blizzard, the team is tasked with clearing the snow, but there is not enough time to clear all the walkways.

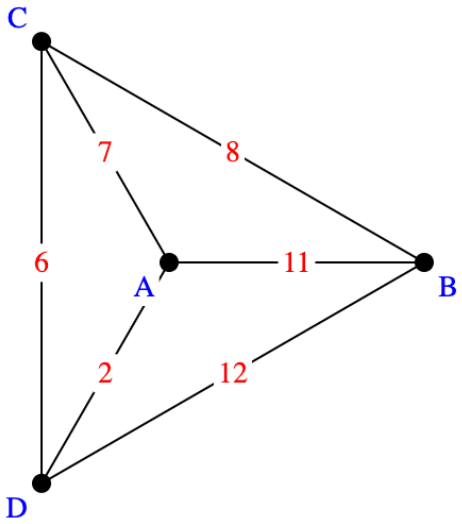


Which walkways should the maintenance team plow in order to connect all the buildings, while minimizing the time needed to do so (really, by minimizing the distance)?

- ☐ Student Center, Library
- ☐ Student Center, Lab
- ☐ Student Center, Athletics
- ☐ Student Center, Arts
- ☐ Library, Lab
- ☐ Library, Athletics
- ☐ Library, Arts
- ☐ Lab, Athletics
- ☐ Lab, Arts
- ☐ Athletics, Arts

What is the total length of walkway that will be plowed?

2) Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. Which of the edges below are included in the minimum cost tree?



A) CD

B) AB

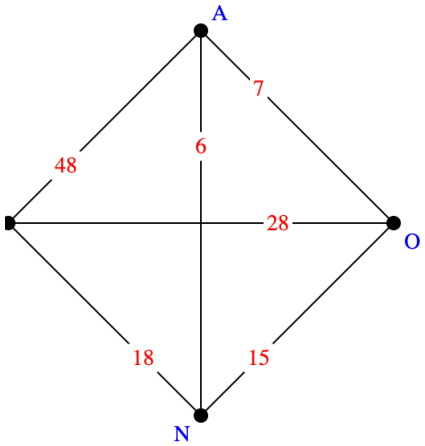
C) AD

D) BD

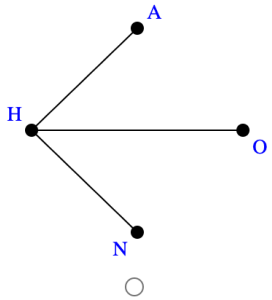
E) AC

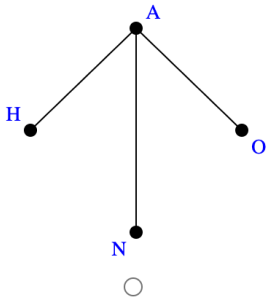
F) BC

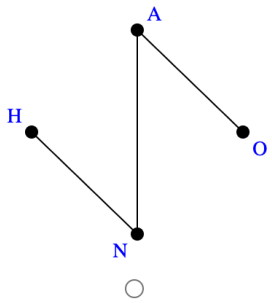
3) Consider the connected, weighted graph below.

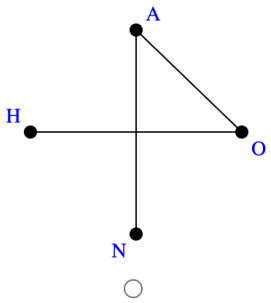


Use Kruskal's Algorithm to find the minimum spanning tree of the graph. Select the correct shape below.









Determine the total weight of the minimum spanning tree.

The total weight of the minimum spanning tree is _____.

4) A power company needs to lay updated distribution lines connecting eight cities in Virginia to the power grid. The distances between these cities are given in the table below. Design a network that will minimize the amount of new line.

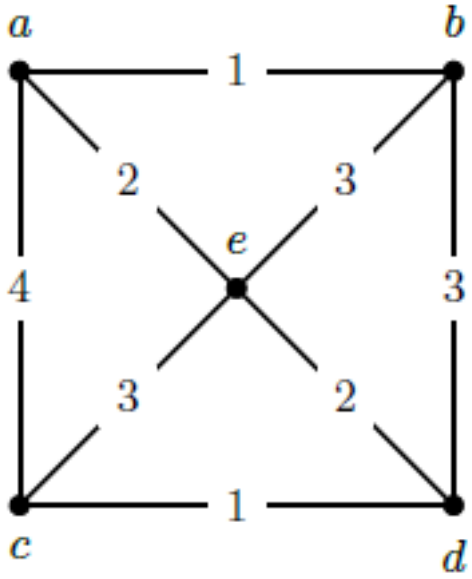
	Purcellville	Leesburg	Middleburg	Chantilly	Sterling	McLean	Arlington	Annandale
Purcellville	--	8	11	23	19	32	37	35
Leesburg	8	--	14	17	10	24	29	27
Middleburg	11	14	--	18	16	30	34	31
Chantilly	23	17	18	--	8	13	18	13
Sterling	19	10	16	8	--	15	20	17
McLean	32	24	30	13	15	--	5	7
Arlington	37	29	34	18	20	5	--	6
Annandale	35	27	31	13	17	7	6	--

What connections should be built between cities and

What is the total required length of line that must be laid?

- ☐ Purcellville, Leesburg
- ☐ Purcellville, Middleburg
- ☐ Purcellville, Chantilly
- ☐ Purcellville, Sterling
- ☐ Purcellville, McLean
- ☐ Purcellville, Arlington
- ☐ Purcellville, Annandale
- ☐ Leesburg, Middleburg
- ☐ Leesburg, Chantilly
- ☐ Leesburg, Sterling
- ☐ Leesburg, McLean
- ☐ Leesburg, Arlington
- ☐ Leesburg, Annandale
- ☐ Middleburg, Chantilly
- ☐ Middleburg, Sterling
- ☐ Middleburg, McLean
- ☐ Middleburg, Arlington
- ☐ Middleburg, Annandale
- ☐ Chantilly, Sterling
- ☐ Chantilly, McLean
- ☐ Chantilly, Arlington
- ☐ Chantilly, Annandale
- ☐ Sterling, McLean
- ☐ Sterling, Arlington
- ☐ Sterling, Annandale
- ☐ McLean, Arlington
- ☐ McLean, Annandale
- ☐ Arlington, Annandale

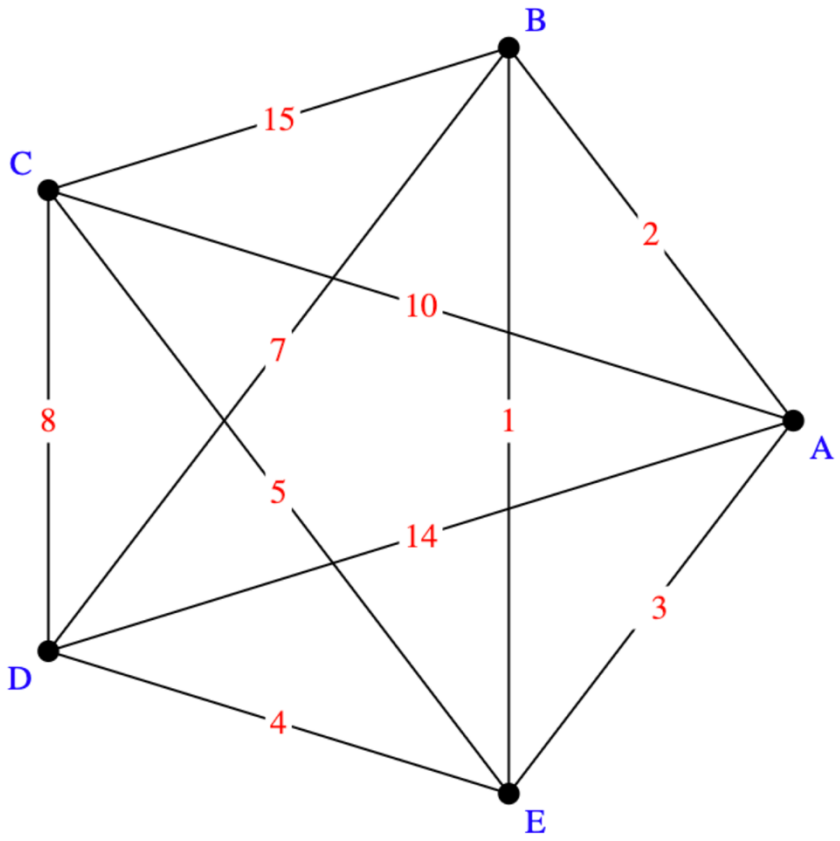
5) Use Kruskal's algorithm to find a minimum spanning tree for the graph below.



Select the edges that belong to this tree.

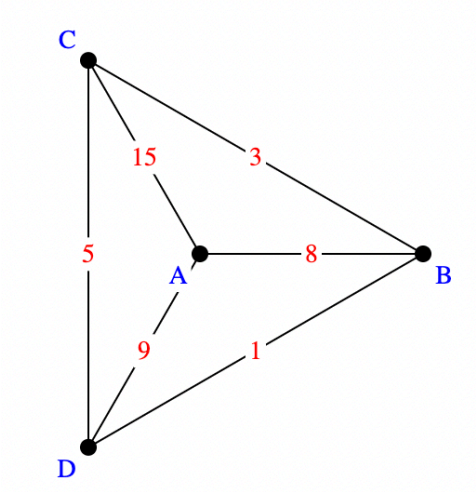
A) AB
B) AC
C) AE
D) BD
E) BE
F) CD
G) CE
H) DE

6) Use **Kruskal's Algorithm** to find a **Minimum Spanning Tree** for the graph below.



The weight of the minimum spanning tree is: _____

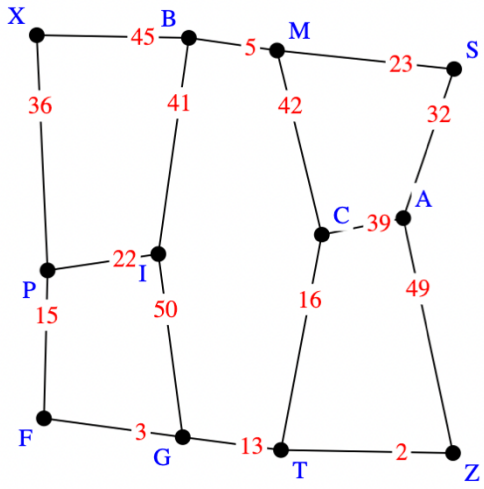
7)



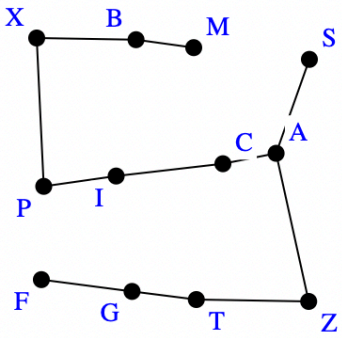
Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. Which of the edges below are included in the minimum cost tree?

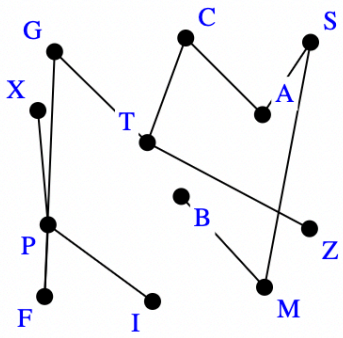
- ☐ BD
- ☐ BC
- ☐ AD
- ☐ CD
- ☐ AC
- ☐ AB

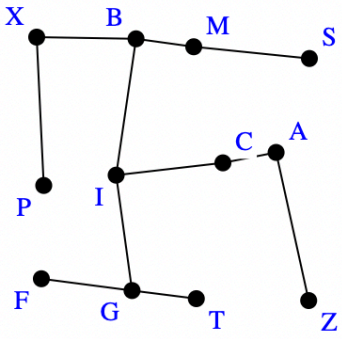
8) Consider the connected, weighted graph below.

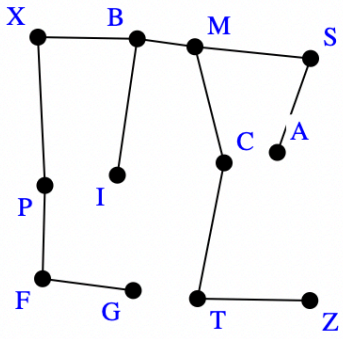


Use Kruskal's Algorithm to find the minimum spanning tree of the graph. Select the correct shape below.





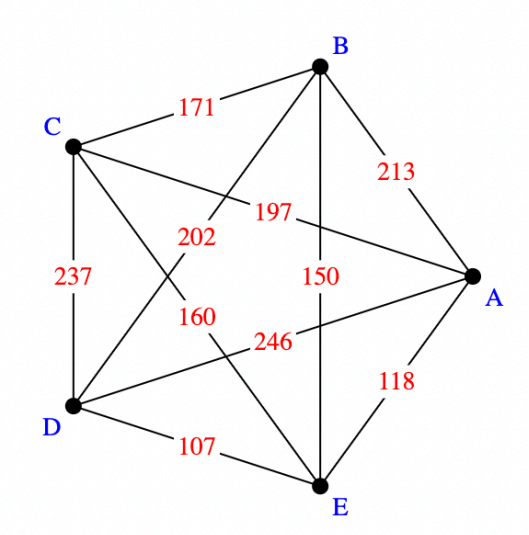




Determine the total weight of the minimum spanning tree.

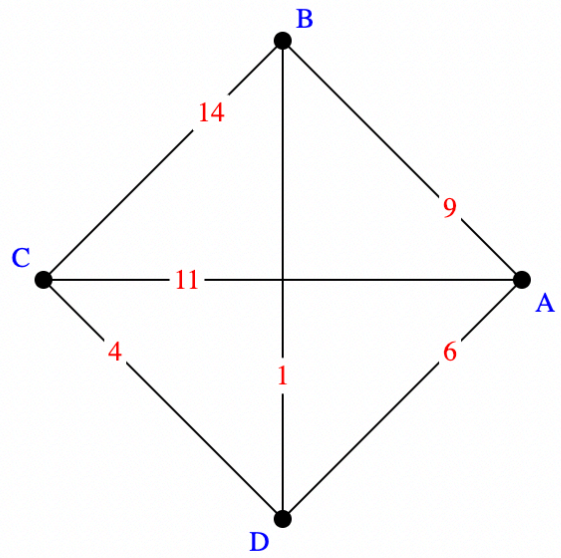
The total weight of the minimum spanning tree is [ ].

9) A city planner wants to work with community leaders to develop a system of walker-friendly paths between the city's five parks. Since cost is an issue, the planner collects data on the cost of each park-to-park route. The planner uses the Nearest Neighbor method to determine an estimate of a least cost system of sidewalks that would allow a walker to visit each park once and return to the park where the walker has parked his/her car. The weighted graph below shows the cost of each possible path in thousands of dollars.



Starting and ending at vertex ***A***, what is the minimum cost to construct paths as estimated by the Nearest-Neighbor method? $ _____.

10)



Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. What is the **total cost** of the tree?

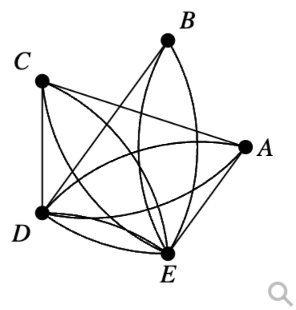
Sources

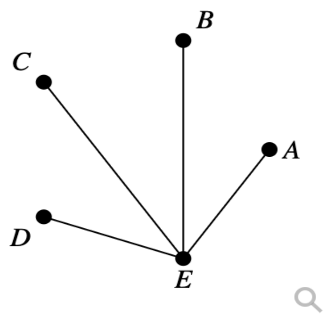
https://www.geeksforgeeks.org/graph-measurements-length-distance-diameter-eccentricity-radius-center/

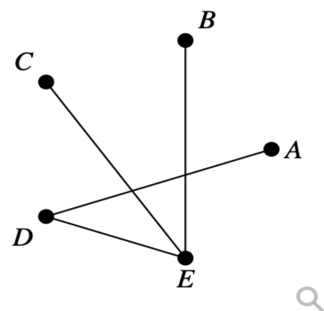
https://www.gatevidyalay.com/tag/cycle-graph-definition/

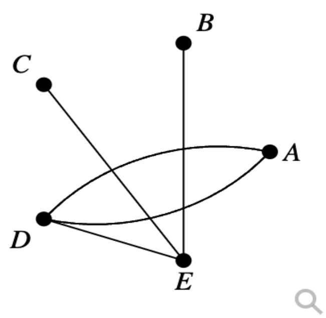
CHECK FOR UNDERSTANDING 2

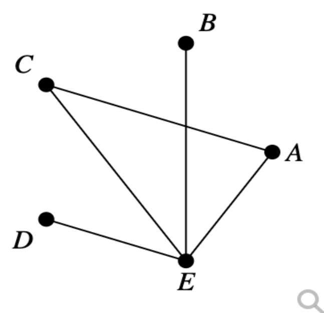
1) Select the graphs that are spanning trees for the given graph.

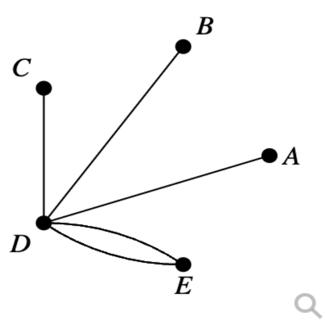


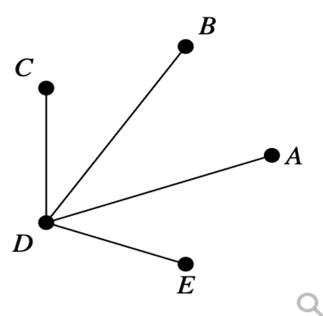
☐ 

☐ 

☐ 

☐ 

☐ 

☐ 

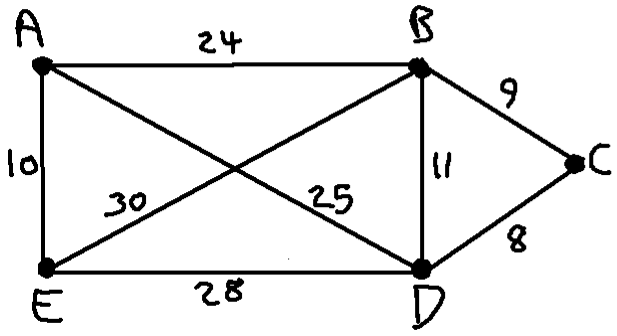
Minimum Spanning Tree

A minimum spanning tree is the tree that spans all of the vertices in a problem with the least cost (or time, or distance).

Key Idea. Spanning is **the amount of area or the amount of time that something encompasses**. An example of span is how long you live. An example of span is a house on three acres.

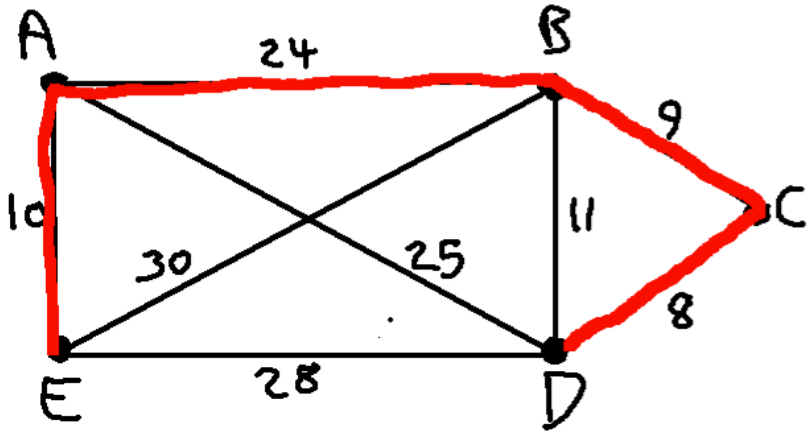
Additional Idea. In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

Example 6.2.4: Minimum Spanning Tree



The above is a weighted graph where the numbers on each edge represent the cost of each edge. We want to find the minimum spanning tree of this graph so that we can find a network that will reach all vertices for the least total cost.

Figure 6.2.6: Minimum Spanning Tree for Weighted Graph 1



This is the minimum spanning tree for the graph with a total cost of 51.

Minimum Spanning-Tree Algorithm

We shall learn about two most important spanning tree algorithms here –

- Kruskal's Algorithm
- Prim's Algorithm

Kruskal's Algorithm

Videos: https://www.youtube.com/watch?v=JZBQLXgSGfs

https://www.youtube.com/watch?v=Yo7sddEVONg

https://www.youtube.com/watch?v=d4BEgzK08JE&t=184s

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

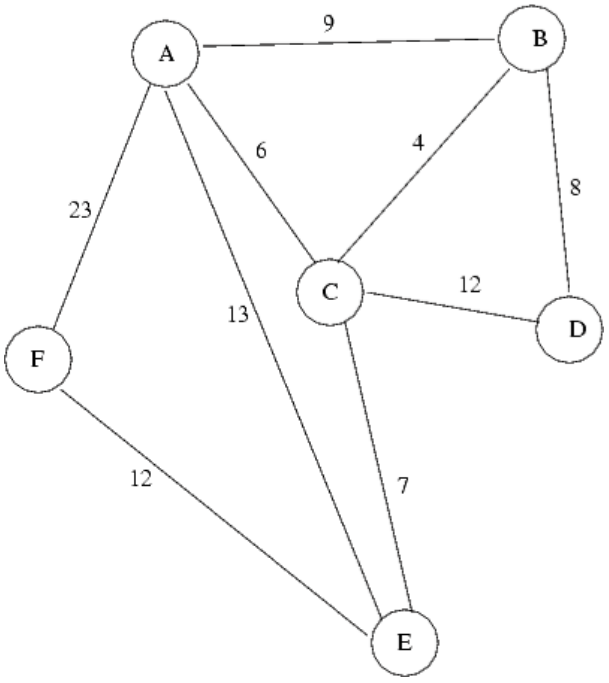
Key Idea. A procedure that produces the minimum spanning tree of a graph is called *Kruskal's* Algorithm. The algorithm specifies that one is always to choose an edge with the least possible *weight* while avoiding the creation of any *circuits*.

Kruskal's Algorithm: Since some graphs are much more complicated than the previous example, we can use Kruskal's Algorithm to always be able to find the minimum spanning tree for any graph.

5. Find the cheapest link in the graph. If there is more than one, pick one at random. Mark it in red.
6. Find the next cheapest link in the graph. If there is more than one, pick one at random. Mark it in red.
7. Continue doing this as long as the next cheapest link does not create a red circuit.
8. You are done when the red edges span every vertex of the graph without any circuits. The red edges are the MST (minimum spanning tree).

Example 6.2.5: Using Kruskal's Algorithm Figure

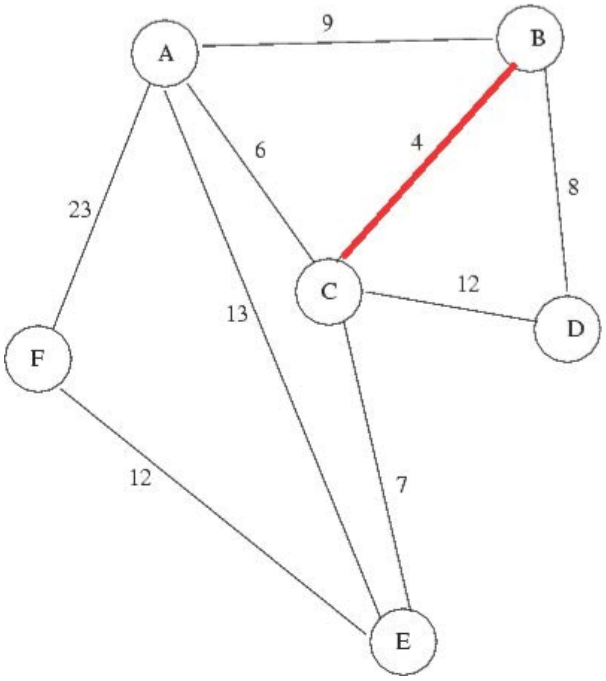
6.2.7: Weighted Graph 2



Suppose that it is desired to install a new fiber optic cable network between the six cities (A, B, C, D, E, and F) shown above for the least total cost. Also, suppose that the fiber optic cable can only be installed along the roadways shown above. The weighted graph above shows the cost (in millions of dollars) of installing the fiber optic cable along each roadway. We want to find the minimum spanning tree for this graph using Kruskal's Algorithm.

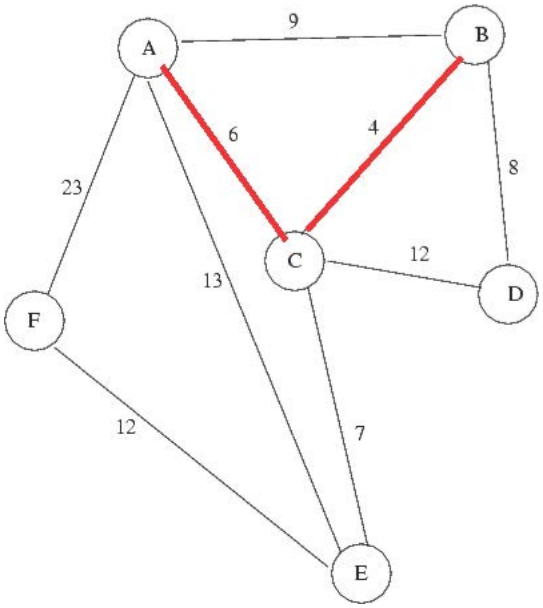
Step 1: Find the cheapest link of the whole graph and mark it in red. The cheapest link is between B and C with a cost of four million dollars.

Figure 6.2.8: Kruskal's Algorithm Step 1



Step 2: Find the next cheapest link of the whole graph and mark it in red. The next cheapest link is between A and C with a cost of six million dollars.

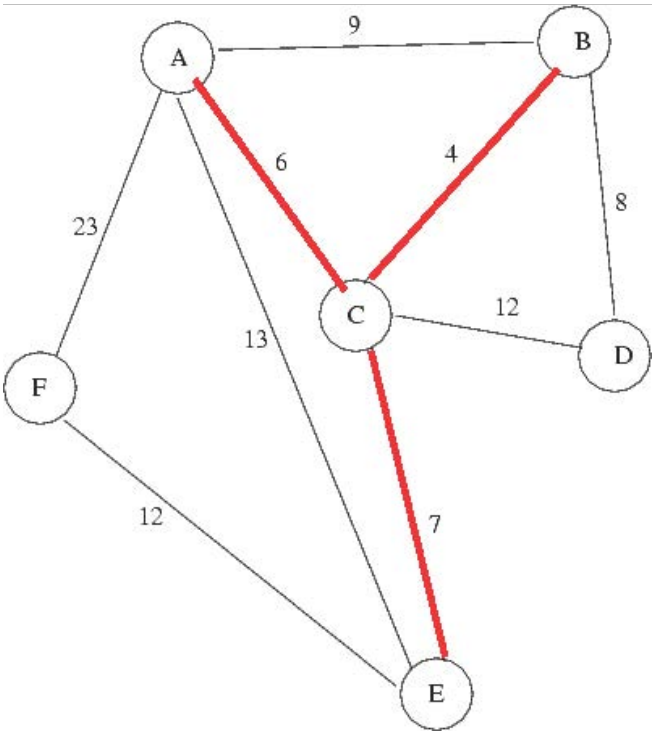
Figure 6.2.9: Kruskal's Algorithm Step 2



Step 3: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit.

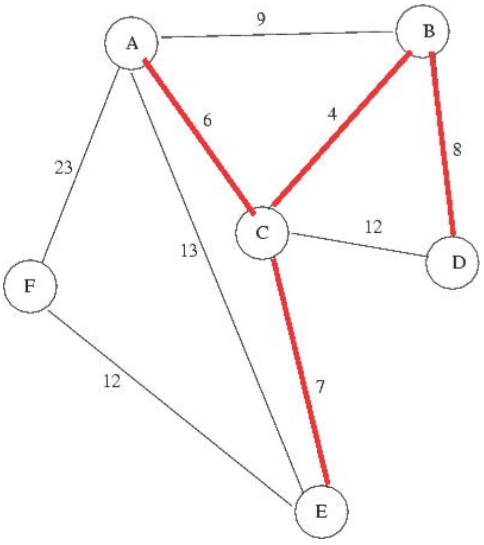
The next cheapest link is between C and E with a cost of seven million dollars.

Figure 6.2.10: Kruskal's Algorithm Step 3



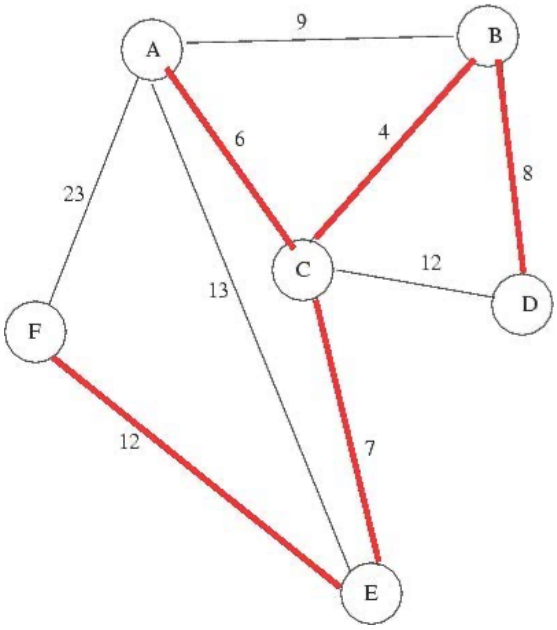
Step 4: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit. The next cheapest link is between B and D with a cost of eight million dollars.

Figure 6.2.11: Kruskal's Algorithm Step 4



Step 5: Find the next cheapest link of the whole graph and mark it in red as long as it does not create a red circuit. The next cheapest link is between A and B with a cost of nine million dollars, but that would create a red circuit so we cannot use it. Therefore, the next cheapest link after that is between E and F with a cost of 12 million dollars, which we are able to use. We cannot use the link between C and D which also has a cost of 12 million dollars because it would create a red circuit.

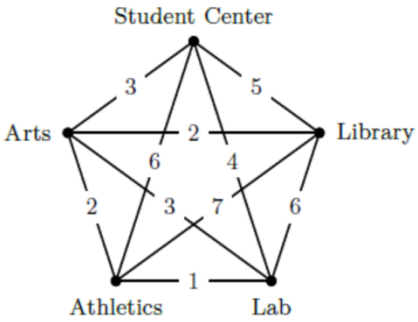
Figure 6.2.12: Kruskal's Algorithm Step 5



This was the last step, and we now have the minimum spanning tree for the weighted graph with a total cost of $37,000,000.

CHECK FOR UNDERSTANDING 3

1) A maintenance team is responsible for a group of five buildings on campus. These buildings are shown in the graph below, with the distance given between each pair of buildings. After a blizzard, the team is tasked with clearing the snow, but there is not enough time to clear all the walkways.

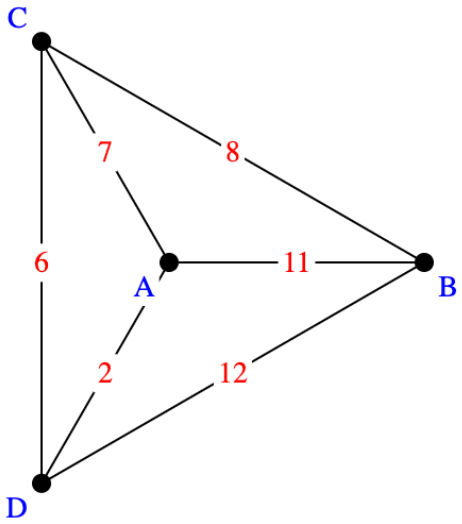


Which walkways should the maintenance team plow in order to connect all the buildings, while minimizing the time needed to do so (really, by minimizing the distance)?

- ☐ Student Center, Library
- ☐ Student Center, Lab
- ☐ Student Center, Athletics
- ☐ Student Center, Arts
- ☐ Library, Lab
- ☐ Library, Athletics
- ☐ Library, Arts
- ☐ Lab, Athletics
- ☐ Lab, Arts
- ☐ Athletics, Arts

What is the total length of walkway that will be plowed?

2) Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. Which of the edges below are included in the minimum cost tree?



A) CD

B) AB

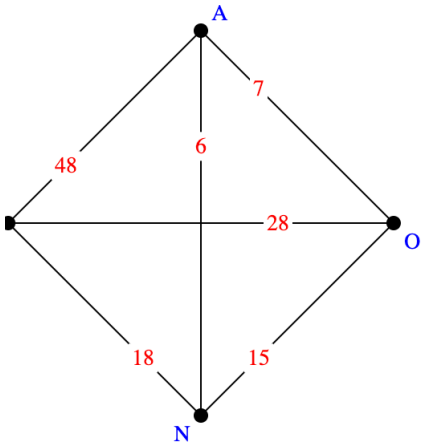
C) AD

D) BD

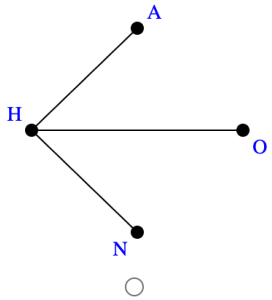
E) AC

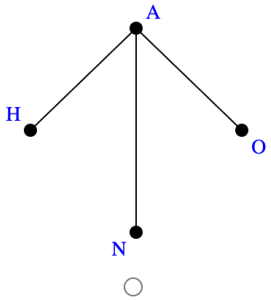
F) BC

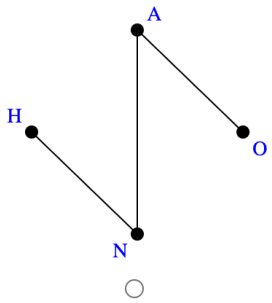
3) Consider the connected, weighted graph below.

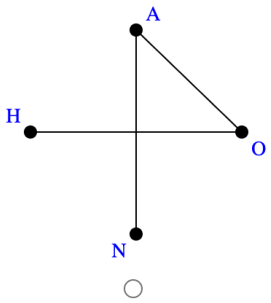


Use Kruskal's Algorithm to find the minimum spanning tree of the graph. Select the correct shape below.









Determine the total weight of the minimum spanning tree.

The total weight of the minimum spanning tree is _____.

4) A power company needs to lay updated distribution lines connecting eight cities in Virginia to the power grid. The distances between these cities are given in the table below. Design a network that will minimize the amount of new line.

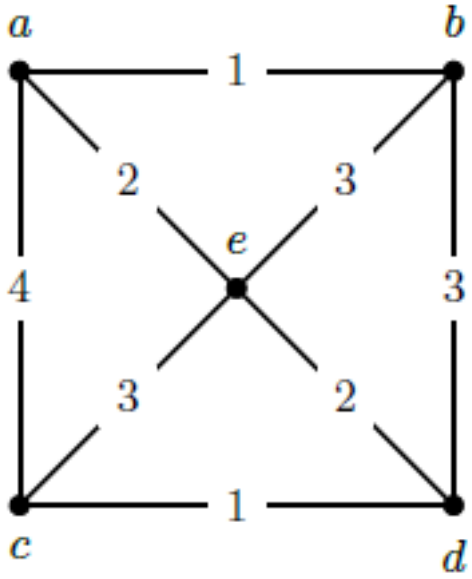
	Purcellville	Leesburg	Middleburg	Chantilly	Sterling	McLean	Arlington	Annandale
Purcellville	--	8	11	23	19	32	37	35
Leesburg	8	--	14	17	10	24	29	27
Middleburg	11	14	--	18	16	30	34	31
Chantilly	23	17	18	--	8	13	18	13
Sterling	19	10	16	8	--	15	20	17
McLean	32	24	30	13	15	--	5	7
Arlington	37	29	34	18	20	5	--	6
Annandale	35	27	31	13	17	7	6	--

What connections should be built between cities and

What is the total required length of line that must be laid?

- ☐ Purcellville, Leesburg
- ☐ Purcellville, Middleburg
- ☐ Purcellville, Chantilly
- ☐ Purcellville, Sterling
- ☐ Purcellville, McLean
- ☐ Purcellville, Arlington
- ☐ Purcellville, Annandale
- ☐ Leesburg, Middleburg
- ☐ Leesburg, Chantilly
- ☐ Leesburg, Sterling
- ☐ Leesburg, McLean
- ☐ Leesburg, Arlington
- ☐ Leesburg, Annandale
- ☐ Middleburg, Chantilly
- ☐ Middleburg, Sterling
- ☐ Middleburg, McLean
- ☐ Middleburg, Arlington
- ☐ Middleburg, Annandale
- ☐ Chantilly, Sterling
- ☐ Chantilly, McLean
- ☐ Chantilly, Arlington
- ☐ Chantilly, Annandale
- ☐ Sterling, McLean
- ☐ Sterling, Arlington
- ☐ Sterling, Annandale
- ☐ McLean, Arlington
- ☐ McLean, Annandale
- ☐ Arlington, Annandale

5) Use Kruskal's algorithm to find a minimum spanning tree for the graph below.



Select the edges that belong to this tree.

I) AB
J) AC
K) AE
L) BD
M) BE
N) CD
O) CE
P) DE

Prim's Algorithm

Video: https://www.youtube.com/watch?v=Uj47dxYPow8

https://www.youtube.com/watch?v=jsmMtJpPnhU&list=PLDV1Zeh2NRsDGO4--qE8yH72HFL1Km93P&index=30

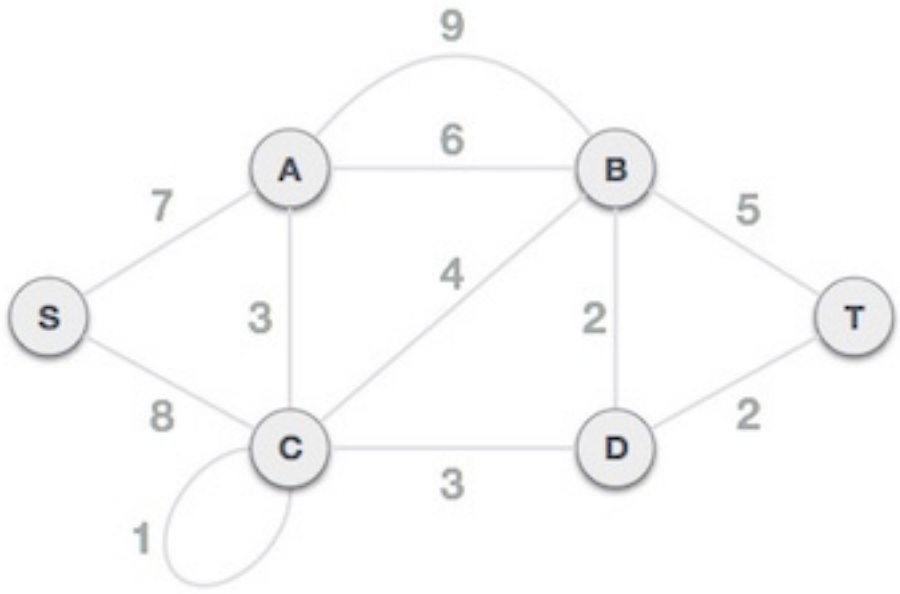
https://www.youtube.com/watch?v=5M7bOXrn54A

Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

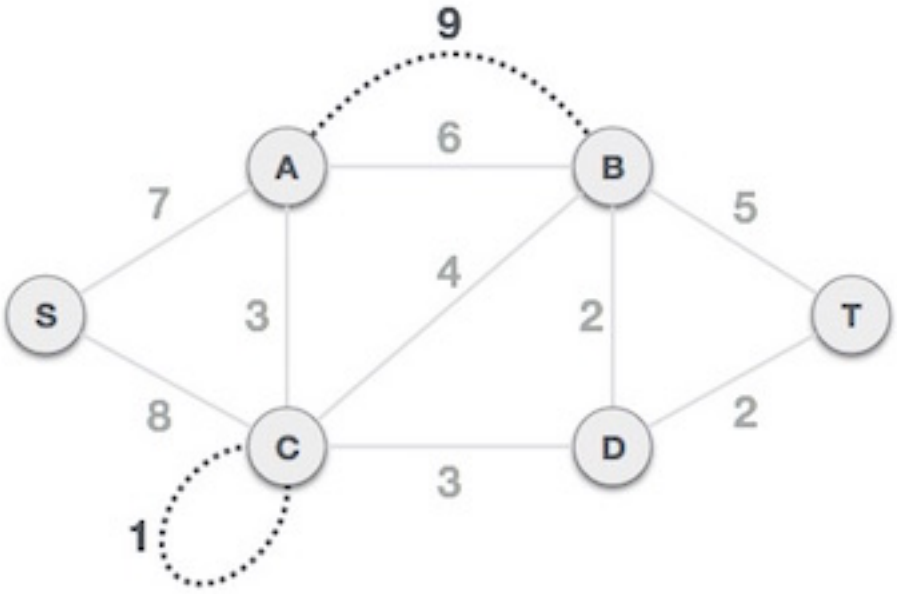
Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

Key Idea. In Prim's algorithm, always select the minimum edge and should be connected and there should be no loop, circuit or cyle.

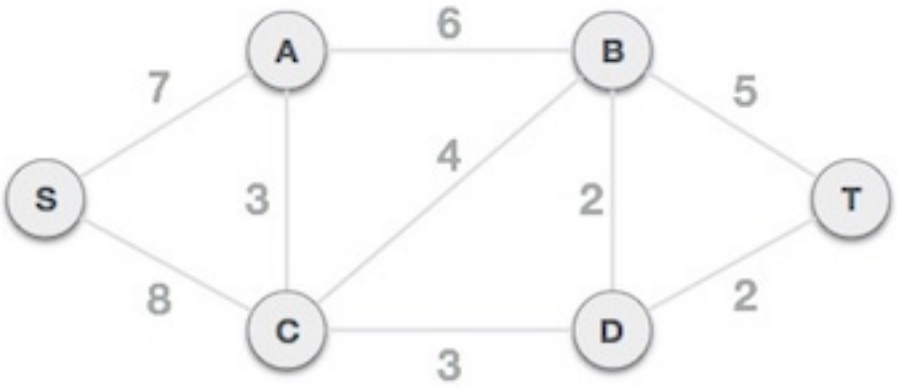
Example:



Step 1 - Remove all loops and parallel edges



Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.



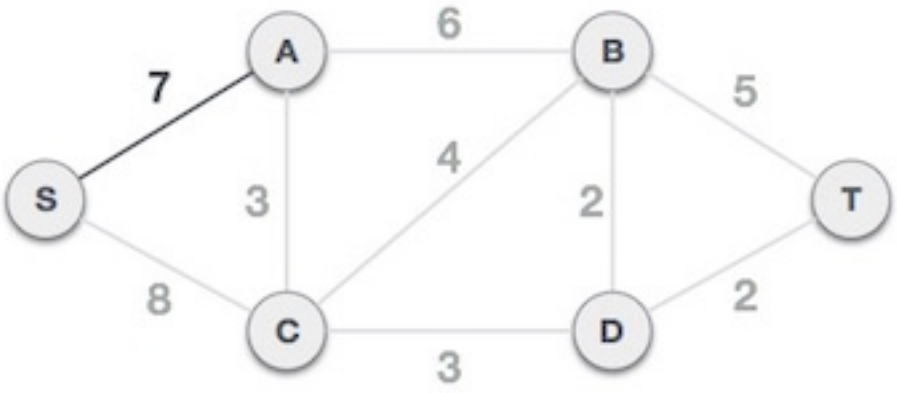
Step 2 - Choose any arbitrary node as root node

In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning

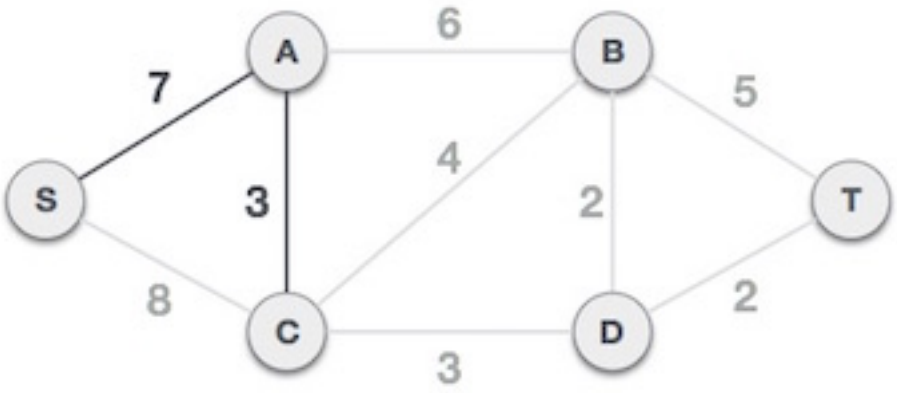
tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

Step 3 - Check outgoing edges and select the one with less cost

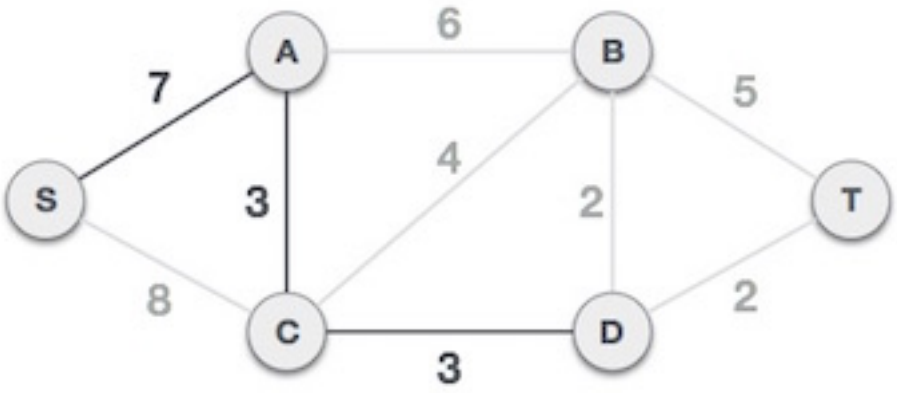
After choosing the root node **S**, we see that S, A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S, A as it is lesser than the other.



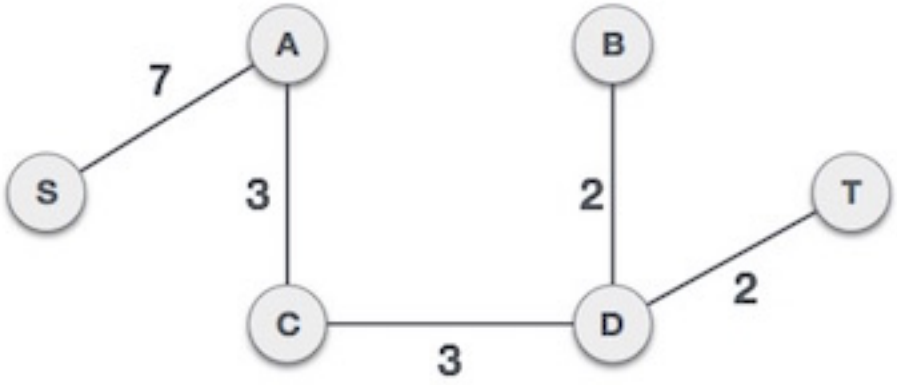
Now, the tree S-7-A is treated as one node, and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



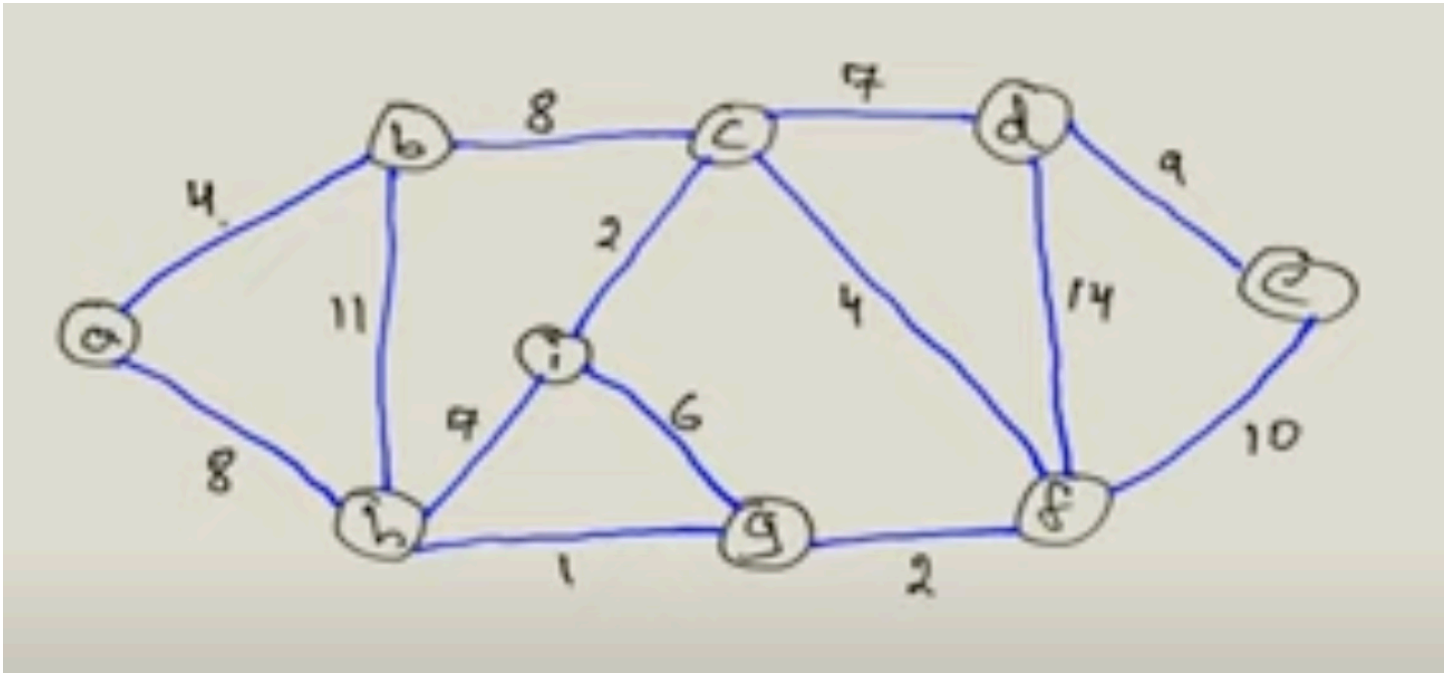
After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.



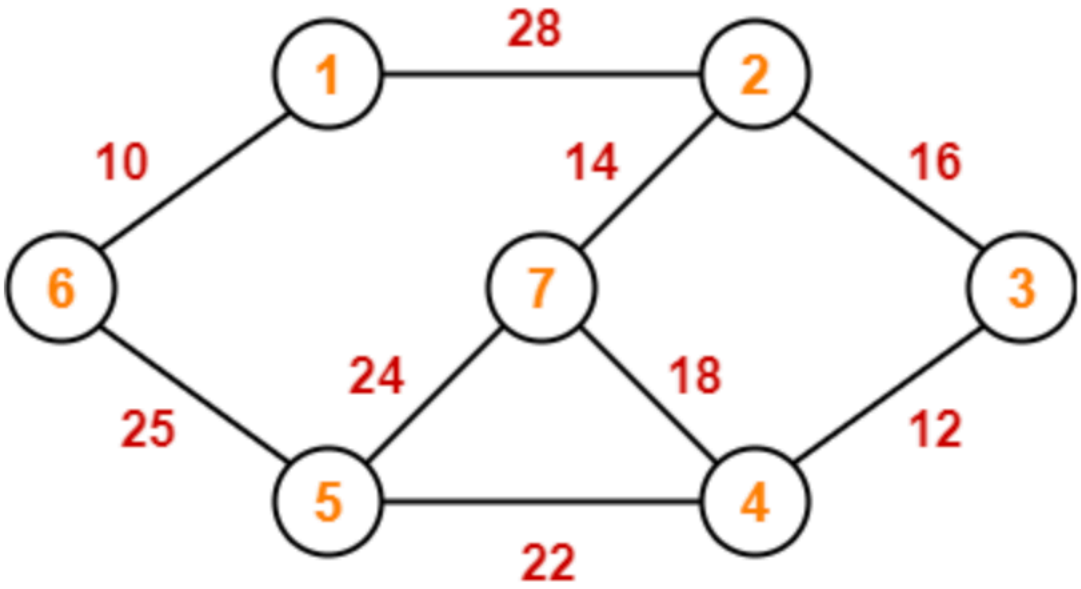
After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.



WE DO 1) Find the minimum spanning cost using Prim's Algorithm

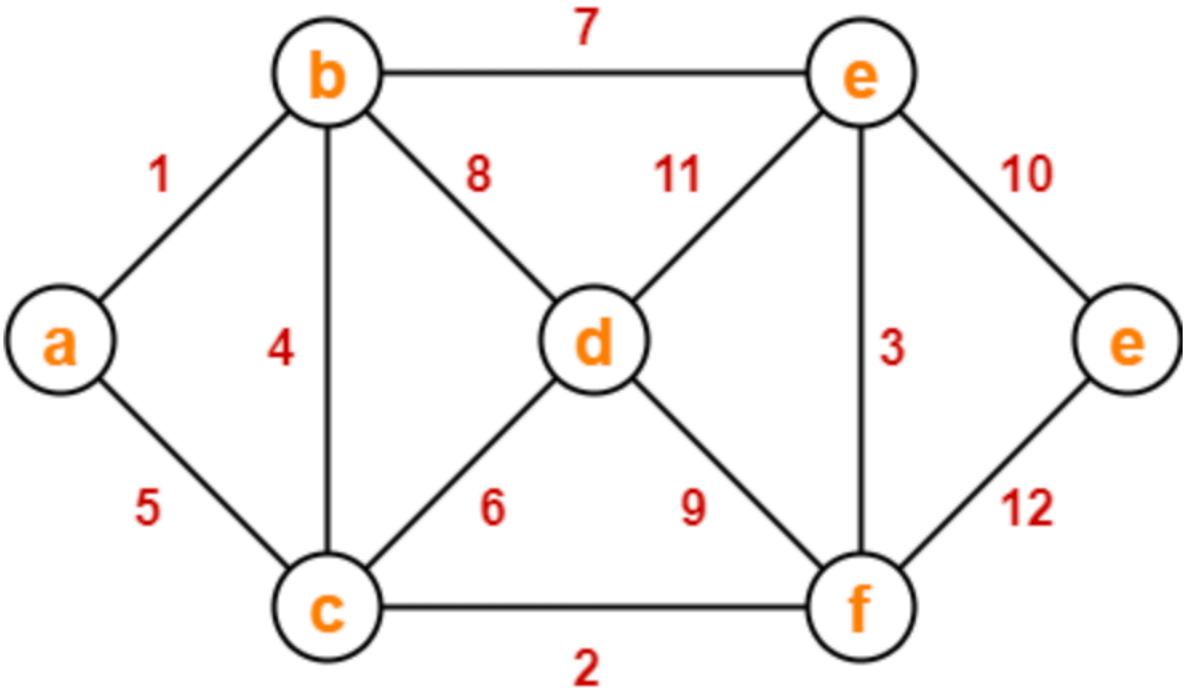


YOU DO 1) Find the minimum spanning cost using Prim's Algorithm



EXTRA PROBLEM

Using Prim's Algorithm, find the cost of minimum spanning tree (MST) of the given graph-



Example Reference (Prim's Algorithm):

https://www.gatevidyalay.com/prims-algorithm-prim-algorithm-example/

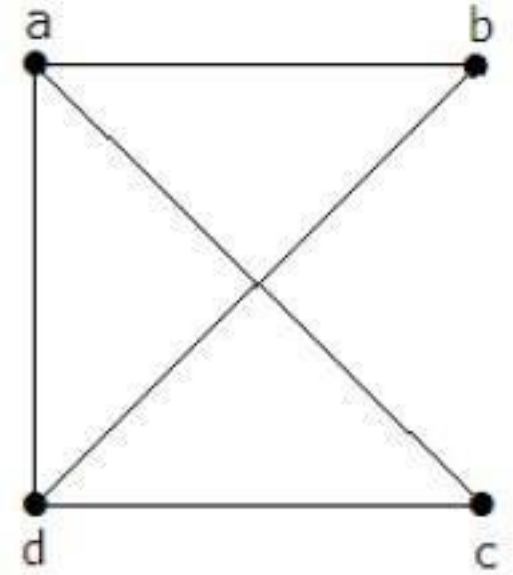
Reference:

https://www.tutorialspoint.com/data_structures_algorithms/spanning_tree.htm

Kirchhoff's Theorem

Kirchhoff's theorem is useful in finding the number of spanning trees that can be formed from a connected graph.

Example



The matrix 'A' be filled as, if there is an edge between two vertices, then it should be given as '1', else '0'.

Videos:

https://www.youtube.com/watch?v=b233VKD6udo

Kruskal's Algorithm
https://www.youtube.com/watch?v=d4BEgzK08JE

Quick Lecture (GREEDY METHOD)

The **greedy method** is one of the strategies like ***Divide and conquer*** used to solve the problems. This method is used for solving optimization problems. An optimization problem is a problem that demands either maximum or minimum results. Let's understand through some terms.

The Greedy method is the simplest and straightforward approach. It is not an algorithm, but it is a technique. The main function of this approach is that the decision is taken on the basis of the currently available information. Whatever the current information is present, the decision is made without worrying about the effect of the current decision in future.

This technique is basically used to determine the feasible solution that may or may not be optimal. The feasible solution is a subset that satisfies the given criteria. The optimal solution is the solution which is the best and the most favorable solution in the subset. In the case of feasible, if more than one solution satisfies the given criteria then those solutions will be considered as the feasible, whereas the optimal solution is the best solution among all the solutions.

Characteristics of Greedy method

The following are the characteristics of a greedy method:

- To construct the solution in an optimal way, this algorithm creates two sets where one set contains all the chosen items, and another set contains the rejected items.
- A Greedy algorithm makes good local choices in the hope that the solution should be either feasible or optimal.

Components of Greedy Algorithm

The components that can be used in the greedy algorithm are:

- **Candidate set:** A solution that is created from the set is known as a candidate set.
- **Selection function:** This function is used to choose the candidate or subset which can be added in the solution.
- **Feasibility function:** A function that is used to determine whether the candidate or subset can be used to contribute to the solution or not.
- **Objective function:** A function is used to assign the value to the solution or the partial solution.
- **Solution function:** This function is used to intimate whether the complete function has been reached or not.

Applications of Greedy Algorithm

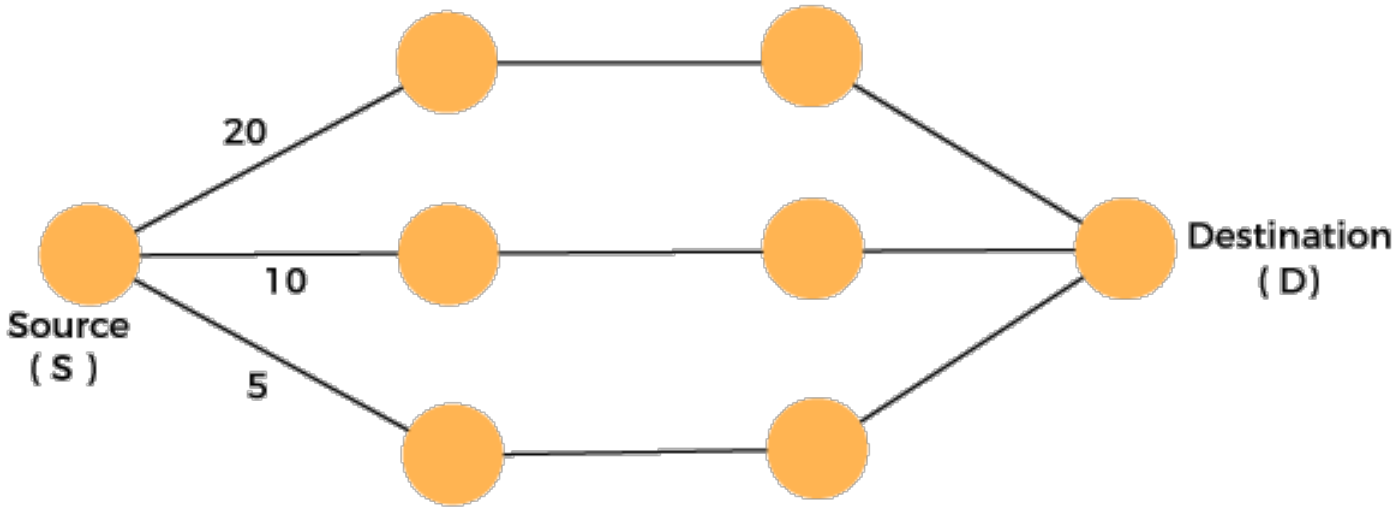
- It is used in finding the shortest path.
- It is used to find the minimum spanning tree using the prim's algorithm or the Kruskal's algorithm.
- It is used in a job sequencing with a deadline.
- This algorithm is also used to solve the fractional knapsack problem.

Disadvantages of using Greedy algorithm

Greedy algorithm makes decisions based on the information available at each phase without considering the broader problem. So, there might be a possibility that the greedy solution does not give the best solution for every problem.

It follows the local optimum choice at each stage with a intend of finding the global optimum. Let's understand through an example.

Consider the graph which is given below:

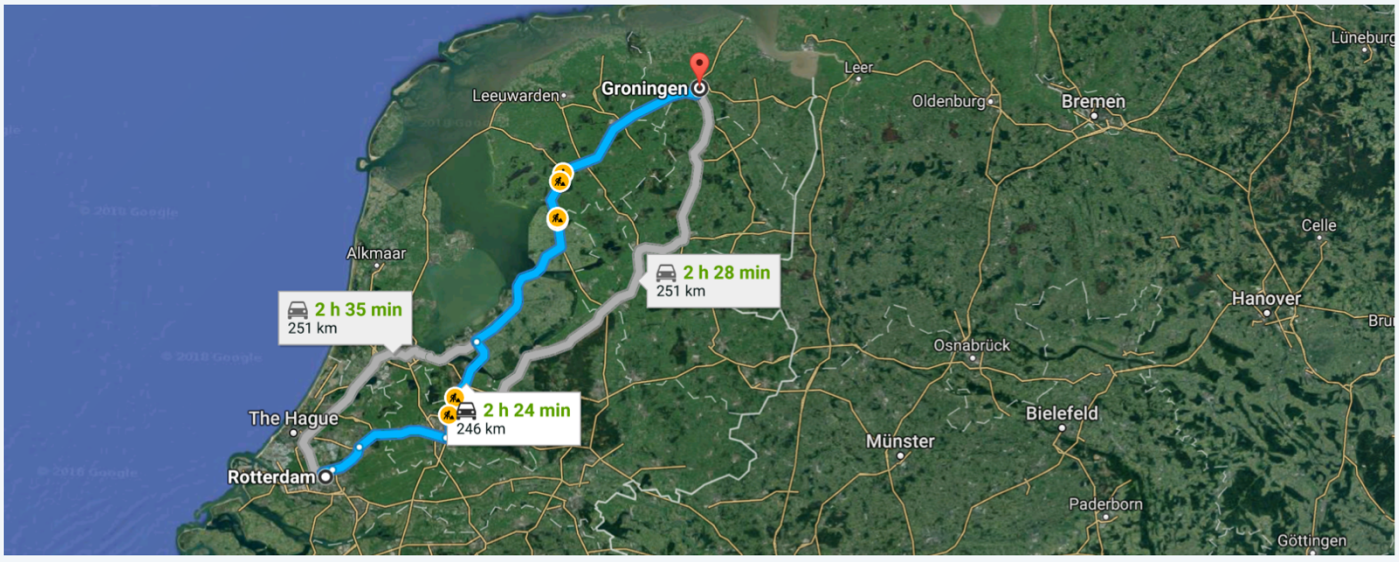


We must travel from the source to the destination at the minimum cost. Since we have three feasible solutions having cost paths as 10, 20, and 5. 5 is the minimum cost path so it is the optimal solution. This is the local optimum, and in this way, we find the local optimum at each stage to calculate the global optimal solution.

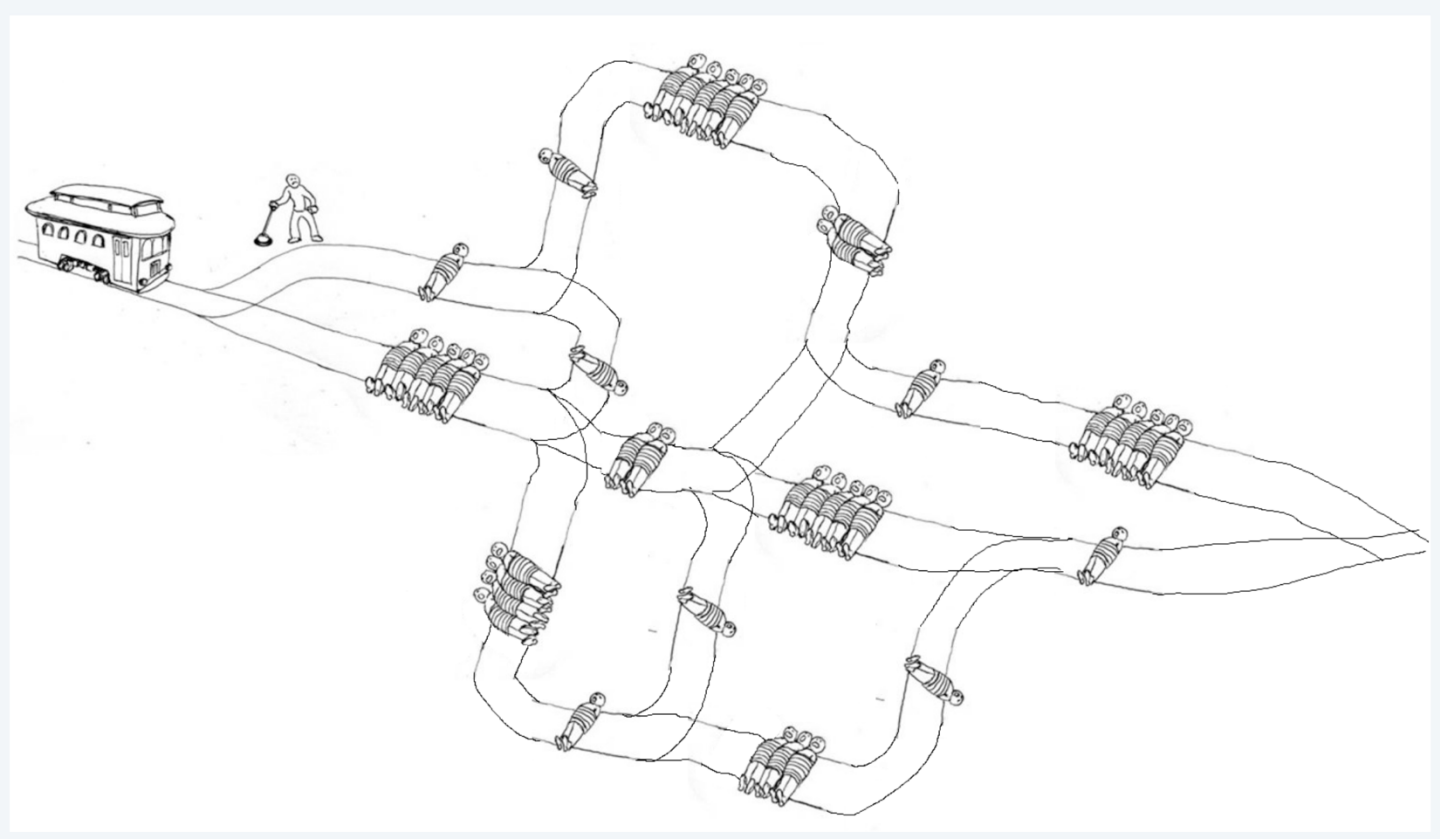
Dijkstra's Minimal Spanning Tree Algorithm
(https://www.youtube.com/watch?v=XB4MIexjvY0)
https://www.youtube.com/watch?v=KvRwplnIoEM
https://www.youtube.com/watch?v=pVfj6mxhdMw

Edsger Dijkstra

> *" What's the shortest way to travel from Rotterdam to Groningen? It is the algorithm for the shortest path, which I designed in about 20 minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path."* — Edsger Dijsktra



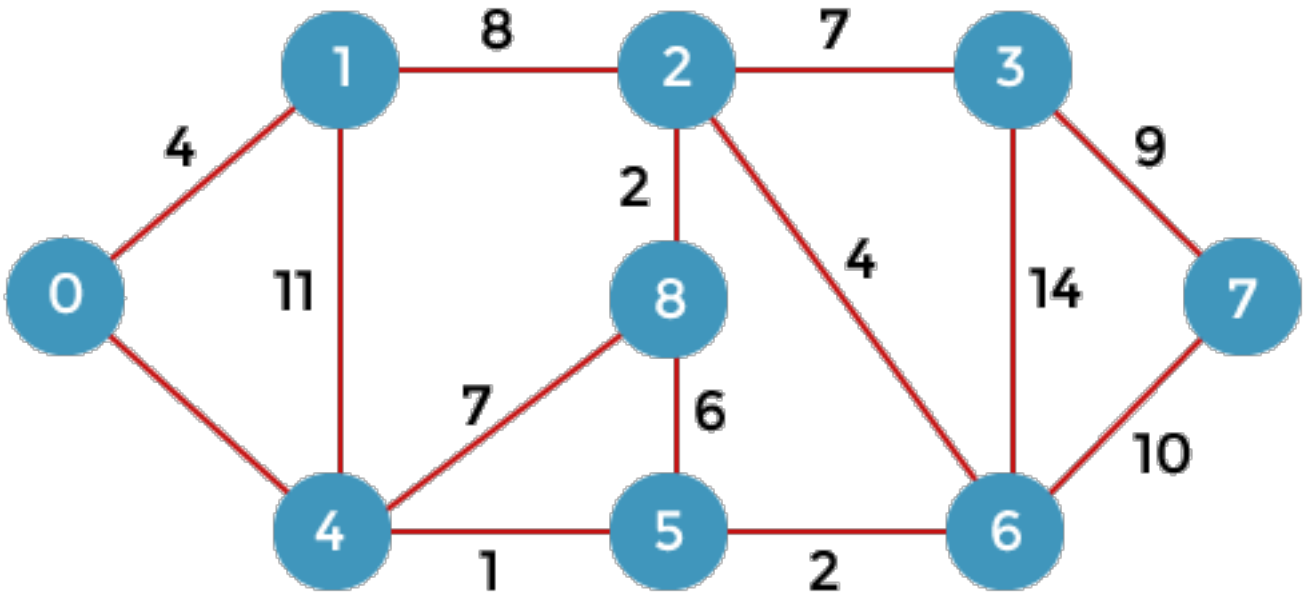
The moral implications of implementing shortest-path algorithms



Dijkstra algorithm is a single-source shortest path algorithm. Here, single source means that only one source is given, and we have to find the shortest path from the source to all the nodes.

Key Idea: Dijkstra Algorithm is a minimization and optimization problem.

Let's understand the working of Dijkstra's algorithm. Consider the below graph.



First, we have to consider any vertex as a source vertex. Suppose we consider vertex 0 as a source vertex.

Here we assume that 0 as a source vertex, and distance to all the other vertices is infinity. Initially, we do not know the distances. First, we will find out the vertices which are directly connected to the vertex 0. As we can observe in the above graph that two vertices are directly connected to vertex 0.

Let's assume that the vertex 0 is represented by 'x' and the vertex 1 is represented by 'y'. The distance between the vertices can be calculated by using the below formula:

d(x, y) = d(x) + c(x, y) < d(y)

= **(0 + 4)** $< \infty$

= **4** $< \infty$

Since 4<∞ so we will update d(v) from ∞ to 4.

Therefore, we come to the conclusion that the formula for calculating the distance between the vertices:

{if(d(u) + c(u, v) < d(v))

d(v) = d(u) +c(u, v) }

Now we consider vertex 0 same as 'x' and vertex 4 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= **(0 + 8)** < ∞

= **8** < ∞

Therefore, the value of d(y) is 8. We replace the infinity value of vertices 1 and 4 with the values 4 and 8 respectively. Now, we have found the shortest path from the vertex 0 to 1 and 0 to 4. Therefore, vertex 0 is selected. Now, we will compare all the vertices except the vertex 0. Since vertex 1 has the lowest value, i.e., 4; therefore, vertex 1 is selected.

Since vertex 1 is selected, so we consider the path from 1 to 2, and 1 to 4. We will not consider the path from 1 to 0 as the vertex 0 is already selected.

First, we calculate the distance between the vertex 1 and 2. Consider the vertex 1 as 'x', and the vertex 2 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= **(4 + 8)** < ∞

= **12** < ∞

Since 12<∞ so we will update d(2) from ∞ to 12.

Now, we calculate the distance between the vertex 1 and vertex 4. Consider the vertex 1 as 'x' and the vertex 4 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= (4 + 11) < 8

= 15 < 8

Since 15 is not less than 8, we will not update the value d(4) from 8 to 12.

Till now, two nodes have been selected, i.e., 0 and 1. Now we have to compare the nodes except the node 0 and 1. The node 4 has the minimum distance, i.e., 8. Therefore, vertex 4 is selected.

Since vertex 4 is selected, so we will consider all the direct paths from the vertex 4. The direct paths from vertex 4 are 4 to 0, 4 to 1, 4 to 8, and 4 to 5. Since the vertices 0 and 1 have already been selected so we will not consider the vertices 0 and 1. We will consider only two vertices, i.e., 8 and 5.

First, we consider the vertex 8. First, we calculate the distance between the vertex 4 and 8. Consider the vertex 4 as 'x', and the vertex 8 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= (8 + 7) $< \infty$

= **15** $< \infty$

Since 15 is less than the infinity so we update d(8) from infinity to 15.

Now, we consider the vertex 5. First, we calculate the distance between the vertex 4 and 5. Consider the vertex 4 as 'x', and the vertex 5 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

$= \mathbf{(8 + 1)} < \infty$

$= \mathbf{9} < \infty$

Since 5 is less than the infinity, we update d(5) from infinity to 9.

Till now, three nodes have been selected, i.e., 0, 1, and 4. Now we have to compare the nodes except the nodes 0, 1 and 4. The node 5 has the minimum value, i.e., 9. Therefore, vertex 5 is selected.

Since the vertex 5 is selected, so we will consider all the direct paths from vertex 5. The direct paths from vertex 5 are 5 to 8, and 5 to 6.

First, we consider the vertex 8. First, we calculate the distance between the vertex 5 and 8. Consider the vertex 5 as 'x', and the vertex 8 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= (9 + 15) < 15

= 24 < 15

Since 24 is not less than 15 so we will not update the value d(8) from 15 to 24.

Now, we consider the vertex 6. First, we calculate the distance between the vertex 5 and 6. Consider the vertex 5 as 'x', and the vertex 6 as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

$= \mathbf{(9 + 2)} < \infty$

$= \mathbf{11} < \infty$

Since 11 is less than infinity, we update d(6) from infinity to 11.

Till now, nodes 0, 1, 4 and 5 have been selected. We will compare the nodes except the selected nodes. The node 6 has the lowest value as compared to other nodes. Therefore, vertex 6 is selected.

Since vertex 6 is selected, we consider all the direct paths from vertex 6. The direct paths from vertex 6 are 6 to 2, 6 to 3, and 6 to 7.

First, we consider the vertex 2. Consider the vertex 6 as 'x', and the vertex 2 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (11 + 4) < 12$

$= 15 < 12$

Since 15 is not less than 12, we will not update d(2) from 12 to 15

Now we consider the vertex 3. Consider the vertex 6 as 'x', and the vertex 3 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (11 + 14) < \infty$

$= 25 < \infty$

Since 25 is less than ∞, so we will update d(3) from ∞ to 25.

Now we consider the vertex 7. Consider the vertex 6 as 'x', and the vertex 7 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (11 + 10) < \infty$

$= 22 < \infty$

Since 22 is less than ∞ so, we will update d(7) from ∞ to 22.

Till now, nodes 0, 1, 4, 5, and 6 have been selected. Now we have to compare all the unvisited nodes, i.e., 2, 3, 7, and 8. Since node 2 has the minimum value, i.e., 12 among all the other unvisited nodes. Therefore, node 2 is selected.

Since node 2 is selected, so we consider all the direct paths from node 2. The direct paths from node 2 are 2 to 8, 2 to 6, and 2 to 3.

First, we consider the vertex 8. Consider the vertex 2 as 'x' and 8 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (12 + 2) < 15$

$= 14 < 15$

Since 14 is less than 15, we will update d(8) from 15 to 14.

Now, we consider the vertex 6. Consider the vertex 2 as 'x' and 6 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (12 + 4) < 11$

$= 16 < 11$

Since 16 is not less than 11 so we will not update d(6) from 11 to 16.

Now, we consider the vertex 3. Consider the vertex 2 as 'x' and 3 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (12 + 7) < 25$

$= 19 < 25$

Since 19 is less than 25, we will update d(3) from 25 to 19.

Till now, nodes 0, 1, 2, 4, 5, and 6 have been selected. We compare all the unvisited nodes, i.e., 3, 7, and 8. Among nodes 3, 7, and 8, node 8 has the minimum value. The nodes which are directly connected to node 8 are 2, 4, and 5. Since all the directly connected nodes are selected so we will not consider any node for the updating.

The unvisited nodes are 3 and 7. Among the nodes 3 and 7, node 3 has the minimum value, i.e., 19. Therefore, the node 3 is selected. The nodes which are directly connected to the node 3 are 2, 6, and 7. Since the nodes 2 and 6 have been selected so we will consider these two nodes.

Now, we consider the vertex 7. Consider the vertex 3 as 'x' and 7 as 'y'.

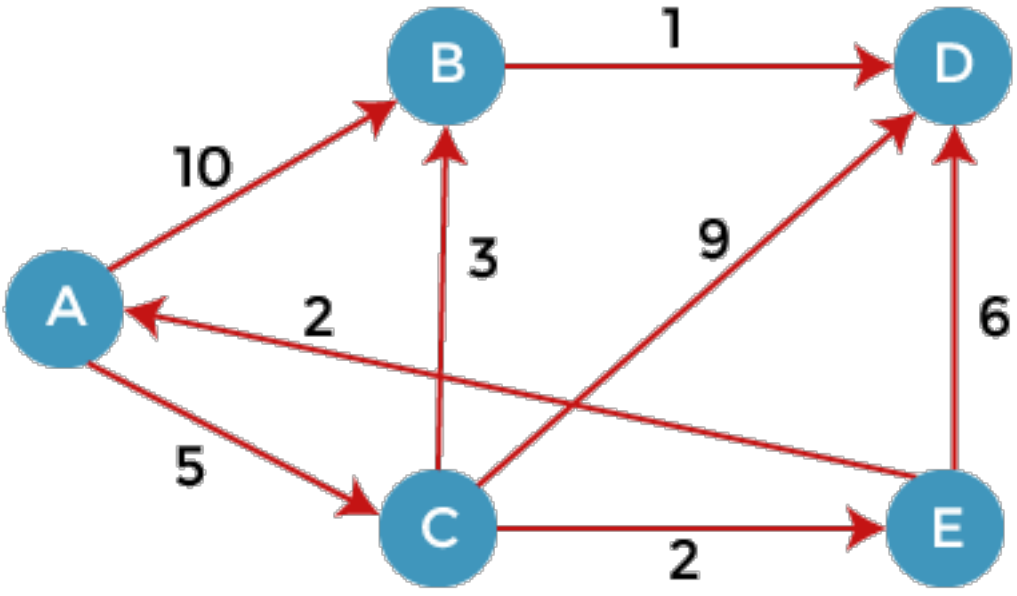
$d(x, y) = d(x) + c(x, y) < d(y)$

$= (19 + 9) < 21$

$= 28 < 21$

Since 28 is not less than 21, so we will not update d(7) from 28 to 21.

Let's consider the directed graph.



Here, we consider A as a source vertex. A vertex is a source vertex so entry is filled with 0 while other vertices filled with ∞. The distance from source vertex to source vertex is 0, and the distance from the source vertex to other vertices is ∞.

We will solve this problem using the below table:

A	B	C	D	E
∞	∞	∞	∞	∞

Since 0 is the minimum value in the above table, so we select vertex A and added in the second row shown as below:

	A	B	C	D	E
A	0	∞	∞	∞	∞

As we can observe in the above graph that there are two vertices directly connected to the vertex A, i.e., B and C. The vertex A is not directly connected to the vertex E, i.e., the edge is from E to A. Here we can calculate the two distances, i.e., from A to B and A to C. The same formula will be used as in the previous problem.

```
If(d(x) + c(x, y)  < d(y))
  Then we update d(y) = d(x) + c(x, y)
```

	A	B	C	D	E
A	0	∞	∞	∞	∞
		10	5	∞	∞

As we can observe in the third row that 5 is the lowest value so vertex C will be added in the third row.

We have calculated the distance of vertices B and C from A. Now we will compare the vertices to find the vertex with the lowest value. Since the vertex C has the minimum value, i.e., 5 so vertex C will be selected.

Since the vertex C is selected, so we consider all the direct paths from the vertex C. The direct paths from the vertex C are C to B, C to D, and C to E.

First, we consider the vertex B. We calculate the distance from C to B. Consider vertex C as 'x' and vertex B as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (5 + 3) < \infty$

$= 8 < \infty$

Since 8 is less than the infinity so we update d(B) from ∞ to 8. Now the new row will be inserted in which value 8 will be added under the B column.

	A	B	C	D	E
A	0	∞	∞	∞	∞
		10	5	∞	∞
		8			

We consider the vertex D. We calculate the distance from C to D. Consider vertex C as 'x' and vertex D as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= (5 + 9) < ∞

= 14 < ∞

Since 14 is less than the infinity so we update d(D) from ∞ to 14. The value 14 will be added under the D column.

	A	B	C	D	E
A	0	∞	∞	∞	∞
C		10	5	∞	∞
		8		14	

We consider the vertex E. We calculate the distance from C to E. Consider vertex C as 'x' and vertex E as 'y'.

d(x, y) = d(x) + c(x, y) < d(y)

= (5 + 2) < ∞

= 7 < ∞

Since 14 is less than the infinity so we update d(D) from ∞ to 14. The value 14 will be added under the D column.

	A	B	C	D	E
A	0	∞	∞	∞	∞
C		10	5	∞	∞
		8		14	7

As we can observe in the above table that 7 is the minimum value among 8, 14, and 7. Therefore, the vertex E is added on the left as shown in the below table:

A	B	C	D	E
A	0	∞	∞	∞	∞
C		10	5	∞	∞
E		8		14	7

The vertex E is selected so we consider all the direct paths from the vertex E. The direct paths from the vertex E are E to A and E to D. Since the vertex A is selected, so we will not consider the path from E to A.

Consider the path from E to D.

d(x, y) = d(x) + c(x, y) < d(y)

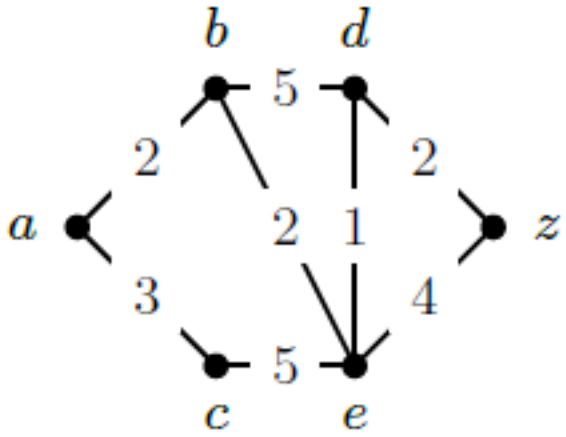
= (7 + 6) < 14

= 13 < 14

Since 13 is less than the infinity so we update d(D) from ∞ to 13. The value 13 will be added under the D column.

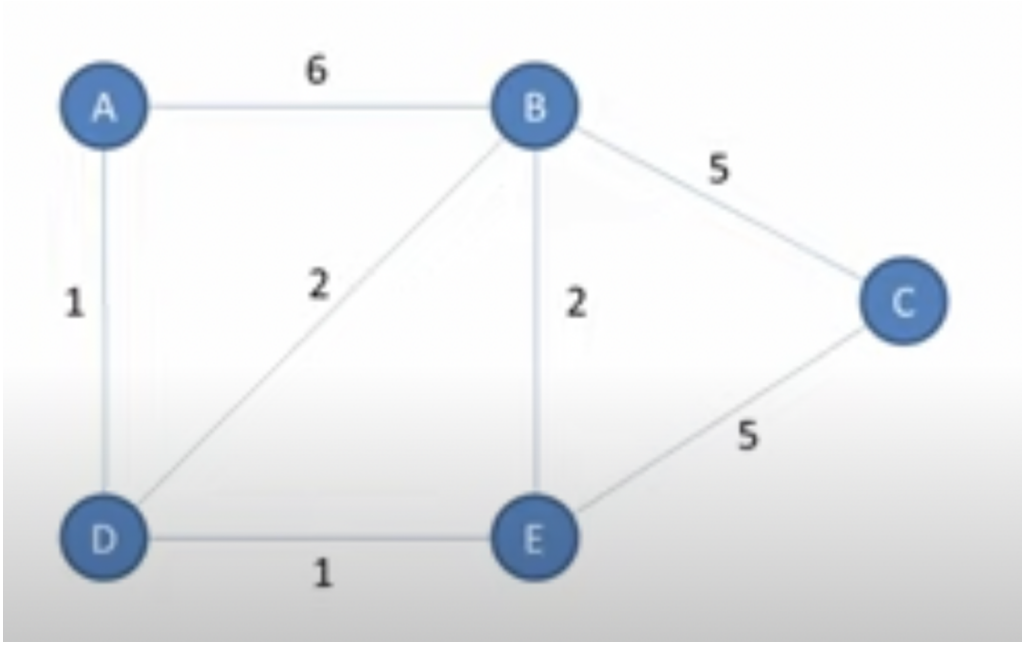
CHECK FOR UNDERSTANDING

1) Use Dijkstra's algorithm to find the shortest path between a and z in the graph below.



What is the length of this path?

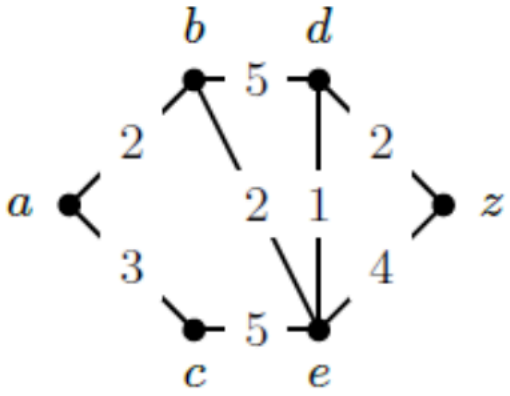
2) Use Dijkstra's algorithm to find the shortest path between a and c in the graph below.



What is the length of this path?

3)

Use Dijkstra's algorithm to find the shortest path between a and z in the graph below.



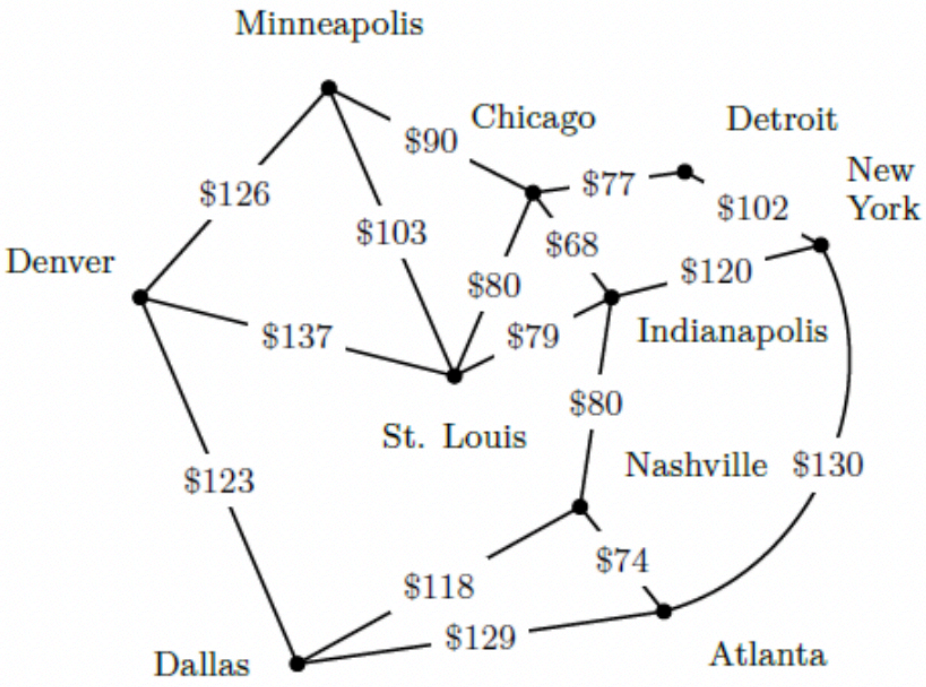
Path: ? → ? → ? → ? → ?

What is the length of this path?

Question Help: Video

4)

The graph below shows the cost of flights between cities.



(a) Use the nearest neighbor algorithm to find a path that starts in St. Louis and visits all the cities shown, while trying to minimize the cost of travel.

? → ? → ? → ? → ? → ? → ? → ? → ? → ? → ? → ?

(b) What is the total cost of the path found in part (a)?

$

(c) Find the cheapest path between Nashville and Denver.

? → ? → ?

What is the cost of this path?

$

(d) Find the cheapest path between Detroit and Denver.

? → ? → ? → ?

What is the cost of this path?

$

Question Help: Video Video

Euler Circuits & Paths

Video

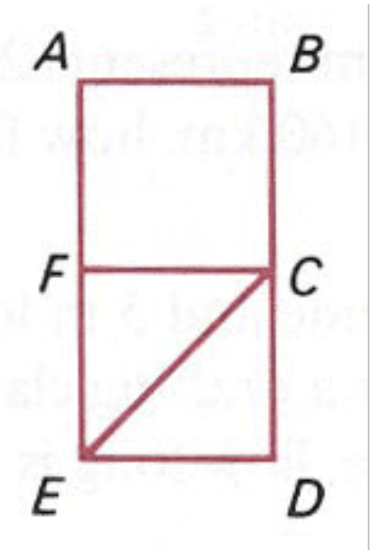
https://www.youtube.com/watch?v=ycRuO-u6rt8

https://www.youtube.com/watch?v=ycRuO-u6rt8

Leonhard Euler first discussed and used Euler paths and circuits in 1736. Rather than finding a minimum spanning tree that visits every vertex of a graph, an Euler path or circuit can be used to find a way to visit every edge of a graph once and only once. This would be useful for checking parking meters along the streets of a city, patrolling the streets of a city, or delivering mail.

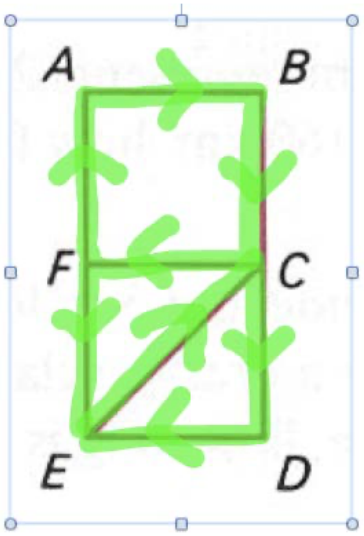
Euler Path: a path that travels through *every* edge of a connected graph once and only once and starts and ends at different vertices. Thus, it is a path that uses every edge in a graph with no repeats. Being a path it does not have to return to the starting vertex.

Example 6.3.1: Euler Path
Figure 6.3.1: Euler Path Example



One Euler path for the above graph is F, A, B, C, F, E, C, D, E as shown below.

Figure 6.3.2: Euler Path

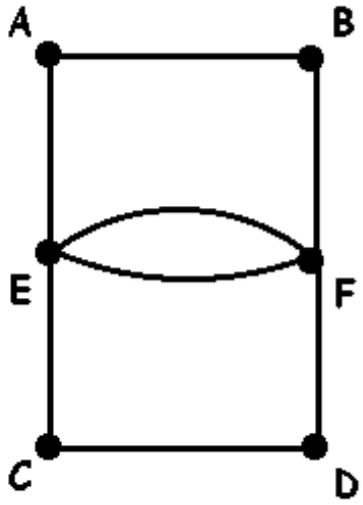


Note: This Euler path travels every edge once and only once and starts and ends at different vertices. This graph cannot have a Euler circuit since no Euler path can start and end at the same vertex without crossing over at least one edge more than once.

Note: This Euler path travels every edge once and only once and starts and ends at different vertices. This graph cannot have a Euler circuit since no Euler path can start and end at the same vertex without crossing over at least one edge more than once.

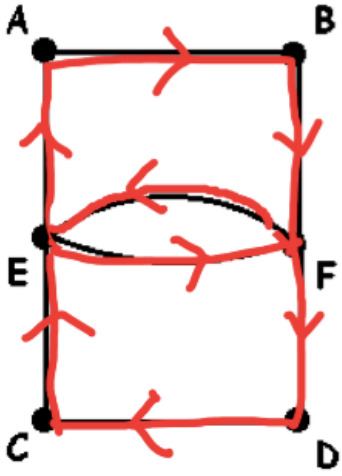
Euler Circuit: a Euler path that starts and ends at the same vertex. Thus, a Euler circuit is a circuit that uses every edge in a graph with no repeats. Being a circuit, it must start and end at the same vertex.

Example 6.3.2: Euler Circuit
Figure 6.3.3: Euler Circuit Example



One Euler circuit for the above graph is E, A, B, F, E, F, D, C, E as shown below.

Figure 6.3.4: Euler Circuit



This Euler path travels every edge once and only once and starts and ends at the same vertex. Therefore, it is also a Euler circuit.

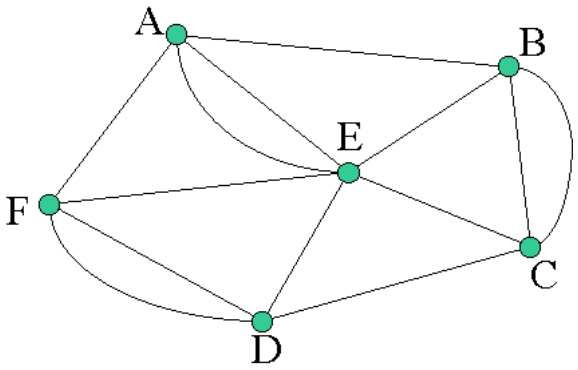
Euler's Theorems:

Euler's Theorem 1: If a graph has any vertices of odd degree, then it cannot have an Euler circuit. If a graph is connected and every vertex has an even degree, then it has at least one Euler circuit (usually more).
Euler's Theorem 2: If a graph has more than two vertices of odd degree, then it cannot have an Euler path. If a graph is connected and has exactly two vertices of odd degree, then it has at least one Euler path (usually more). Any such path must start at one of the odd-degree vertices and end at the other one.
Euler's Theorem 3: The sum of the degrees of all the vertices of a graph equals twice the number of edges (and therefore must be an even number). Therefore, the number of vertices of odd degree must be even.

Finding Euler Circuits:

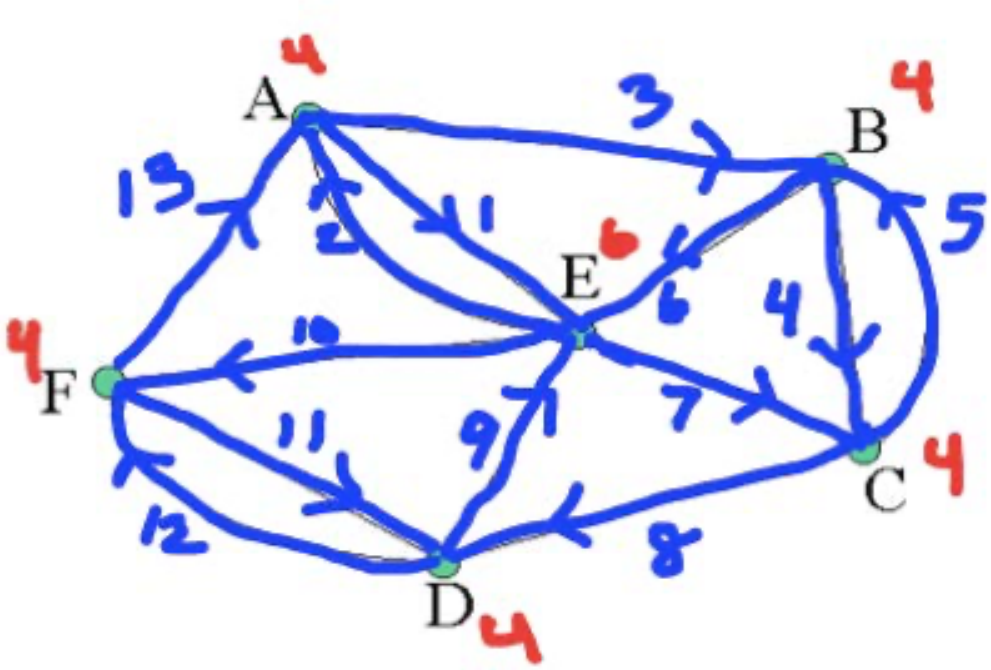
1. Be sure that every vertex in the network has even degree.
2. Begin the Euler circuit at any vertex in the network.
3. As you choose edges, never use an edge that is the only connection to a part of the network that you have not already visited.
4. Label the edges in the order that you travel them and continue this until you have travelled along every edge exactly once and you end up at the starting vertex.

Example 6.3.3: Finding a Euler Circuit
Figure 6.3.5: Graph for Finding a Euler Circuit



The graph shown above has a Euler circuit since each vertex in the entire graph is even degree. Thus, start at one even vertex, travel over each vertex once and only once, and end at the starting point. One example of a Euler circuit for this graph is A, E, A, B, C, B, E, C, D, E, F, D, F, A. This is a circuit that travels over every edge once and only once and starts and ends in the same place. There are other Euler circuits for this graph. This is just one example.

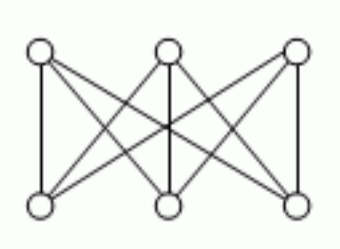
Figure 6.3.6: Euler Circuit



The degree of each vertex is labeled in red. The ordering of the edges of the circuit is labeled in blue and the direction of the circuit is shown with the blue arrows.

CHECK FOR UNDERSTANDING

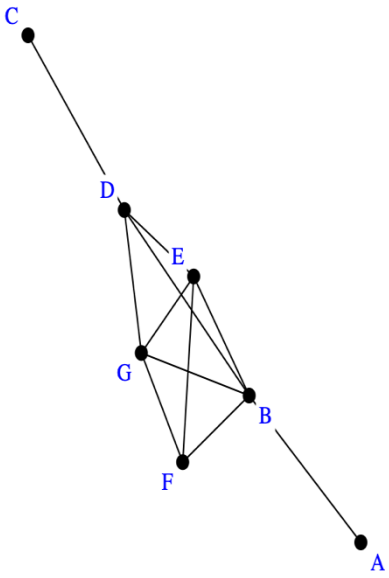
1) Identify the correct description



A) This graph does not have a Euler Circuit

B) This graph has a Euler Circuit

2) Consider the following graph.



A) According to Euler's Theorem, the graph has an Euler path and Euler circuit since the graph has exactly two odd vertices.

B) According to Euler's Theorem, the graph at least one Euler path, but no Euler circuit, since the graph has exactly two odd vertices.

C) According to Euler's Theorem, the graph has no Euler paths nor Euler circuits since the graph has more than two odd vertices.

D) According to Euler's Theorem, the graph at least one Euler circuit (and thus at least one Euler path) since the graph has no odd vertices.

E) According to Euler's Theorem, the graph at least one Euler circuit, but no Euler path, since the graph has exactly two odd vertices.

F) According to Euler's Theorem, the graph has no Euler paths nor Euler circuits since the graph has exactly two odd vertices.

3) For the situation below, should the problem be solved using a Euler circuit or a Hamiltonian circuit?

A health inspector need to visit certain restaurants in town to determine if they are obeying the health code.

A) Euler circuit

B) Hamiltonian circuit

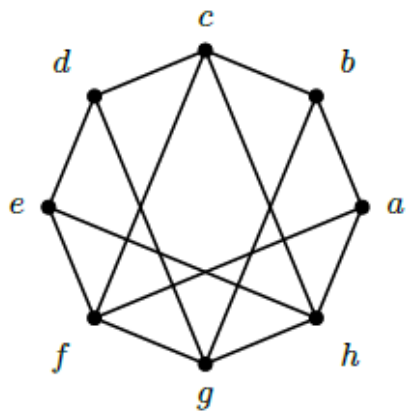
4) For the situation below, should the problem be solved using a Euler circuit or a Hamiltonian circuit?

A tour guide wants to plan a route through an art museum that covers all the rooms and hallways.

A) Euler circuit

B) Hamiltonian circuit

5) Does the graph below have a Euler path/circuit?



A) There is a Euler circuit

B) There is no Euler circuit, but there is a Euler path

C) There is no Euler path or circuit

Hamiltonian Circuits & Paths

https://www.youtube.com/watch?v=AamHZhAmR7o

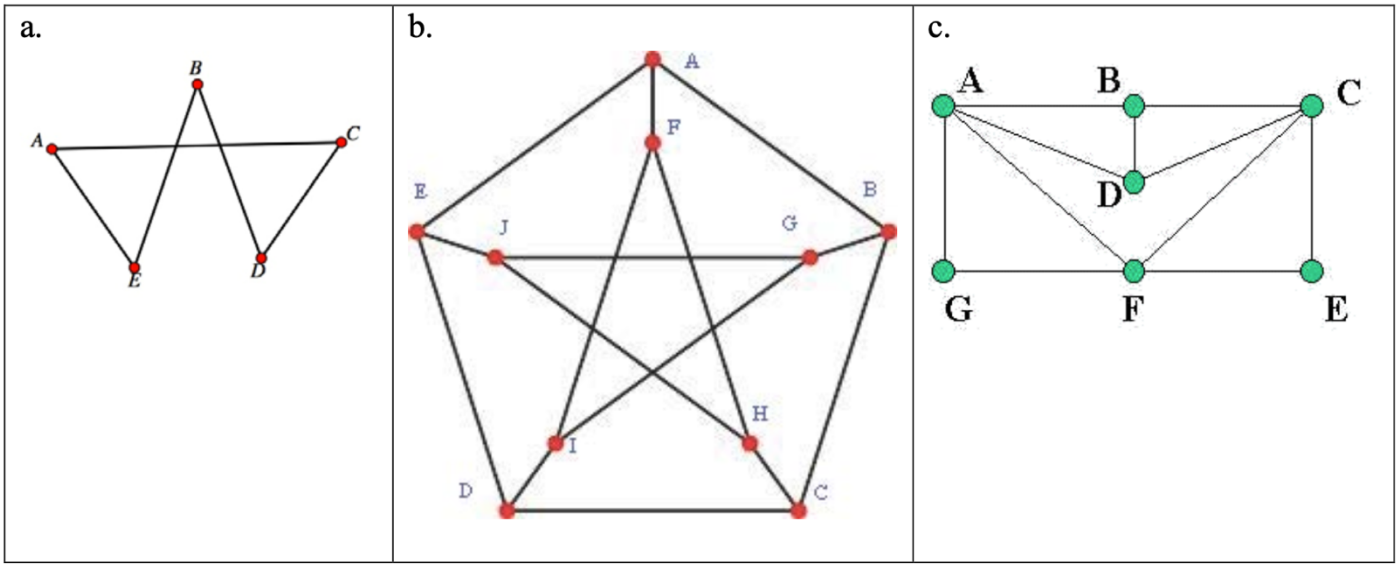
https://www.youtube.com/watch?v=dQr4wZCiJJ4 (Indian)

The Traveling Salesman Problem (TSP) is any problem where you must visit every vertex of a weighted graph once and only once, and then end up back at the starting vertex. Examples of TSP situations are package deliveries, fabricating circuit boards, scheduling jobs on a machine and running errands around town.

Hamilton Circuit: a circuit that must pass through each vertex of a graph once and only once. A path that starts and ends at the same vertex and goes through every vertex exactly once.

Hamilton Path: a path that must pass through each vertex of a graph once and only once

Example 6.4.1: Hamilton Path:
Figure 6.4.1: Examples of Hamilton Paths



Not all graphs have a Hamilton circuit or path. There is no way to tell just by looking at a graph if it has a Hamilton circuit or path like you can with

a Euler circuit or path. You must do trial and error to determine this. By the way if a graph has a Hamilton circuit, then it has a Hamilton path. Just do not go back to home.

Graph a. has a Hamilton circuit (one example is ACDBEA)

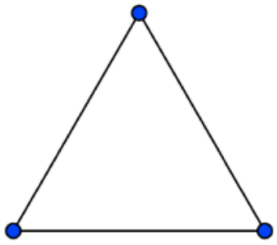
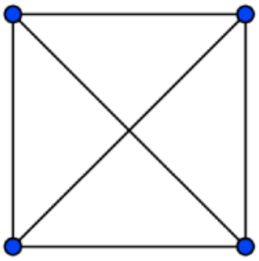
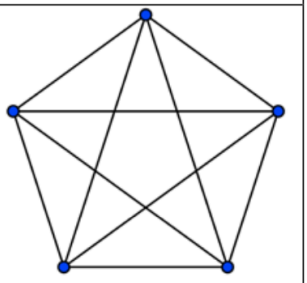
Graph b. has no Hamilton circuits, though it has a Hamilton path (one example is ABCDEJGIFH)

Graph c. has a Hamilton circuit (one example is AGFECDBA)

Complete Graph: A complete graph is a graph with N vertices in which every pair of vertices is joined by exactly one edge. The symbol used to denote a complete graph is K_N.

Example 6.4.2: Complete Graphs

Figure 6.4.2: Complete Graphs for N = 2, 3, 4, and 5

a. K_2	b. K_3	c. K_4	d. K_5
			
two vertices and one edge	three vertices and three edges	four vertices and six edges	five vertices and ten edges

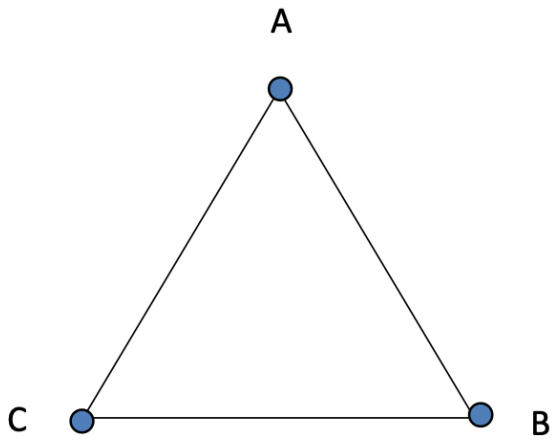
In each complete graph shown above, there is exactly one edge connecting each pair of vertices. There are no loops or multiple edges in complete graphs. Complete graphs do have Hamilton circuits.

Reference Point: the starting point of a Hamilton circuit

Example 6.4.3: Reference Point in a Complete Graph

Many Hamilton circuits in a complete graph are the same circuit with different starting points. For example, in the graph K_3, shown below in Figure 6.4.3, ABCA is the same circuit as BCAB, just with a different starting point (reference point). We will typically assume that the reference point is A.

Figure 6.4.3: K_3



Number of Hamilton Circuits: A complete graph with N vertices is (N-1)! Hamilton circuits. Since half of the circuits are mirror images of the other half, there are actually only half this many unique circuits.

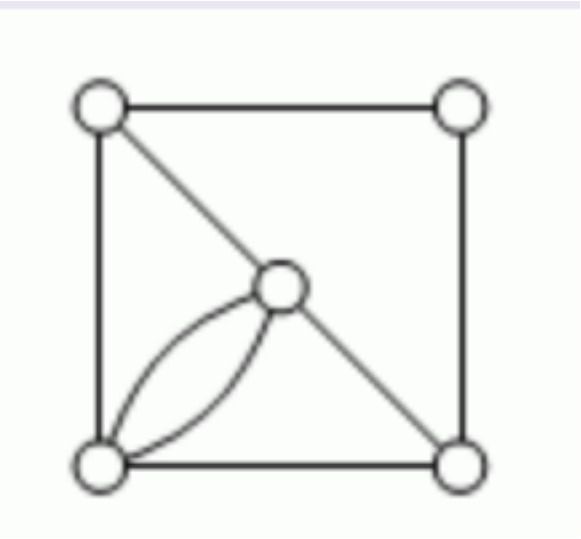
Example 6.4.4: Number of Hamilton Circuits

How many Hamilton circuits does a graph with five vertices have?

$(N - 1)! = (5 - 1)! = 4! = 4*3*2*1 = 24$ Hamilton circuits.

CHECK FOR UNDERSTANDING

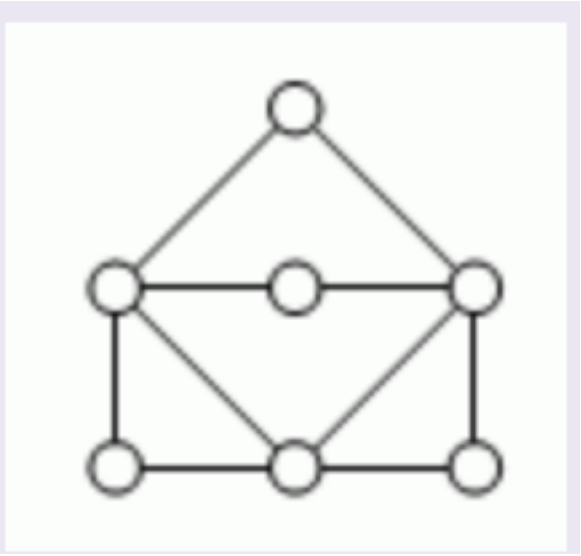
1) Determine whether each figure has a Hamiltonian Circuit



A) This graph does not have a Hamiltonian Circuit

B) This graph has a Hamiltonian Circuit

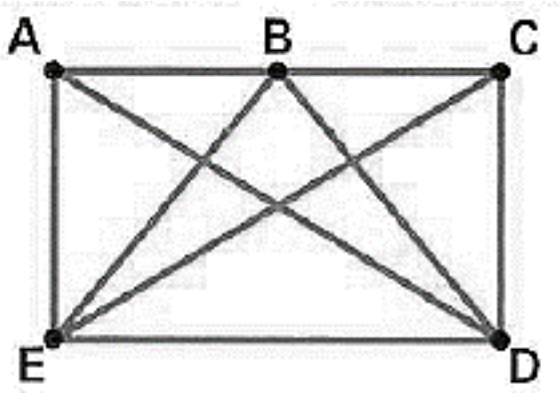
2) Determine whether each figure has a Hamiltonian Circuit



A) This graph does not have a Hamiltonian Circuit

B) This graph has a Hamiltonian Circuit

3) Find a Hamiltonian circuit in the graph that begins with AD.



Select ALL correct responses given.

A) ADCEBA

B) ADBCEDA

C) ADBECA

D) ADECBA

E) ADBCEA

How to solve a Traveling Salesman Problem (TSP):

https://www.youtube.com/watch?v=Q4zHb-Swzro (Indian)

https://www.youtube.com/watch?v=1pmBjIZ20pE

Lesson:
https://math.libretexts.org/Bookshelves/Applied_Mathematics/Book%3A_College_Mathematics_for_Everyday_Life_(Inigo_et_al)/06%3A_Graph_Theory/6.04%3A_Hamiltonian_Circuits#:~:text=Example%206.4.,-5%3A%20Brute%20Force&text=Recall%20the%20way%20to%20find,%3D%20(4%20%E2%80%93%201)!

The Traveling Salesman Problem (TSP) is any problem where you must visit every vertex of a weighted graph once and only once, and then end up back at the starting vertex. Examples of TSP situations are package deliveries, fabricating circuit boards, scheduling jobs on a machine and running errands around town.

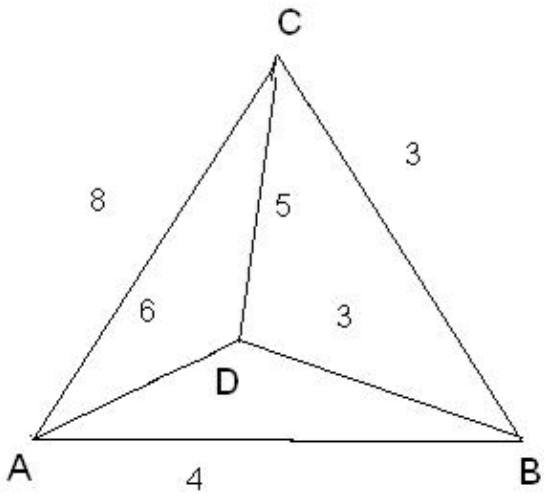
A traveling salesman problem is a problem where you imagine that a traveling salesman goes on a business trip. He starts in his home city (A) and then needs to travel to several different cities to sell his wares (the other cities are B, C, D, etc.). ***To solve a TSP, you need to find the cheapest way for the traveling salesman to start at home, A, travel to the other cities, and then return home to A at the end of the trip. This is simply finding the Hamilton circuit in a complete graph that has the smallest overall weight. There are several different algorithms that can be used to solve this type of problem.***

A. Brute Force Algorithm

1. List all possible Hamilton circuits of the graph.
2. For each circuit find its total weight.
3. The circuit with the least total weight is the optimal Hamilton circuit.

Example 6.4.5: Brute Force Algorithm:
Figure 6.4.4: Complete Graph for Brute Force Algorithm

.



Suppose a delivery person needs to deliver packages to three locations and return to the home office A. Using the graph shown above in Figure 6.4.4, find the shortest route if the weights on the graph represent distance in miles.

Recall the way to find out how many Hamilton circuits this complete graph has. The complete graph above has four vertices, so the number of Hamilton circuits is:

$(N - 1)! = (4 - 1)! = 3! = 3*2*1 = 6$ Hamilton circuits.

However, three of those Hamilton circuits are the same circuit going the opposite direction (the mirror image).

Hamilton Circuit	Mirror Image	Total Weight (Miles)
ABCDA	ADCBA	18
ABDCA	ACDBA	20
ACBDA	ADBCA	20

The solution is ABCDA (or ADCBA) with total weight of 18 mi. This is the optimal solution.

B. Nearest-Neighbor Algorithm:

1. Pick a vertex as the starting point.
2. From the starting point go to the vertex with an edge with the smallest weight. If there is more than one choice, choose at random.
3. Continue building the circuit, one vertex at a time from among the vertices that have not been visited yet.
4. From the last vertex, return to the starting point.

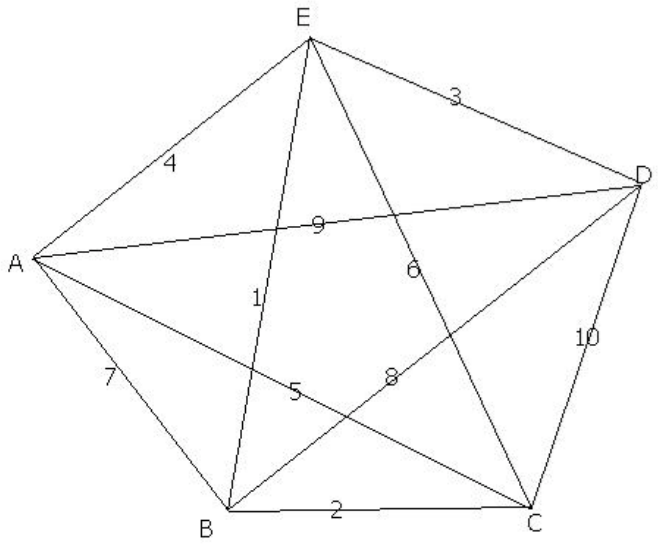
Example 6.4.6: Nearest-Neighbor Algorithm

https://www.youtube.com/watch?v=zPgsNsOfxQ8

A delivery person needs to deliver packages to four locations and return to the home office A as shown in Figure 6.4.5 below. Find the shortest route if the weights represent distances in miles.

Starting at A, E is the nearest neighbor since it has the least weight, so go to E. From E, B is the nearest neighbor so go to B. From B, C is the nearest neighbor so go to C. From C, the first nearest neighbor is B, but you just came from there. The next nearest neighbor is A, but you do not want to go there yet because that is the starting point. The next nearest neighbor is E, but you already went there. So go to D. From D, go to A since all other vertices have been visited.

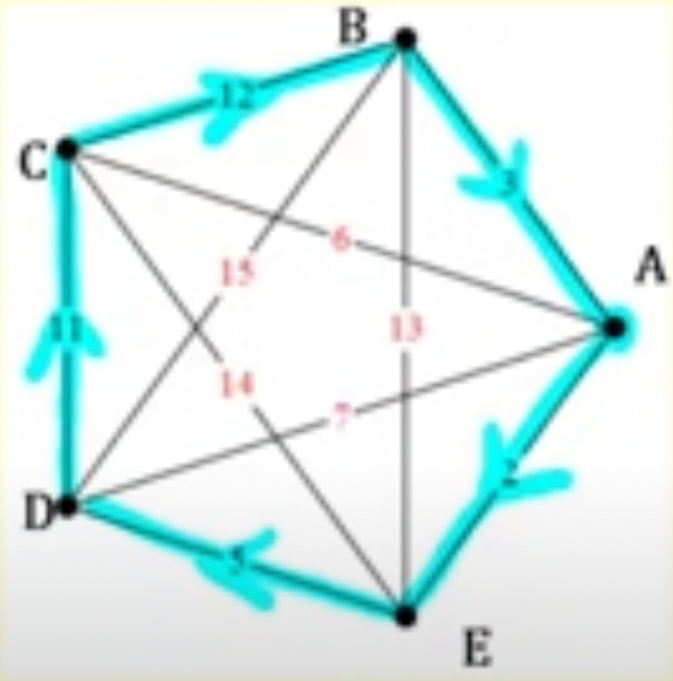
Figure 6.4.5: Complete Graph for Nearest-Neighbor Algorithm



The solution is AEBCDA with a total weight of 26 miles. This is not the optimal solution, but it is close, and it was a very efficient method.

EXAMPLE 1:

Draw and find the weight of the Hamiltonian circuit produced using the **nearest neighbor algorithm** starting at the vertex A.



AEDCBA

or

ABCDEA

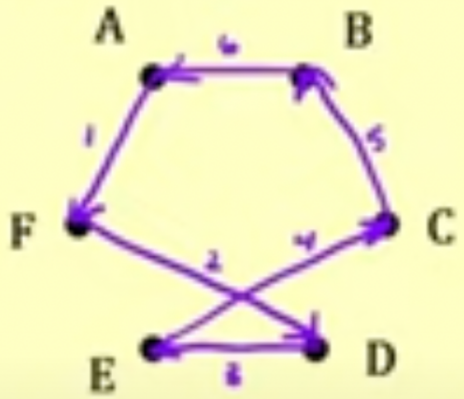
Total Weight

2+5+11+12+3 = 33

EXAMPLE 2:

The weights of edges in a graph are shown in the table. Apply the nearest neighbor algorithm to find a Hamiltonian circuit starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDEFA

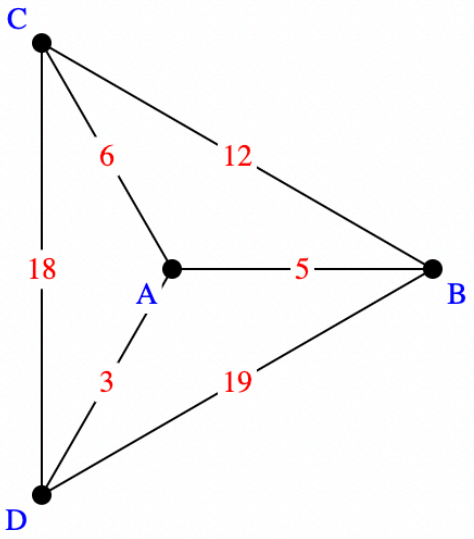
	A	B	C	D	E	F
A	--	58				
B	58	--	44	17	53	36
C	37		--	45	48	50
D	57			--		5
E	49			6	--	24
F	34					--



AFDECBA
or
ABCEDFA

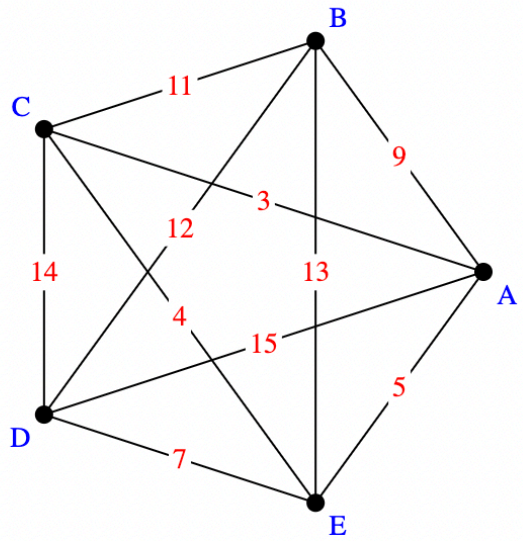
Total Weight:
34+58+44+6+48+5
= 195

WE DO 1)



Apply the nearest neighbor algorithm to the graph above starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDA

YOU DO 1)



Apply the nearest neighbor algorithm to the graph above starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDEA

WE DO 2)

	A	B	C	D	E	F
A	--	30	32	40	12	31
B	30	--	23	45	26	35
C	32	23	--	20	13	41
D	40	45	20	--	47	11
E	12	26	13	47	--	27
F	31	35	41	11	27	--

The weights of edges in a graph are shown in the table above. Apply the nearest neighbor algorithm to the graph starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDEFA

Thank you for listening!
Do the assigned activities!