week-in-review

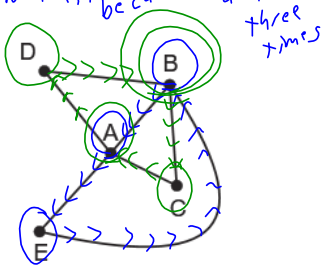
9/9/19

CHAPTER 2 – BUSINESS EFFICENCY

Path? ⇒ Not a path
↓ Y
Circuit? N
Special Circuit
Special Path
Euler Hamiltonian
Euler Ham

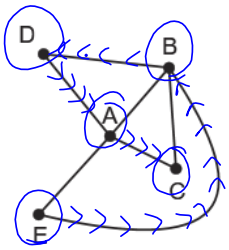
A path that visits every vertex exactly once is a ***Hamiltonian path***.
A circuit that visits every vertex exactly once (except the beginning point will be visited again) is a ***Hamiltonian circuit***.
Classify the following choosing from the terms: not a circuit, not a path, path, circuit, Euler path, Euler circuit, Hamiltonian path, Hamiltonian circuit.

Not Hamiltonian because use A twice and B three times



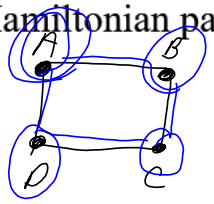
BAEBCADB

Euler Circuit



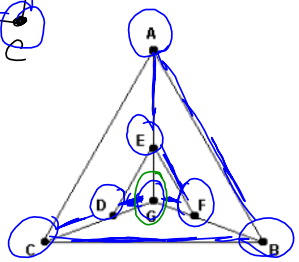
EBDAC

Hamiltonian Path



ABCDA

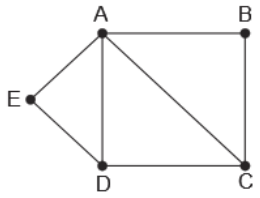
Euler and Hamiltonian Circuit



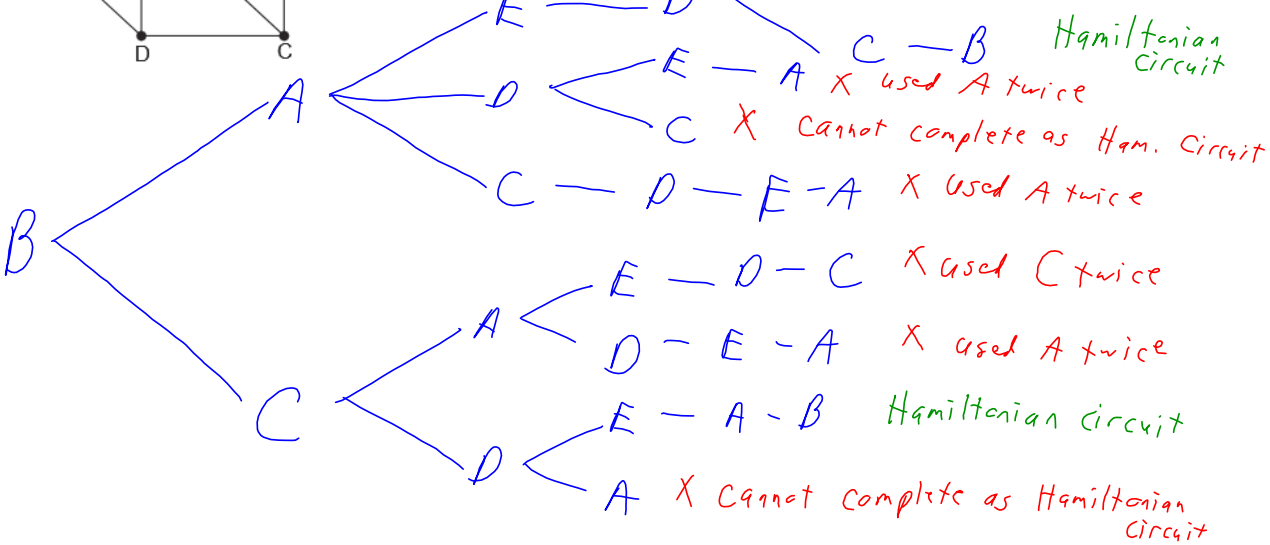
GDCBAEFG

Hamiltonian Circuit

Use the method of trees to find all Hamiltonian circuits in the graph below starting at vertex B. Make note of any mirror images.



BAEDCB
BCDEAB
mirror images of each other

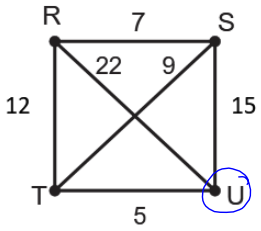


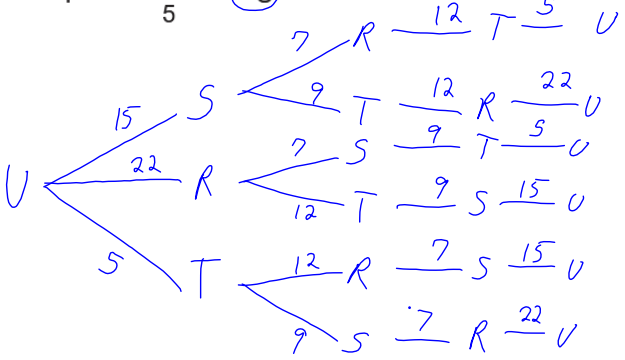
The ***Traveling Salesman Problem*** (***TSP***) tries to find a least cost Hamiltonian circuit.

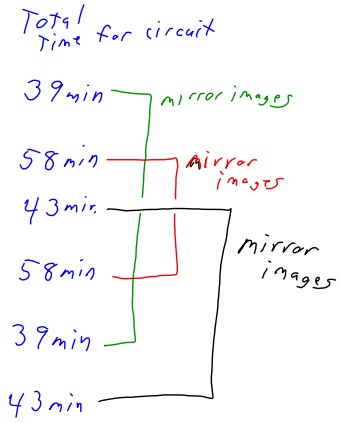
The graph below shows the time in minutes to travel between the vertices R, S, T, and U. Find the least cost Hamiltonian circuit for this graph starting at vertex U.

U S R T U or U T R S U at 39min

mirror images





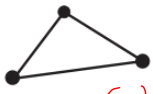


A ***complete*** graph is a graph in which every pair of vertices is connected by exactly one edge. How many edges does a graph with *v* vertices have?

v = 2
e = 1

$e = \frac{2(2-1)}{2} = 1$

v = 3
e = 3

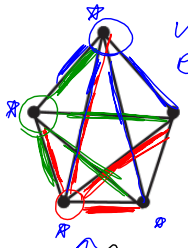


$e = \frac{3(3-1)}{2} = 3$

v = 4
e = 6

$e = \frac{4(4-1)}{2} = 6$

v = 5
e = 10



$E = \frac{5(5-1)}{2} = 10$

$\#\text{edges} = \frac{V(V-1)}{2}$ ← Because counted each edge twice

Hamiltonian Circuits

$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1$

$= 120$ Hamiltonian Circuits

If circuit and mirror image are counted as the same, we have

$\frac{5!}{2} = 60$ Ham. Circuits if mirror images are counted the same as each other

If I choose where to start.....

$(5-1)! = 1 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1$

$= 24$

If circuit and mirror image counted the same...

$\frac{(5-1)!}{2} = \frac{4!}{2} = 12$

Brute Force Method on a complete graph with *n* vertices:

- There are $n!$ Hamiltonian circuits in a complete graph.
- There are $\frac{n!}{2}$ Hamiltonian circuits in a complete graph, if a circuit and its mirror image are not counted as separate circuits.
- If a starting point is specified, there are $(n-1)!$ Hamiltonian circuits in a complete graph.
- If a starting point is specified, there are $\frac{(n-1)!}{2}$ Hamiltonian circuits in a complete graph, if a circuit and its mirror image are not counted as separate circuits.

Remember that $n! = n(n-1)(n-2)\cdots(3)(2)(1)$

Consider a complete graph with 8 vertices.

a) How many edges does the graph have?

$E = \frac{V(V-1)}{2} = \frac{8(7)}{2} = \frac{56}{2} = 28$

b) If you can start at any vertex, how many different circuits are there if every vertex is visited exactly once (and the beginning vertex is visited twice)? This is asking for the number of Hamiltonian circuits

$n! = 8! = 40,320$

c) If you can start at any vertex, how many Hamiltonian circuits are there if mirror images are not counted as separate circuits?

$\frac{n!}{2} = \frac{8!}{2} = 20,160$

d) If you start at a specified vertex, how many Hamiltonian circuits are there?

$(n-1)! = (8-1)! = 7! = 5040$

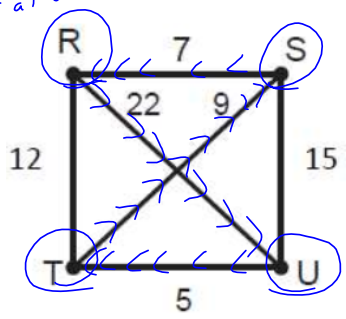
A ***heuristic algorithm*** is an algorithm that is fast but may not be optimal.
A ***greedy algorithm*** is one in which the choices are made by what is best at the next step.

example of a heuristic and greedy algorithm

Nearest Neighbor (NN) Algorithm (for finding low-cost Hamiltonian circuits):
Starting from the home city, visit the nearest city first. Then visit the nearest city that has not already been visited. Return to the home city when no other choices remain.

The graph below shows the time in minutes to travel between the vertices R, S, T, and U. Find a low-cost Hamiltonian circuit for this graph using the nearest neighbor algorithm starting at U.
Is the result different if you start at T?

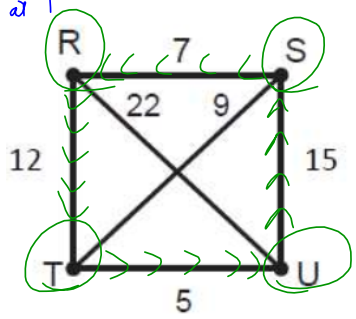
start at U



U $\xrightarrow{5}$ T $\xrightarrow{9}$ S $\xrightarrow{7}$ R $\xrightarrow{22}$ U

43 min

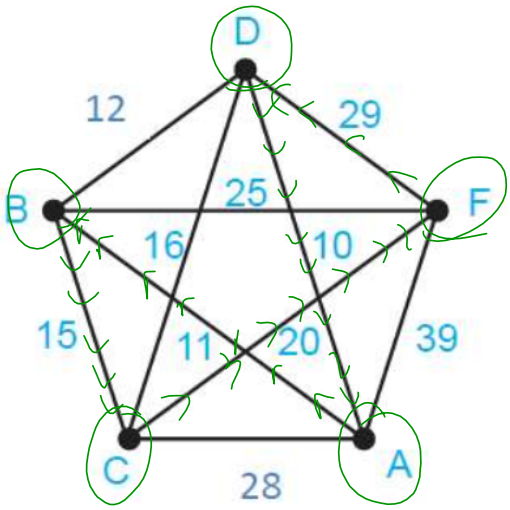
start at T



T $\xrightarrow{5}$ U $\xrightarrow{15}$ S $\xrightarrow{7}$ R $\xrightarrow{12}$ T

39 min

Use the nearest neighbor algorithm to find a low-cost Hamiltonian circuit for the graph below starting at D. The values on the edges are the distance in km between the vertices.



D $\xrightarrow{10}$ A $\xrightarrow{11}$ B $\xrightarrow{15}$ C $\xrightarrow{20}$ F $\xrightarrow{29}$ D

85 km

The chart below shows the distance between vertices of a complete graph in miles. Find a short Hamiltonian circuit using the nearest neighbor algorithm starting at vertex E

Ending Vertex

save for last

Starting Vertex	A	B	C	D	E	F
A	0	26	49	50	7	34
B	26	0	48	24	28	36
C	49	48	0	18	40	15
D	50	24	18	0	13	17
E	7	28	40	13	0	20
F	34	36	15	17	20	0

E $\xrightarrow{7}$ A $\xrightarrow{26}$ B $\xrightarrow{24}$ D $\xrightarrow{17}$ F $\xrightarrow{15}$ C $\xrightarrow{40}$ E

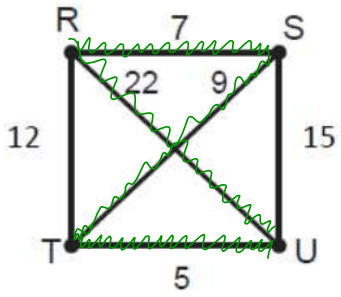
EABDFCE at a cost of 129 miles

Another Heuristic Alg.

Sorted Edges (SE) Algorithm (for finding low-cost Hamiltonian circuits):

1. Arrange edges of the complete graph in order of increasing cost
2. Select the lowest cost edge that has not already been selected that
   a. Does not cause a vertex to have 3 edges
   b. Does not close the circuit unless all vertices have been included.

Find a low-cost Hamiltonian circuit using the sorted edges algorithm for the graphs below. Costs are in minutes.

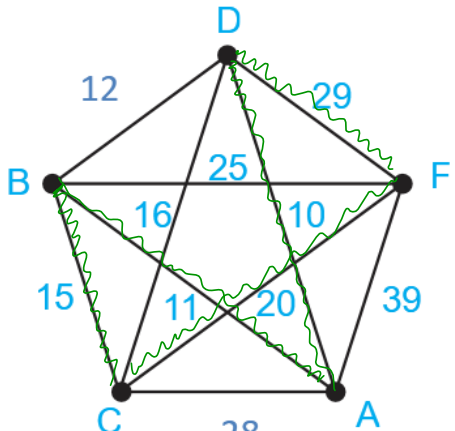


5 ✓ 7 ✓ 9 ✓ 12 ✗ 15 ✗ 22 ✓

12 caused T to have 3 edges. Also creates TRST circuit that does include U

15 caused S to have 3 edges. Also creates STUS circuit that does not include R

RSTUR or other arrangements of that circuit at a cost of 43 min



10 ✓ 11 ✓ 12 ✗ 15 ✓ 16 ✗ 20 ✓ 25 ✗ 28 ✗ 29 ✓ 39

12 closes circuit too early

16 closes circuit too early

25 causes 3 edges at B. Also creates a mini-circuit BCFB

28 causes 3 edges at C and A. Also creates a mini-circuit

DFCBAD or other variations of that circuit at a cost of 85 min

How do these solutions compare to the solutions using the Nearest Neighbor algorithm from our specified starting points?

1st graph was the same as NN starting with U but was more than NN starting at T

2nd graph was the same as NN starting at D

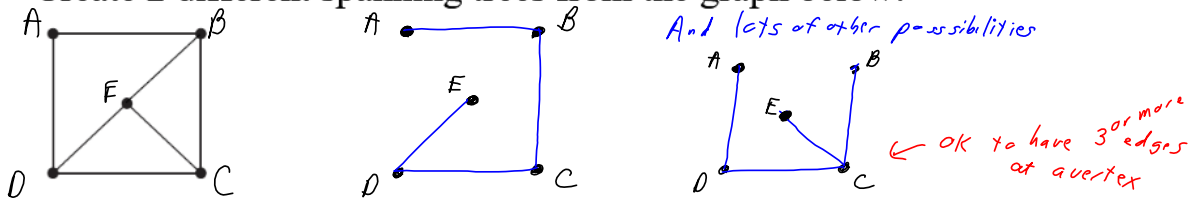
So the two methods may or may not give the same solution.

A connected graph that has no circuits is a ***tree***. A ***spanning tree*** is a tree that has all the vertices of the original graph.

To create a spanning tree from a graph,

1. Copy the vertices with no edges
2. Add edges back one by one until you have a connected graph that uses all vertices and contains no circuits

Create 2 different spanning trees from the graph below:



A ***minimal spanning tree*** is a spanning tree with the smallest possible weight.

optimal – the best

Kruskal's Algorithm (for finding minimum-cost spanning trees): (not heuristic)

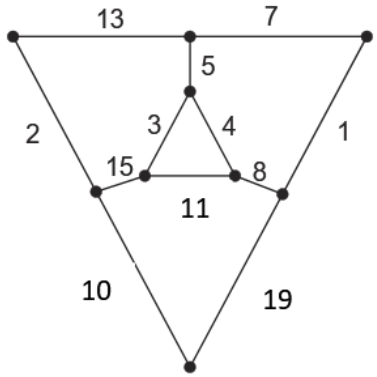
Finding a minimum-cost spanning tree by adding edges in order of increasing cost so that no circuit is formed.

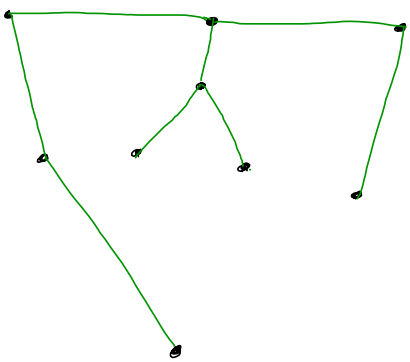
use all vertices
Connected
No circuits

Use Kruskal's algorithm to find the minimal spanning tree for the graph below. The weights are in minutes. ~~What is the cost?~~

Notice, it is ok to have 3 edges at the same vertex

1 2 3 4 5 7 8 10 11 13 15 19
✓ ✓ ✓ ✓ ✓ ✓ X ✓ X ✓
causes circuit
causes circuit





Finished, we have a minimum cost spanning tree of 45 min

A list of vertices connected by arrows is a ***directed graph*** or ***digraph***.

If the tasks cannot be completed in a random order, then the order can be specified in an ***order-requirement digraph***.

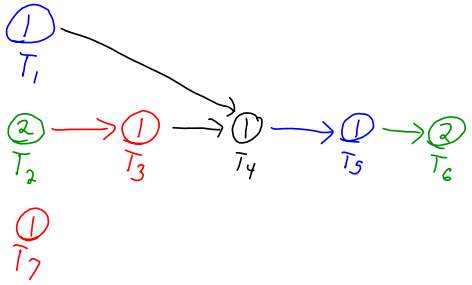
If the time to complete a task is shown on the digraph, it is a ***weighted*** digraph.

An ***independent task*** is one that can be done independently of any of the other tasks. It is not connected by arrows to other tasks.

Quick Dinner * class chose times for completion of tasks

To heat soup, you need to find a bowl, pick your can of soup, open the can of soup, pour the soup into the bowl, add a can of water to the bowl, and heat. You also need to get the crackers. Show these tasks in a weighted order requirement digraph.

T_1 = Find a bowl
T_2 = Pick your can of soup
T_3 = Open the can of soup
T_4 = Pour soup into the bowl
T_5 = Add can of water to bowl
T_6 = Heat
T_7 = Get Crackers



Which tasks, if any, are independent? T_7

How long is required to make dinner if one person is available (no multi-tasking)? 9min (add up all the times)

How long is required to make dinner if a lot of people are available to help? 7min

1 person get Crackers
1 person get bowl
1 person does the rest

longest time

A ***critical path*** on the diagraph is the *longest path* and it determines the *earliest completion time* (the earliest possible time for the completion of all the tasks making up the job in the digraph).

What is the critical path for making the soup? What is the earliest completion time?

Choices

$T_1\ T_4\ T_5\ T_6 = 5$ min

$T_2\ T_3\ T_4\ T_5\ T_6 = 7$ min ← Longest time

$T_7 = 1$ min

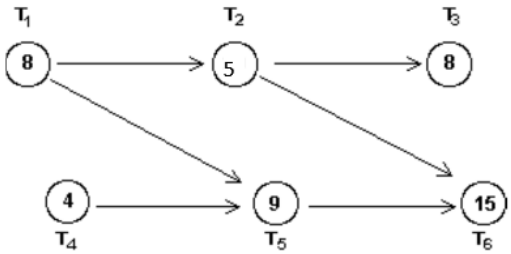
Critical path

$T_2\ T_3\ T_4\ T_5\ T_6$

at a cost of 7min

where you would put your efforts to speed up the process

What is the critical path in the digraph below? What is the earliest completion time? 32min



$T_1\ T_2\ T_3 = 21$min

$T_1\ T_2\ T_6 = 28$min

$T_1\ T_5\ T_6 = 32$min ← Critical path

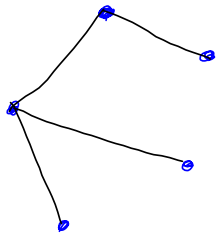
$T_4\ T_5\ T_6 = 28$min

critical path is $T_1\ T_5\ T_6$ and it takes 32min

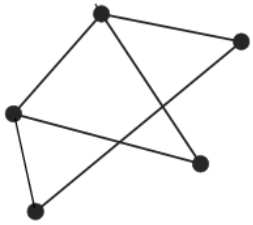
SAMPLE EXAM QUESTIONS FROM CHAPTER 2

1. Create a spanning tree from the graph on the right.

NO circuits & use all the vertices & connected graph



and lots of other possibilities



2. After a storm, a rescue team needs to visit every home (about 1000 homes) to make sure no one is trapped. The technique most likely to be useful in solving this problem is:

(A) finding an Euler circuit on a graph.

edges

(B) applying the nearest-neighbor algorithm for finding a Hamiltonian circuit.

vertices

(C) applying Kruskal's algorithm for finding a minimum-cost spanning tree for a graph.

Rescue team wants to get home, so need a circuit

(D) applying the method of trees for the traveling salesman problem.

takes too long - even with a computer

(E) None of these/need more information

Hamiltonian circuit

D gives the optimal solution, but it takes an incredibly long time to find that solution.

B gives a reasonably good solution so we can get started

vertices

3. Which of the following statements are true about a spanning tree? Mark all true answers

(A) A spanning tree must contain all the edges of the original graph. F

(B) A spanning tree must contain all of the vertices of the original graph.

(C) A spanning tree must contain a circuit

(D) A spanning tree must not contain a circuit

(E) None of these/need more information

also must be connected

4. Which, if any, of the statements below are true about a heuristic algorithm? Mark all true answers.

(A) An optimal result will be found

(B) An optimal result may be found Sometimes they are optimal, but not always

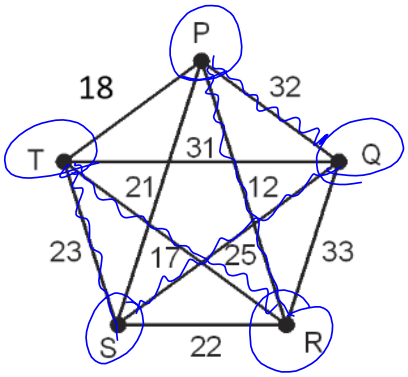
(C) An optimal result will not be found

(D) The brute force method is an example of a heuristic algorithm.

(E) The nearest neighbor method is an example of a heuristic algorithm.

also sorted edges is a heuristic algorithm

5. The graph below shows the distance between houses in a rural neighborhood in yards. Use the nearest neighbor method to find a low-cost solution to the traveling salesman problem for these houses starting at house P.



P →12 R →17 T →23 S →25 Q →32 P

109 yd

Make sure you go home for a circuit

6. Use the sorted edges algorithm to find a low-cost solution to the same TSP for the graph above.

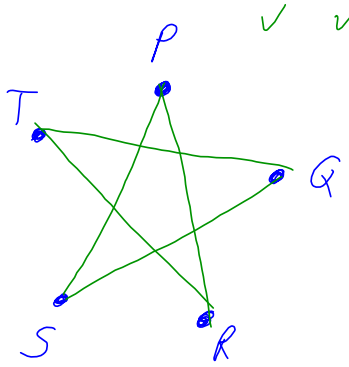
12 ✓ 17 ✓ 18 X 21 ✓ 22 X 23 X 25 ✓ 31 ✓ 32 33

Done

Forms mini-circuit

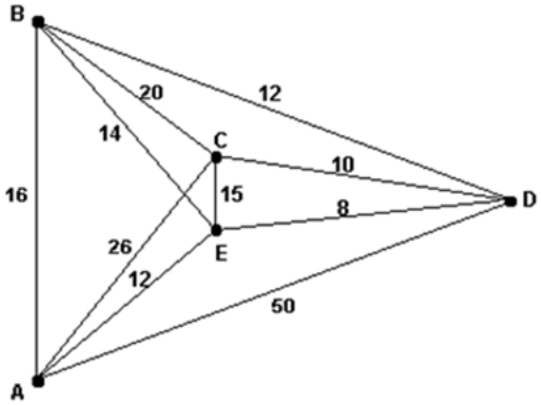
Causes 3 edges at R and forms mini-circuit

Closes circuit too early



106 yd

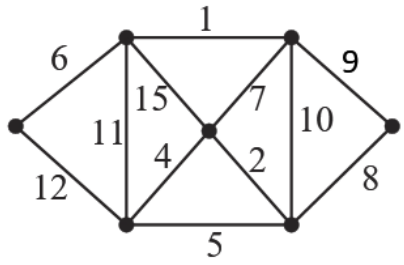
7. For the graph below, find the Hamiltonian circuit obtained by using the nearest-neighbor algorithm, starting at A. What is the cost if the numbers shown represent the distance in miles?



8. The chart below shows the travel time between cities in hours. Use the sorted edges method to find a good solution to the TSP.

	G	H	I	J	K	L
G	0	16	21	13	32	9
H	16	0	19	29	37	30
I	21	19	0	47	27	8
J	13	29	47	0	23	17
K	32	37	27	23	0	22
L	9	30	8	17	22	0

9. Apply Kruskal's algorithm to create a minimum cost spanning tree from the given graph. The edges show the time between vertices in seconds. What is the total time?



10. What is the critical path for the digraph below? The time for each task is given in minutes. What is the earliest completion time for these tasks?