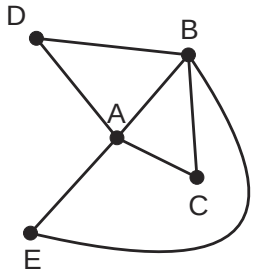
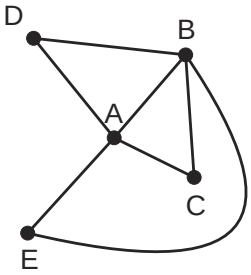
CHAPTER 2 – BUSINESS EFFICENCY

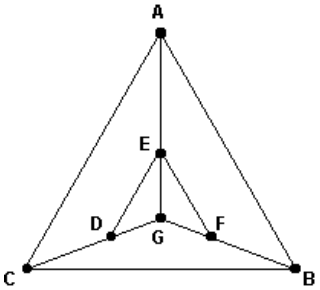
A path that visits every vertex exactly once is a ***Hamiltonian path***.
A circuit that visits every vertex exactly once (except the beginning point will be visited again) is a ***Hamiltonian circuit***.
Classify the following choosing from the terms: not a circuit, not a path, path, circuit, Euler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



BAEBCADB

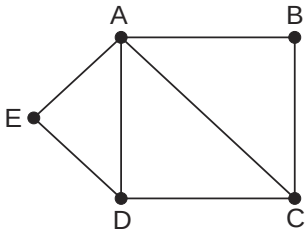


EBDAC



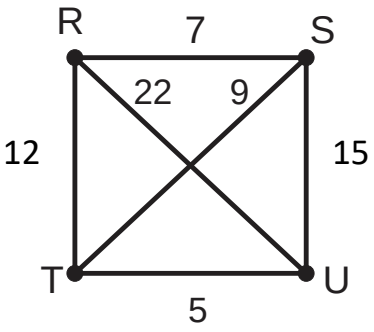
GDCBAEFG

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

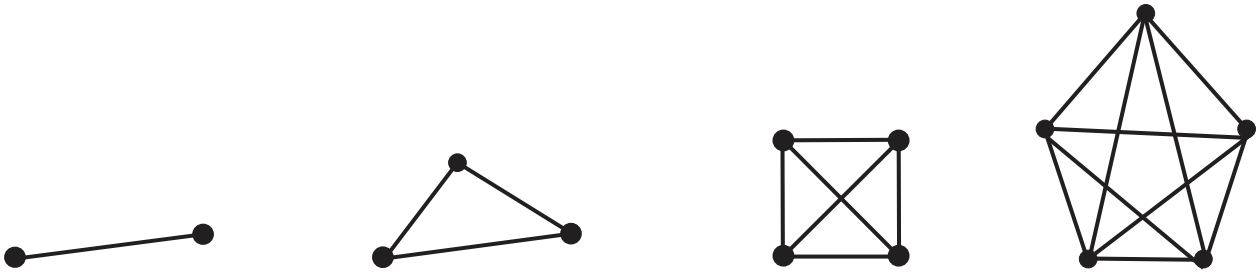


The ***Traveling Salesman Problem*** (***TSP***) tries to find a least cost Hamiltonian circuit.

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.



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?



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?

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)?

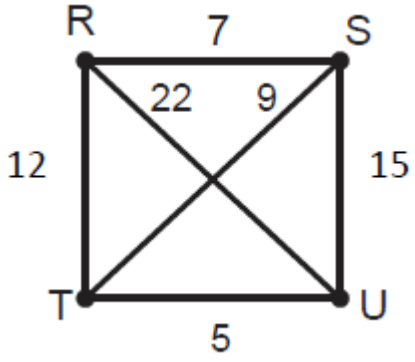
c) If you can start at any vertex, how many Hamiltonian circuits are there if mirror images are not counted as separate circuits?

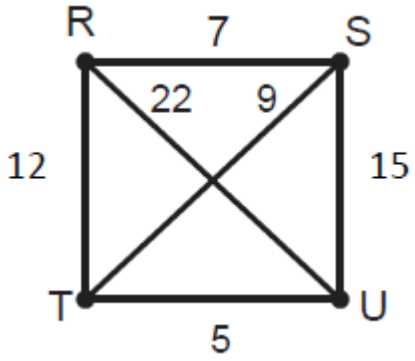
d) If you start at a specified vertex, how many Hamiltonian circuits are there?

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.

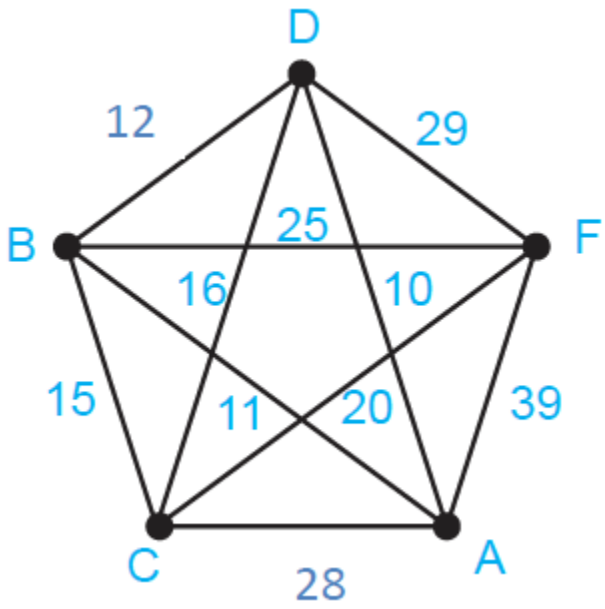
Nearest Neighbor (NN) Algorithm (for finding low-cost Hamiltonian circuits):
Starting from the home city, visit the nearest city first. Then visit the nearest city that has not already been visited. Return to the home city when no other choices remain.

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?





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.



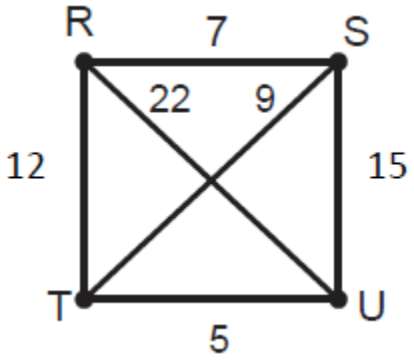
The chart below shows the distance between vertices of a complete graph in miles. Find a short Hamiltonian circuit using the nearest neighbor algorithm starting at vertex E

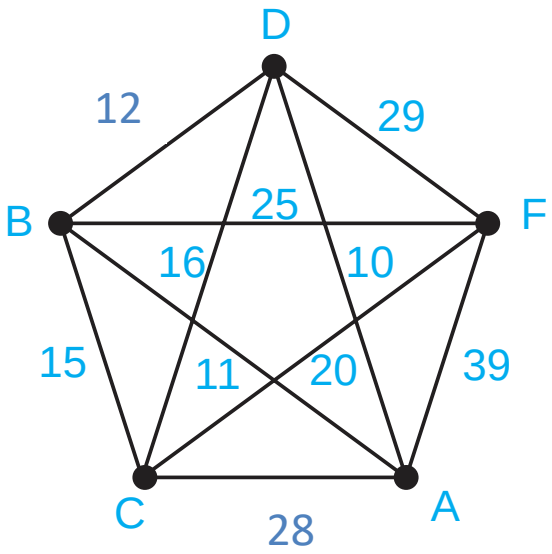
	A	B	C	D	E	F
A	0	26	49	50	7	34
B	26	0	48	24	28	36
C	49	48	0	18	40	15
D	50	24	18	0	13	17
E	7	28	40	13	0	20
F	34	36	15	17	20	0

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.





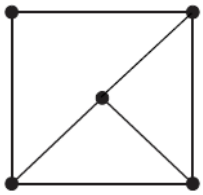
How do these solutions compare to the solutions using the Nearest Neighbor algorithm from our specified starting points?

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:

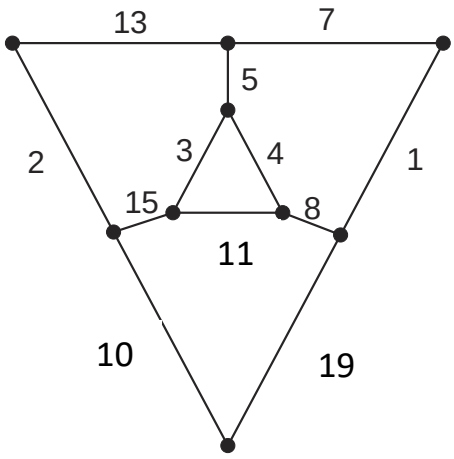


A ***minimal spanning tree*** is a spanning tree with the smallest possible weight.

Kruskal's Algorithm (for finding minimum-cost spanning trees):

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

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



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
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.

Which tasks, if any, are independent?

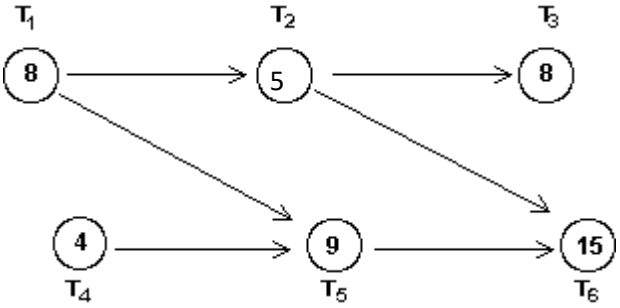
How long is required to make dinner if one person is available (no multi-tasking)?

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

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).

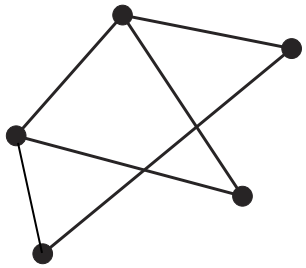
What is the critical path for making the soup? What is the earliest completion time?

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



SAMPLE EXAM QUESTIONS FROM CHAPTER 2

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



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.
(B) applying the nearest-neighbor algorithm for finding a Hamiltonian circuit.
(C) applying Kruskal's algorithm for finding a minimum-cost spanning tree for a graph.
(D) applying the method of trees for the traveling salesman problem.
(E) None of these/need more information

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.
(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

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

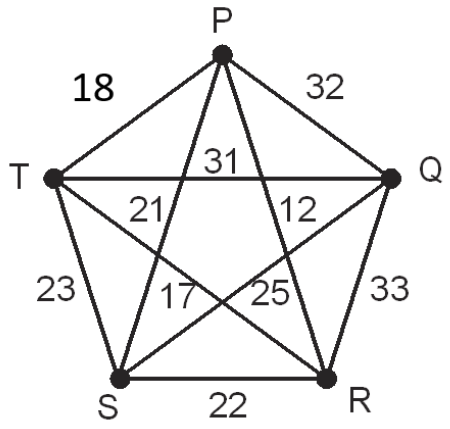
(B) An optimal result may be found

(C) An optimal result will not be found

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

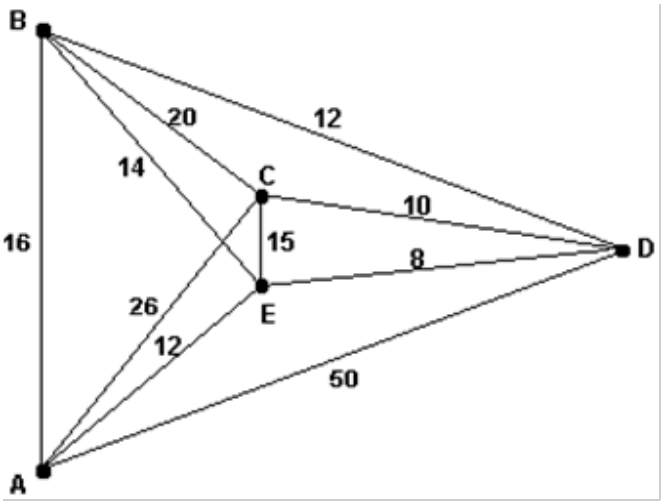
(E) The nearest neighbor method is an example of a heuristic algorithm.

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.



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

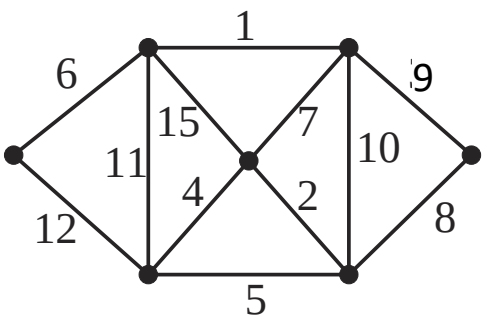
7. For the graph below, find the Hamiltonian circuit obtained by using the nearest-neighbor algorithm, starting at A. What is the cost if the numbers shown represent the distance in miles?



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?