Q1) As in quick sort, we have to split array in 2 halves, and then in halves of a half, but this time, we only need to do the next round partition in one single partition (half) of the two where the element is expected to lie in.
which is approximately

n + (1/2)*n + (1/4)* n + (1/8)*n + ... < 2*n

Which is O(n).

Q2)

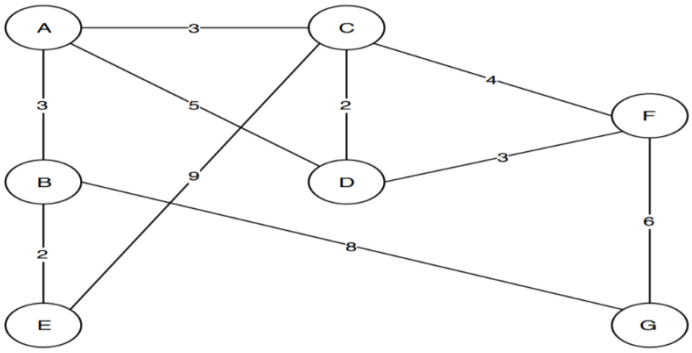


Figure 1: An undirected weighted graph.

Djikstra Algorithm:

i. Known vertexes: -
   Unknown vertexes: A (-, ∞) B (-, ∞) C (-, ∞) D (-, ∞) E (-, ∞) F (-, ∞) G (-, ∞) H (-, ∞)
ii. Known vertexes: E(E,0)
   Unknown vertexes: A(-, ∞) B(E, 2) C(E,9) D(-,∞) F(-,∞) G(-,∞) H(-,∞)
iii. Known vertexes: E(E,0) B(E,2)
   Unknown vertexes: A(B,5) C(E,9) D(-,∞) F(-,∞) G(B,10) H(-,∞)
iv. Known vertexes: E(E,0) B(E,1) A(B,5)
   Unknown vertexes: C(A,8) D(A,10) G(B,10) H(-,∞)
v. Known vertexes: E(E,0) B(E,1) A(B,5) C(A,8)
   Unknown vertexes: D(A,10) F(C,12) G(B,10) H(-,∞)
vi. Known vertexes: E(E,0) B(E,1) A(B,5) C(A,8) D(A,10)
   Unknown vertexes: F(C,12) G(B,10)
vii. Known vertexes: E(E,0) B(E,1) A(B,5) C(A,8) G(B,10)
   Unknown vertexes: F(C,12)

viii. Known vertexes: E(E,0) B(E,1) A(B,5) C(A,8) G(B,10) F(C,12)
Unknown vertexes: -

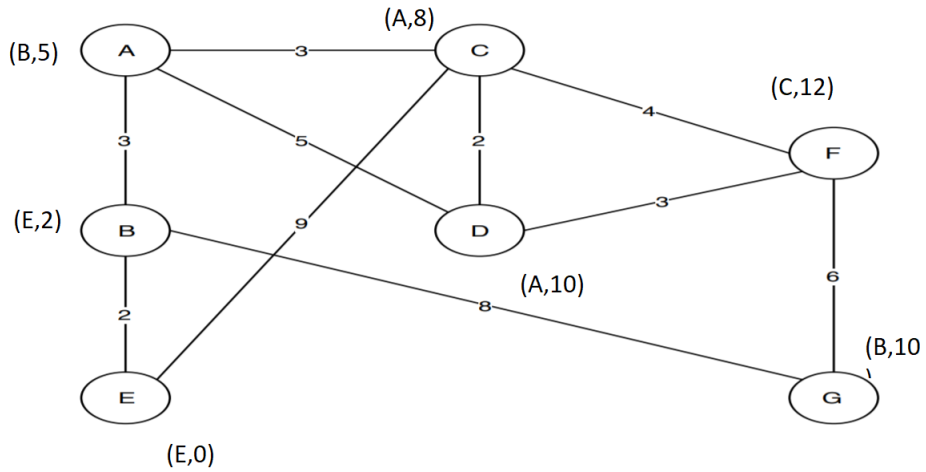


Figure 1: An undirected weighted graph.

Q3)

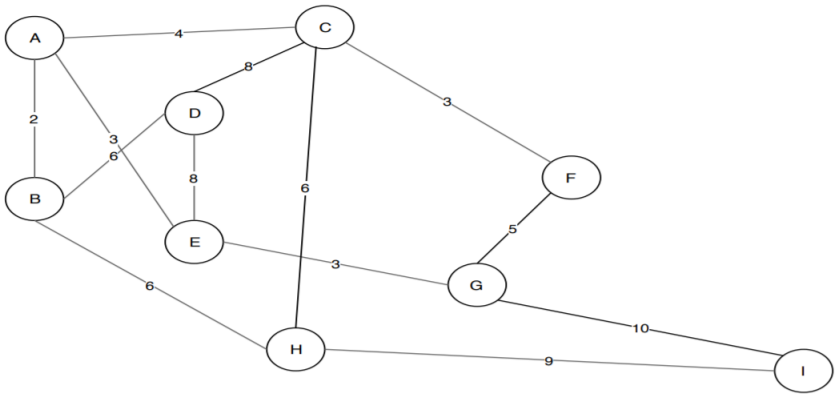


Figure 2: An undirected weighted graph.

Vertices included at each step = V

We start with the given node E add it to the vertices and then compare the adjacent edges which include EG(3) AE(3) to find the smallest of them and choose the end node of that edge which in this case is G because of EG AND AE have same weight and this process is repeated with each node along with the condition of avoiding cycles in the new tree (avoiding edges which might produces a cycle) in the end we get the following vertices:

I. V = (E)
II. V = (E,G) (*EG weight=3)
III. V = (E,G,A) (*EA weight=3 compared to GF(5),GI(10))
IV. V= (E,G,A,B)
V. V= (E,G,A,B,C)

VI. V=(E,G,A,B,C,F)

VII. V=(E,G,A,B,C,F,D)

VIII. V=(E,G,A,B,C,F,D,H)

IX. V=(E,G,A,B,C,F,D,H,I) **(ALL NODES HAVE BEEN VISITED ONCE WE ARE DONE NOW)

Q4)

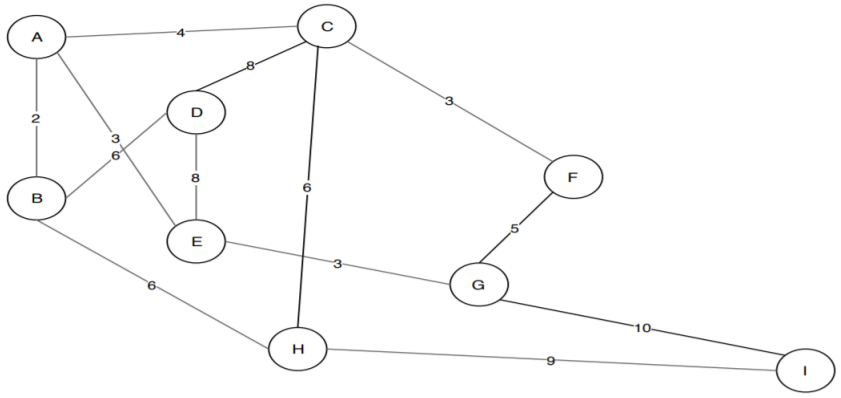


Figure 2: An undirected weighted graph.

We add the edges with smallest weight to trees and then union them if required while avoiding cycles.

Through this process we add the following edges:

I.	AB	TREE 1 = A,B	
II.	AE	TREE 1 = A,B,E	
III.	CF	TREE 1 = A,B,E	TREE 2 = C,F
IV.	EG	TREE 1 = A,B,E,G	TREE 2 = C,F
V.	AC	TREE 1 = A,B,E,G,C,F	
VI.	BD	TREE 1 = A,B,E,G,C,F,D	
VII.	BH	TREE 1 = A,B,E,G,C,F,D,H	
VIII.	HI	TREE 1 = A,B,E,G,C,F,D,H,I	

Q5)

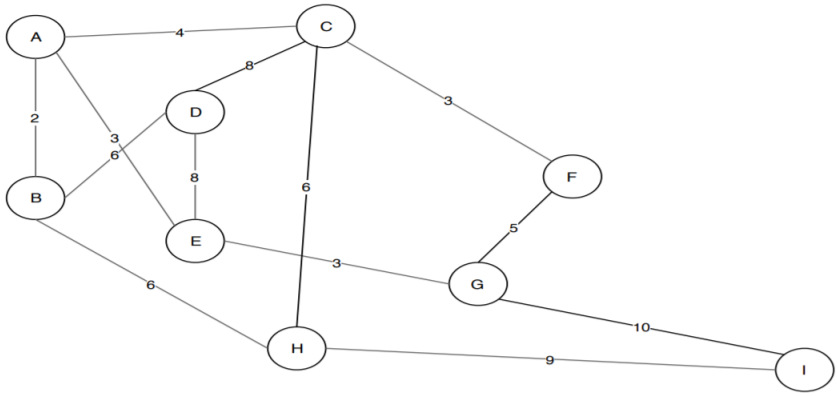
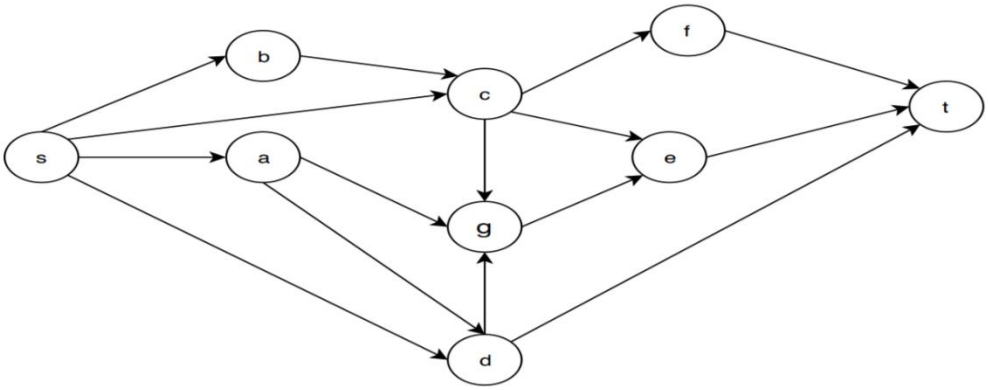


Figure 2: An undirected weighted graph.

Breadth First Search:

i. Visit G
ii. Visit unvisited children of G (E F I)
iii. Visit unvisited children of E (A D)
iv. Visit unvisited children of F (C)
v. Visit unvisited children of I (H)
vi. Visit unvisited children of A (B)
vii. Visit unvisited children of D (-)
viii. Visit unvisited children of C (-)
ix. Visit unvisited children of H (-)
x. Visit unvisited children of B (-)



Q6)

TOPOLOGICAL SORT OF ABOVE FIG:

S B A C D G F E T