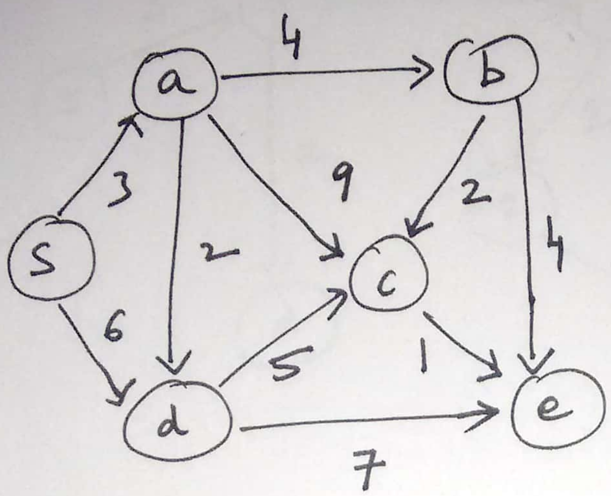
CS 6363 - ASSIGNMENT 4 - PART ONE

PROBLEM-1:

(a). DIJKSTRA'S ALGORITHM:



Source: s.

Order of visited vertices:
s, a, d, b, c, e.

Distance table:

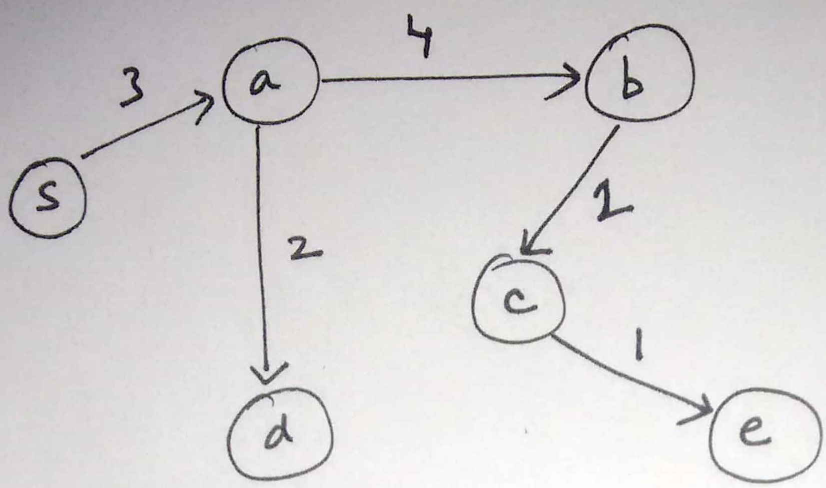
Active vertex in each round.

	Initial	s	a	d	b	c	e
s	0	0	0	0	0	0	0
a	∞	3	3	3	3	3	3
b	∞	∞	7	7	7	7	7
c	∞	∞	12	10	9	9	9
d	∞	6	5	5	5	5	5
e	∞	∞	∞	12	11	10	10

Parent Table:

	Initially	s	a	d	b	c
s	NULL	NULL	NULL	NULL	ϕ	ϕ
a	–	s	s	s	s	s
b	–	–	a	a	a	a
c	–	–	a	d	b	b
d	–	s	a	a	a	a
e	–	–	–	e	b	c

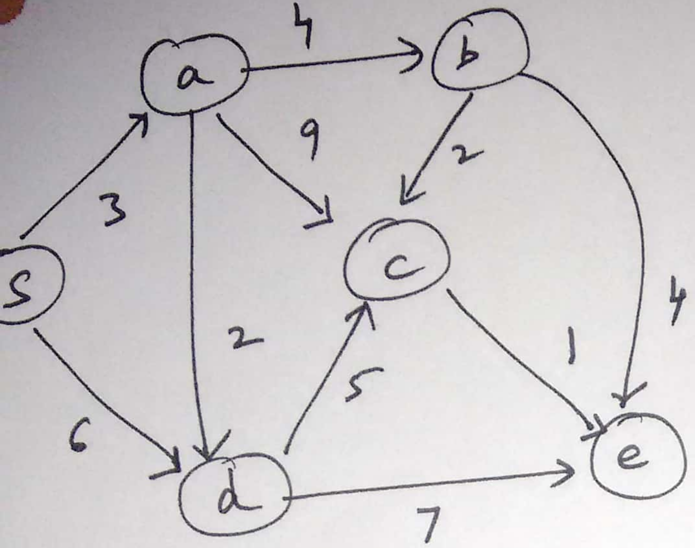
Shortest path Tree:



From, s
Shortest distance to:

a = 3
b = 7
c = 9
d = 5
e = 10.

(b). Bellman Ford Algorithm.



Visiting Order of the edges:

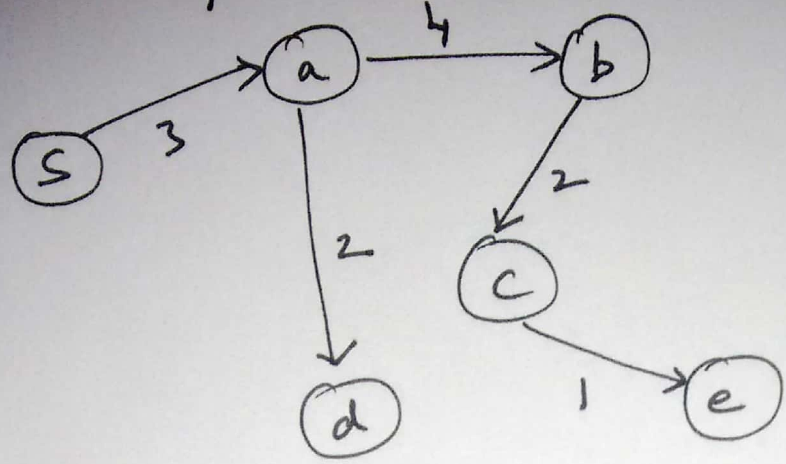
S → a
S → d
a → b
a → c
a → d
d → c
d → e
b → c
b → e
c → e

Distance Table:

Vertex	Initial	Round 1	Round 2 ...	Round 5
S	0	0	0	0
a	∞	3	3	3
b	∞	7	7	7
c	∞	~~12~~ ~~10~~ 9	9	9
d	∞	~~6~~ 5	5	5
e	∞	~~12~~ ~~11~~ 10	10	10

Vertex	Initially Parent	Round 1	. . . .	Round 5
S	Φ	Φ		Φ
a	—	S		S
b	—	a		a
c	—	b		b
d	—	a		a
e	—	c		c

Shortest path tree:



PROBLEM-2 :

```
Algorithm CheckSPT (s,V,E, pred[1...n], dist[1...n])
{
  if (pred(s) ≠ NULL or dist[s] ≠ 0)
      return false
  for all (u,v) ∈ E
      if dist[u] + weight(u,v) < dist[v]
          return false
      else if dist[u] + weight(u,v) = dist[v]
          if pred(v) ≠ u
              return false
  return true
}
```

Explanation :

1) We simply loop through all edges in the input graph and check if there is a tense edge.

2) Since all the edges have positive weights, there cannot be a negative weight cycle in the graph.

3) Hence, the shortest path tree must not contain tense edges.

Running time :

1) Base case checks for pred(s) = NULL and dist[s] = 0. This takes O(1) time.

2) Looping over all edges takes linear time.

i.e. $O(|V|+|E|)$

3) The comparisons inside the for loop take $O(1)$ time.

$\boxed{\therefore T(n) = O(|V|+|E|)}$

Also, for every edge which is in the shortest path tree, we check if:

$$pred(v) = u$$

where (u, v) is the edge in the SPT.

PROBLEM-3 :

Algorithm BestEdge (V, E, F, s, t)

```
{
    p = size(E\F)
    init array A[1...p]
    init array B[1...p]
    for all edges (u,v) ∈ (E\F)
        put u in A
        put v in B.

    Run Dijkstra(s) on H=(V,F) and update the distances
    of the vertices in A

    Reverse the direction of the edges in
    H.

    Run Dijkstra(t) on H and update
    the distances of the vertices in B.
    min = ∞, edge = NULL
    for i=1 to p
        if ( dist(A[i]) + weight(A[i],B[i]) +
                    dist(B[i]) ) < min
            min = dist(A[i]) + weight(A[i],B[i]) + dist(B[i])
            edge = (A[i], B[i])

    return edge
}
```

Explanation:

We first initialize an array for all the starting points of the <u>edges removed</u> and an array for the ending points of these same edges.

2) We traverse through these edges and put the start and end points in arrays A and B respectively.

3) We run Dijkstra with source s and in the process update the distances of all vertices in A.

4) We reverse the edges in $H=(V,F)$ and run Dijkstra with source t. We update the distances of all vertices in B.

5) The best edge $e=(A[i],B[i])$ is the one with minimum value of:

$$dist_s[A[i]] + dist_t[B[i]] + weight(A[i], B[i])$$

where subscript denotes the vertex which is taken as the source for computing that distance using Dijkstra.

<u>Running time:</u>

Looping over edges removed: $O(|E|)$

Running Dijkstra from S: $O(|E| \log |V|)$ if we use binary heap.

Reversing graph: $O(|V|+|E|)$

Running Dijkstra from t: $O(|E| \log |V|)$

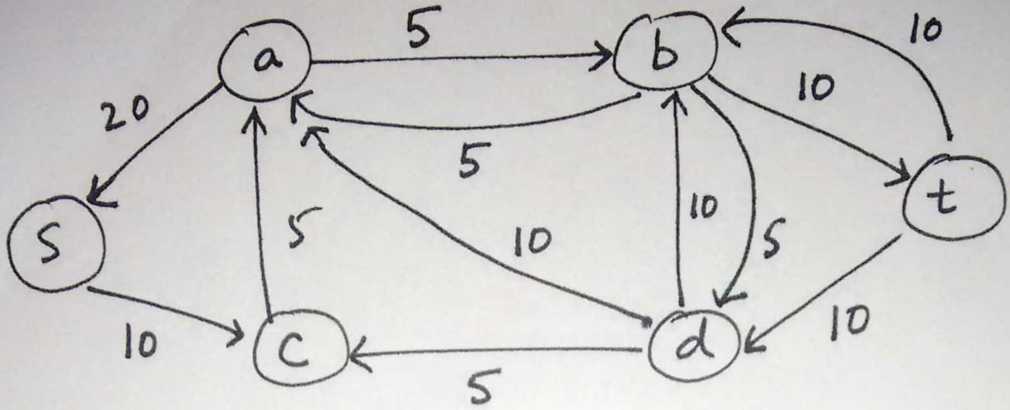
Looping over edges removed: $O(|E|)$

$$\therefore T(n) = O(|E|) + O(|E| \log |V|) + O(|V| + |E|) + O(|E| \log |V|) + O(|E|)$$

$$T(n) = O(|E| \log |V|)$$

PROBLEM-4:

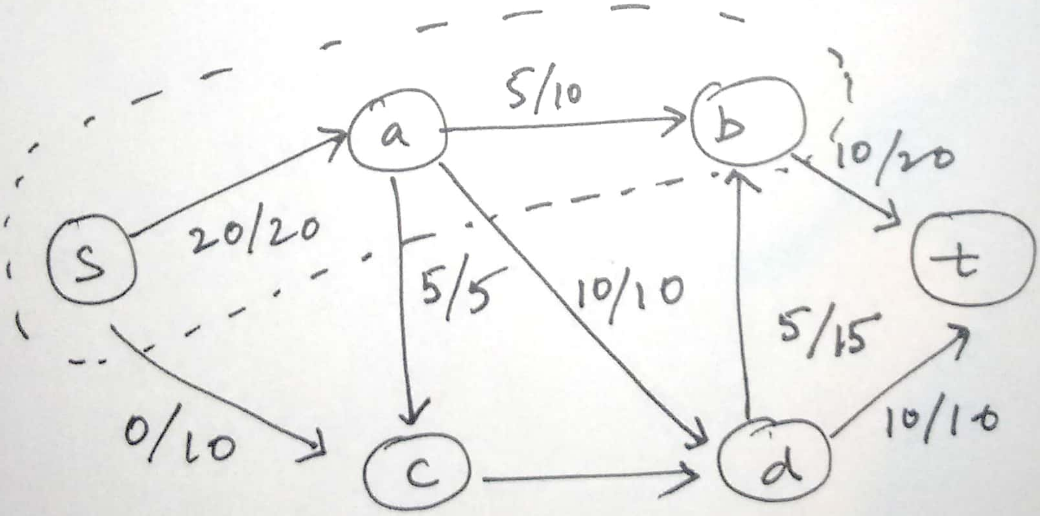
a). Residual graph.



(b). Assuming s is the source and t is the destination, an example of augmenting path:

$s \to c \to a \to b \to t$

(c)



$\{s, a, b\}$ and $\{c, d, t\}$

Capacity $= 10 + 5 + 10 + 20 = 45$