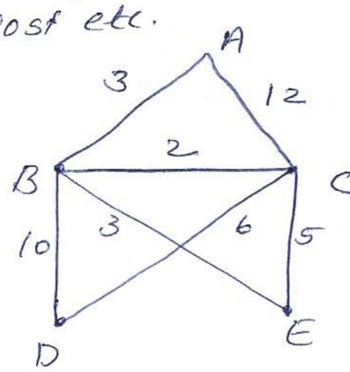
Weighted graph

A graph G is called a weighted graph, if each edge or vertex is assigned a data of one kind or another.

Each edge 'e' of G can be assigned a non-negative number called weight or length. The weights may represent distance, time, cost etc.



~~A graph G is called a weighted~~

A minimum path problem in a weighted graph is to find a path of minimum length b/w two vertices. Such a minimum path must be a simple path.

Prob- Find a minimum path α b/w A & D in the weighted graph G given above.

A → D
Path length

1. ABD - 13
2. ACD - 18
3. ACEBD - 12 + 5 + 3 + 10 = 30
4. ABCD - 3 + 2 + 6 = 11
5. ACBD - 12 + 2 + 10 = 24
6. ABECD - 3 + 3 + 5 + 6 - 17

Minimum length α b/w A & D is

ABCD = 11

Shortest Path Problems

A shortest path between two vertices in a weighted graph is a path of least weight. In an unweighted graph, a shortest path means one with least no. of edges.

Dijkstra's Algorithm

This algorithm is used to find the shortest path in a weighted graph.

To find the length (weight) of the shortest path b/w 2 vertices, say a and z, in a weighted graph, the algorithm assigns numerical labels to the vertices of the graph by an iterative procedure. At any stage of iteration, some vertices will have temporary labels (that are not bracketed) and the others will have permanent labels (that are bracketed). Let us denote the label of the vertex v by $L(v)$

Initial Iteration (0)

Let V_0 denote the set of all the vertices v_0 of the graph. The starting vertex is assigned the permanent label (0) and all other v_0's the temporary label ∞ each.

Let $V_1 = V_0 - \{v_0^*\}$ where v_0^* is the starting vertex which has been assigned a permanent label.

Iteration 1

Let the elements of V_1 be now denoted by v_1 (The elements v_1 are the same as the elements v_0 excluding v_0^*)

For the elements of V_1 that are adjacent to v_0^*, The temporary labels are revised by using $L(v_i) = L(v_0^*) + w(v_0^* v_i)$, where $L(v_0^*) = 0$, $w(v_0^* v_i)$ is the weight of the edge $v_0^* v_i$ and for the other elements of V_1, the previous temporary labels are not altered.

Let v_1^* be the vertex among the v_i's for which $L(v_i)$ is minimum.

If there is a tie for the choice of v_1^* it is broken arbitrarily. Now $L(v_1^*)$ is given a permanent label. Let $V_2 = V_1 - \{v_1^*\} \equiv \{v_2\}$

Iteration i

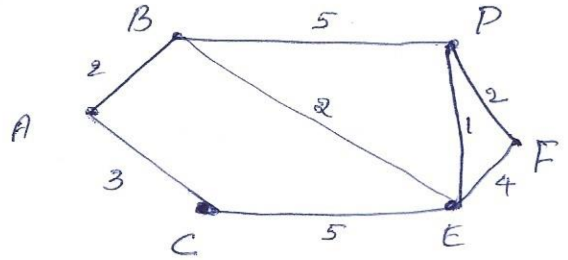
For the elements of V_i that are adjacent to v_{i-1}^*, the temporary labels are revised by using $L(v_i) = L(v_{i-1}^*) + w(v_{i-1}^* v_i)$ and for the other elements of V_i, the previous temporary labels are not altered. If the temporary label to be assigned to any vertex in the i^{th} iteration is greater than or equal to that assigned to it in the $(i-1)^{th}$ iteration, the previous label is not changed.

The iteration is stopped when the final vertex z is assigned a permanent label eventhough some vertices might not have been assigned permanent labels.

The permanent label of z is the length of the shortest path from a to z. The shortest path itself is identified by working backward from z and including those permanently labeled vertices from which the subsequent permanent labels arose.

Explanation of Dijkstra's Algorithm by an example

Prob: Find the shortest path from vertex A to the vertex F for the following weighted graph.



Iteration no.	Iteration Details.	Remarks
0	V_0 : A B C D E F $L(V_0)$ (0) ∞ ∞ ∞ ∞ ∞	Initial labels for all the vertices are assumed. A gets permanent label and $L(A^*) = 0$ is bracketed.
1.	V_1: A^* B C D E F $L(V_1)$: – (2) 3 ∞ ∞ ∞	B & C are adjacent vertices of A^*. $L(B) = L(A^*) + W(A^*B) = 0 + 2 = 2$ $L(C) = L(A^*) + W(A^*C) = 0 + 3 = 3$ Since $L(B) < L(C)$ B gets permanent label and $L(B^*) = 2$ is bracketed.
2.	V_2: $A^* B^*$ C D E F $L(V_2)$: – – (3) 7 4 ∞	D & E are adjacent vertices to B^*. $L(D) = L(B^*) + W(B^*D) = 2 + 5 = 7$ $L(E) = L(B^*) + W(B^*E) = 2 + 2 = 4$ Since C is not adjacent to B^*, $L(C)$ is brought forward from the previous iteration as 3 Since $L(C)$ is minimum along $L(C)$, $L(D)$ & $L(E)$, C gets permanent label and $L(C^*) = 3$ is bracketed

3. V_3: $A^* B^* C^*$ D E F
— — — 7 (4) ∞

D and F are not adjacent to C^*. So $L(D)$ and $L(F)$ are brought forward from iteration (2)

$L(E) = L(C^*) + W(C^*E) = 3+5 = 8$

Since the revised $L(E) >$ the previous $L(E)$, the previous value of $L(E) = 4$ is retained. Now E gets permanent label and $L(E^*) = 4$ is bracketed.

4. V_4: $A^* B^* C^* D E^* F$
— — — (5) — 8

D & F are adjacent to E^*

$L(D) = L(E^*) + W(E^*D) = 4+1 = 5$

$L(F) = L(E^*) + W(E^*F) = 4+4 = 8$

Since $L(D) < L(F)$, D gets the permanent label and $L(D^*) = 5$ is bracketed.

5. V_5: $A^* B^* C^* D^* E^* F$
— — — — — (7)

Since F is the only vertex adjacent to D^* and

Since $L(F) = L(D^*) + W(D^*F)$

$= 5 + 2 = 7$, the final vertex F gets the permanent label and

$L(F^*) = 7$ is bracketed.

Since $L(F^*) = 7$, the length of the shortest path from A to F $= 7$

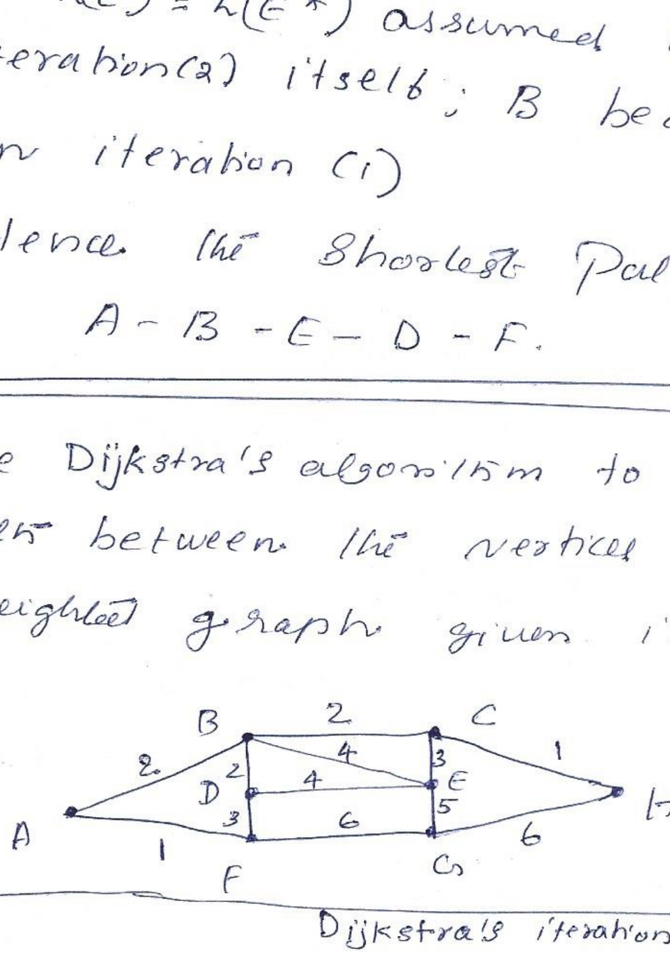
To find the shortest path, we work backwards from F explained as follows.

F became F* from D* in iteration (5)
D became D* from E* in iteration(4);
E became E* from B (but not from G) as L(E) = L(E*) assumed the label 4 in iteration(2) itself; B became B* from A* in iteration (1)

Hence the shortest path is,

A - B - E - D - F.

Prob: 2 Use Dijkstra's algorithm to find the shortest path between the vertices A and H in the weighted graph given in the following figure.



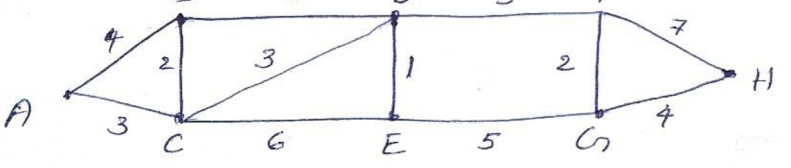
Dijkstra's iteration

Number	Details of V and L(v)									Adjacent Vertices of Latest V*
0	V_0 :	A	B	C	D	E	F	G	H	B and F
	$L(V_0)$:	(0)	∞	∞	∞	∞	∞	∞	∞	
1	V_1 :	A*	B	C	D	E	F	G	H	D and G
	$L(V_1)$:	—	2	∞	∞	∞	(1)	∞	∞	
2	V_2 :	A*	B	C	D	E	F*	G	H	C D and E
	$L(V_2)$:	—	(2)	∞	4	∞	—	7	∞	
3	V_3 :	A*	B*	C	D	E	F*	G	H	E and H
	$L(V_3)$:	—	—	(4)	4	6	—	∞	∞	
4.	V_4 :	A*	B*	C*	D	E	F*	G	H	—
	$L(V_4)$:	—	—	—	∞	7	—	∞	(5)	

Since H is reached from G, C is reached from B and B is reached from A, the shortest path is, A - B - C - H.

Length of shortest path $= W(AB) + W(BC) + W(CH)$
$= 2 + 2 + 1 = \underline{5}$

Pro: 3
H.W. Use Dijkstra's algorithm to find the shortest path b/w the indicated vertices in the weighted graph given below.



(between A & H)

Note.
In the complete graph K_n of n vertices there are $\frac{(n-1)!}{2}$ different Hamiltonian circuits.

Travelling Salesman Problem.

Suppose a travelling salesman's territory includes several towns with roads connecting certain pairs of these towns. His job is to visit all the towns. Is it possible ~~so as~~ to visit each town exactly once and return to the starting town. If such a trip is possible then can he plan a trip which minimises the total distance travelled?

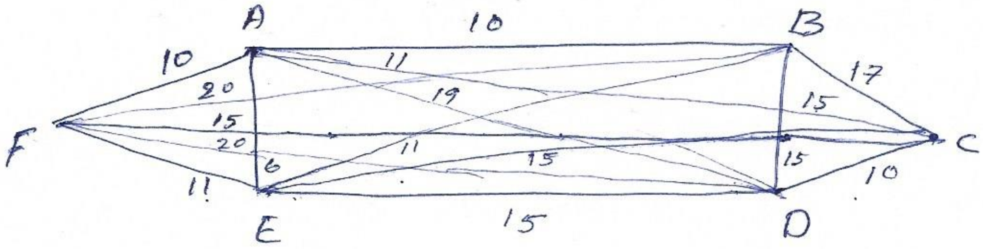
We can represent the salesman's territory by a weighted graph G, where the vertices corresponds to towns and 2 vertices are joined by a weighted edge if and only if there is a road connecting the corresponding towns, which does not pass through any other towns. Weight of the edge represents length of the road b/w the towns. Now the problem becomes a graphical problem.

Is the graph G, a Hamiltonian graph?

If so can we construct a Hamiltonian circuit with minimum weights?

If G is a complete graph with n vertices, then there are $\frac{(n-1)!}{2}$ different Hamiltonian circuits in G. Theoretically the problem of Travelling salesman can be solved by enumerating all the $\frac{(n-1)!}{2}$ Hamiltonian circuits, calculating the distance travelled in each and then finding the minimum distance, but for large values of n the labour involved is too great even for a digital computer. An efficient algorithm for the problem has yet to be found. However there are several heuristic methods suitable to find a route very close to the shortest one.

Prob. Find a Hamiltonian circuit of minimum weight for the weighted graph given below.



From	A	B	C	D	E	F
A	–	10	11	19	6	10
B	10	–	17	15	11	20
C	11	17	–	10	15	15
D	19	15	10	–	15	20
E	6	11	15	15	–	11
F	10	20	15	20	11	–

Write the weights of the edges in an $n \times n$ (6x6) table. Select the smallest value in the 1st column which is 6. So start from E & go to A. Select the smallest value in the row corresponding to A which is 10. Go to B and then select the smallest value in the row B which is 15. Go to D then select the smallest value in the row D which is 10. Then go to C then select the smallest value in row C which is 15. Go to F. Proceeding like this, we get the

Hamiltonian circuit EABDCFE

Whose length is $6+10+15+10+15+11=67$

This is the Hamiltonian circuit with minimum length 67

Another method is using nearest neighbour method

Select vertex A. The nearest neighbours is E (because AE is the edge with least weight) Then go to the nearest neighbours of E which is B. Goto D. Then go to C, then F & return back to A.

The Hamiltonian circuit obtained is AEBDCFA whose weight is 67.

This is also a Hamiltonian circuit with minimum weight (length) 67