

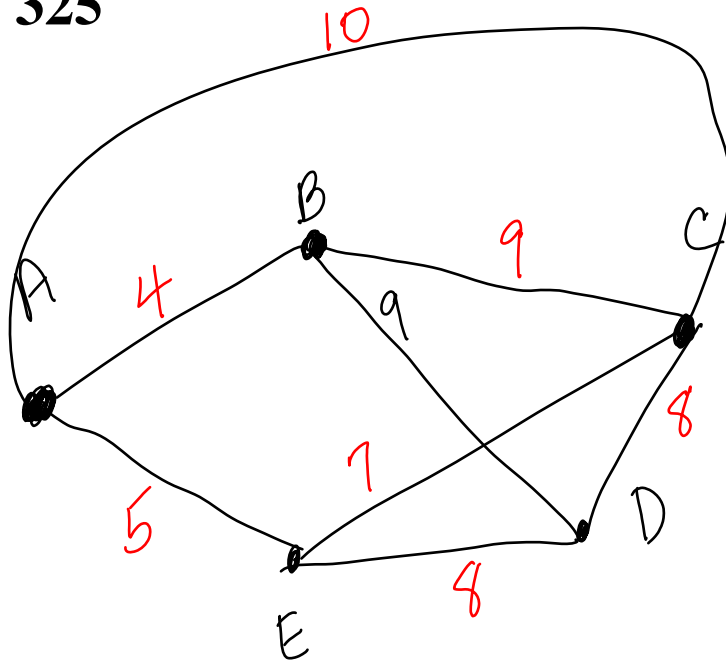
Weighted graphs

A weighted graph is a graph in which a numerical value is assigned to each edge of the graph. The weight can be time, \$, distance, etc.

There are two algorithms that the IB uses for finding a minimum spanning tree.

KRUSKAL'S ALGORITHM

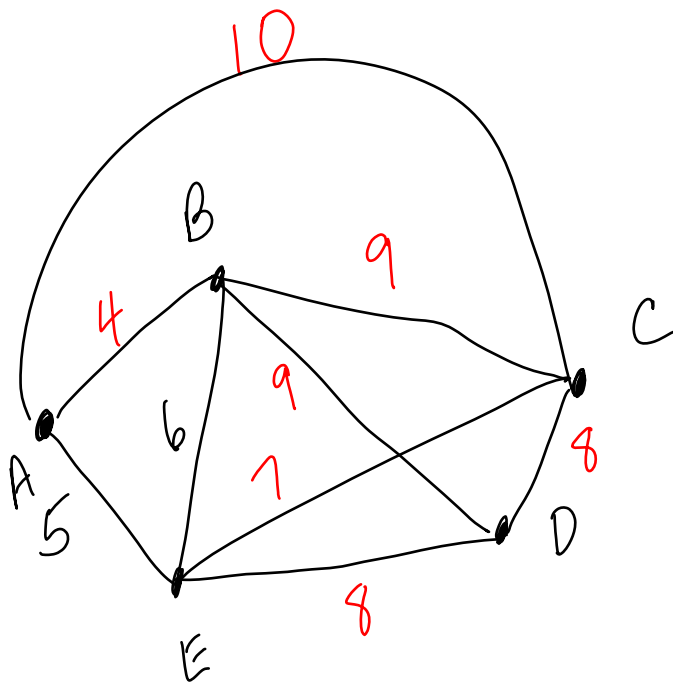
See page 325



1. Is it connected? If yes, then move on to the next step.
2. List the edges and their weights.

3. Start with the shortest edge. If there are several, choose one at random. Darken the first edge on the list.
4. Select the next edge on the list. If it does not form a cycle with the darkened edges, then darken it.
5. For a graph with n vertices, continue until $(n-1)$ edges have been darkened. That's your minimum edges.

$AB=4$
 $AE=5$
 $BE=6$
 $EC=7$
 $CD=8$
 $ED=8$



NO
 cycles

Is the minimum spanning tree unique?

Please see Example 38 on page 325.

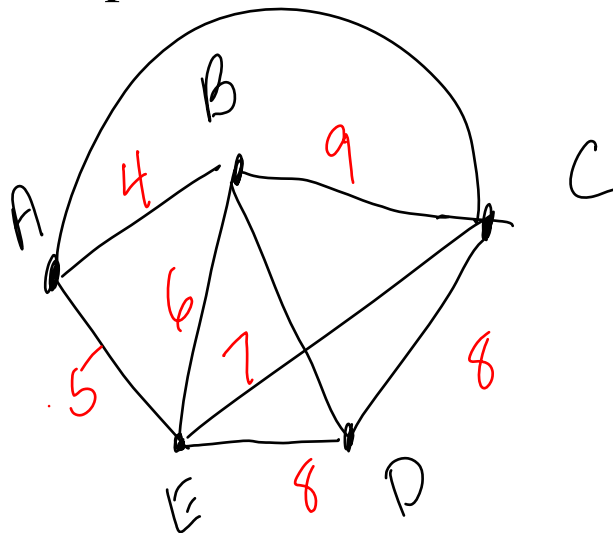
The IB wishes to see the work done on page 326 for complete credit.

Another minimum spanning tree algorithm is Prim's algorithm.

PRIM'S ALGORITHM

Please see bottom of page 326

1. Find the shortest edge of the graph. Darken it and circle its two vertices. Ties are broken arbitrarily.
2. Find the shortest remaining undarkened edge have one circled vertex and one uncircled vertex. Darken this edge and circle its uncircled vertex.
3. Repeat Step 2 until all vertices are circled.

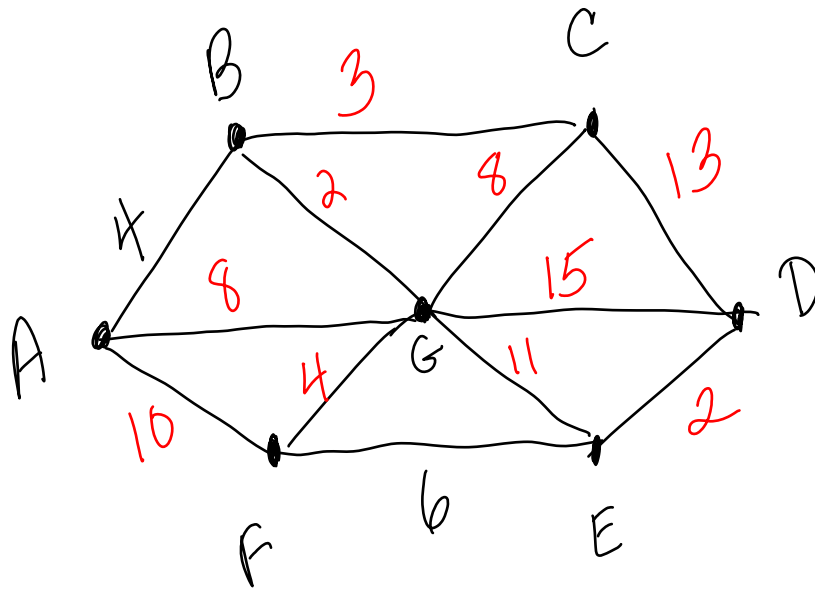


Please do problems 1 & 2 on page 328

Dijkstra's algorithm is used for finding minimum weight path between two given vertices.

DIJKSTRA'S ALGORITHM

1. Assign a value of 0 to the starting vertex. Draw a box [or a circle] around this vertex.
2. Consider all unboxed [uncircled] vertices adjacent to the latest boxed [circled] vertex. Label them with the minimum weight from the starting weight from the starting vertex via the set of boxed vertices.
3. Choose the least of ALL of the unboxed labels on the WHOLE graph, and make it permanent by boxing it.
4. Repeat steps 2 and 3 until the destination vertex has been boxed, then backtrack through the set of boxed vertices to find the shortest path through the graph.



START AT A END AT D

Please do pages 331 and 332 #1, 2, and 3

The Chinese Postman Problem

Finding the minimum weight Eulerian circuit of a weighted connected graph.

[What is the minimum weight closed walk that covers each EDGE at least once?]

If all of the vertices of the graph have even degrees, then the graph is Eulerian and there exists an Eulerian circuit that traverses every edge exactly once. BUT, if all of the vertices are not even, then you will have to travel some edge(s) twice. Our job is to minimize this distance.

We will have to walk twice over edges that are between pairs of odd vertices.

See examples 40 and 41 on page 333.

See example 42 on page 334

Do problems 1, 2 on page 335

TSP – The Traveling Salesman Problem

We must visit each vertex once [think of the vertices as clients]. We are actually finding a Hamiltonian circuit of the minimum total weight for a given graph.

In 1991 two mathematicians reported a solution to the classic TSP problem. [Brute force was commonly used – listing all possible routes!!!]

Classical TSP – visit each vertex only once
Practical TSP – allow vertices to be used on more than one occasion

Nearest Neighbor Algorithm

Caution: does not always find the optimal solution

Begin at your starting vertex and move to the nearest neighboring vertex, then to the next vertex not yet visited, and return to your starting vertex when all of the other vertices have been visited.

