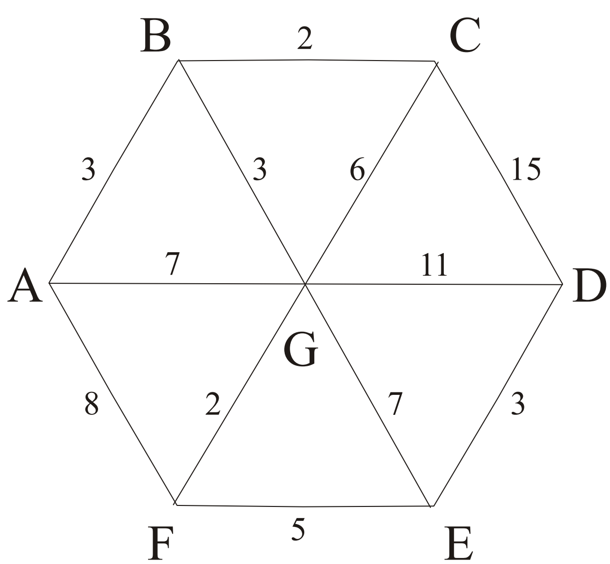
Dijkstra's algorithm and the shortest of path problem

Given a network (a weighted graph) there is a problem of finding the shortest path between any two vertices.

For example consider the following network.



It shows the distance in km between, let us imagine, seven towns. The question is , what is the shortest distance from *A* to *D*?

Several algorithms exist for the solution to the shortest path problem. One such algorithm is Dijkstra's algorithm.

Dijkstra's algorithm

Step 1 Assign a value 0 to the starting vertex and box this vertex.

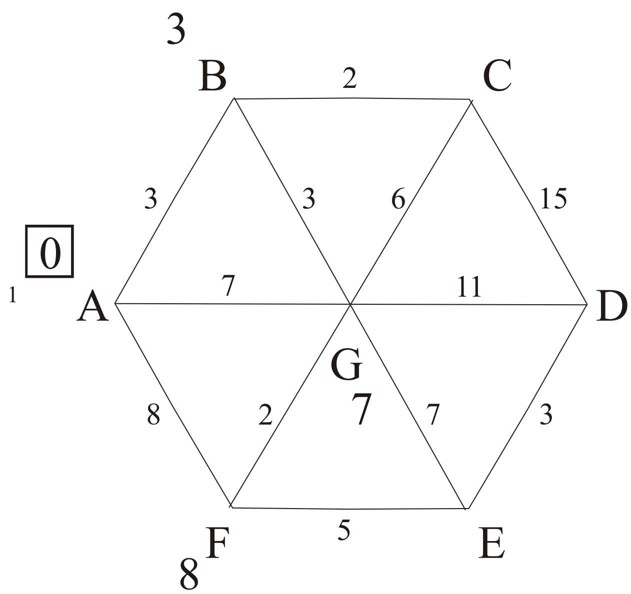
Step 2 For each vertex directly joined to the latest box vertex give a temporary label, which is the weight of the boxed vertex plus the weight of the edge joining the boxed vertex to that vertex.

Step 3 Select the least of all the temporary labels in the network and box it, thus making it permanent. Also assign to it a label indicating the order in which it was boxed.

Step 4 If the destination vertex is now boxed then stop; otherwise, go to step 2.

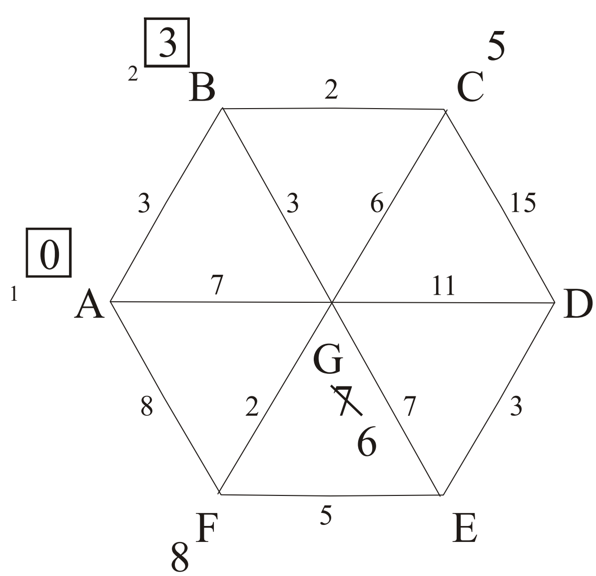
To illustrate this process on our existing graph

1st Iteration



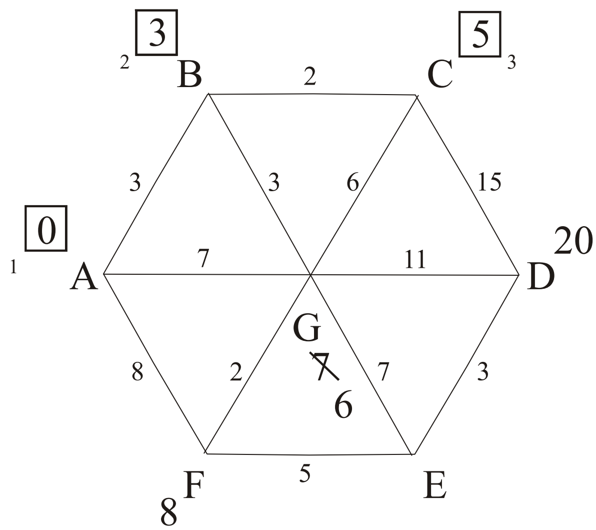
The label O is assigned to A and is boxed. A small 1 is also placed next to it to indicate that it was the first vertex to receive a permanent assignment. Temporary labels are assigned to vertices B, G, F corresponding to the weights of the edges joining A to them.

2nd Iteration



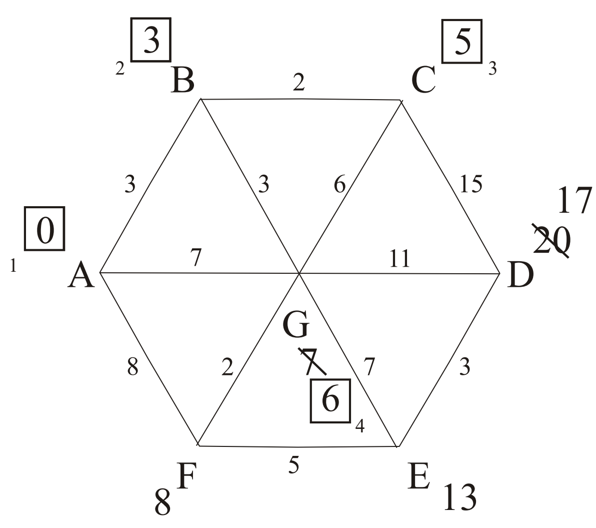
The least of the temporary labels is that of B which is boxed and marked with a subscript c indicating that it was the second in order . Temporary labels are assigned to vertices C and G. The original label for G is crossed out because the path A→B→G with weight 3+3=6 is shorter than the path A→ G with weight 7.

3rdIteration



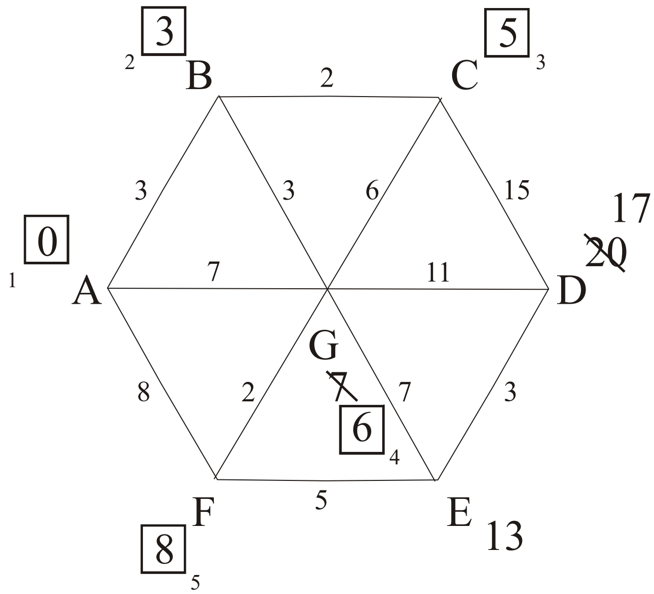
The least temporary label is that for C which is now boxed. A temporary label is assigned to D. No new label is given to G since the path A→B→C→G has greater weight than the path A→B→C which has the least weight of all paths to G.

4th Iteration



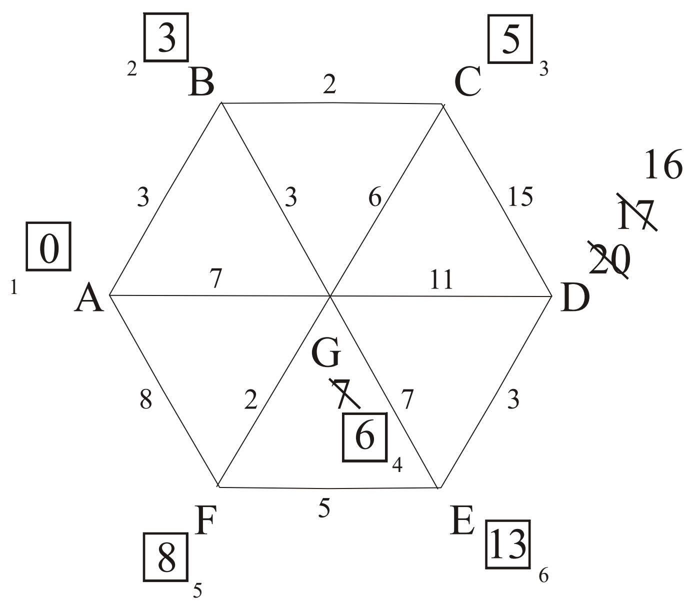
G is boxed and temporary labels given to E & D the temporary for F is not altered.

5th Iteration



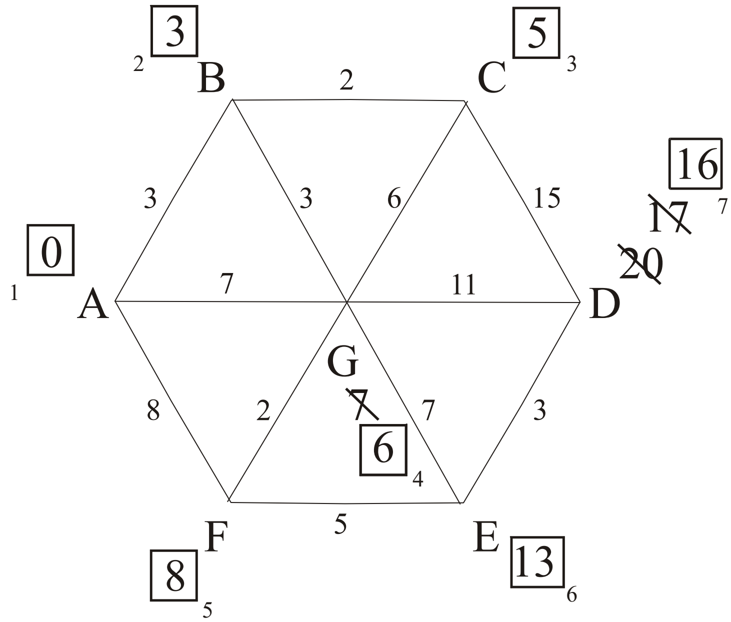
F is boxed but no temporary labels are altered.

<u>6[th] Iteration</u>



E is boxed, and the temporary label for D is altered.

<u>7[th] Iteration</u>



The destination vertex D is now reached and boxed. Working backwards the shortest path from A to D is

$$A \to B \to G \to E \to D$$

with weight $3 + 3 + 7 + 3 = 16$

Even with the assignment of the labels showing the order in which the vertices were boxed, the process of working backwards to find the shortest path can be an additional expense of time. One way to overcome this is to tabulate the results of the algorithm iteration by iteration.

Here

Iteration	$\boxed{A}$	B	C	D	E	F	G
1	0	$\boxed{3}$ (A)	-	-	-	8 (A)	7 (A)
2			$\boxed{5}$ (A)	-	-	8 (A)	6 (B)
3				20 (C)	-	8 (A)	$\boxed{6}$ (B)
4				17 (G)	13 (G)	$\boxed{8}$ (A)	
5				17 (G)	$\boxed{13}$ (G)		
6				$\boxed{16}$ (E)			
7				Finished			

We can now work backwards to find the path.

Shortest path $A \rightarrow B \rightarrow G \rightarrow E \rightarrow D$

Note : If negatively weighted edges are introduced into the network Djkstra's algorithm no longer works. In that case, another algorithm, called dynamic programming, is required.