Assignment 4

Question 1. Show how to compute the shortest paths between all pairs of vertices in the graph attached using the Floyd-Warshall algorithm. It is enough to give the D matrix at the end of each iteration.

```
procedure floydWarshall (graph)
  // Initialize a matrix to store the shortest distances
  let matrix be a |V| x |V| array of integers
  for each vertex v in graph
    for each vertex u in graph
      matrix[v][u] = graph[v][u]
  end for

  // Find the shortest paths between all pairs of vertices
  for each vertex k in graph
    for each vertex i in graph
      for each vertex j in graph
        if matrix[i][k] + matrix[k][j] < matrix[i][j]
          matrix[i][j] = matrix[i][k] + matrix[k][j]
        end if
      end for
    end for
  end for

  // Print the resulting matrix with shortest distances
  printMatrix(matrix)
end procedure
```

D1:

$$\begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & 6 & 0 & \infty \end{bmatrix}$$

D2:

$$\begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

D3:

$$\begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -2 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

D4:

$$\begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

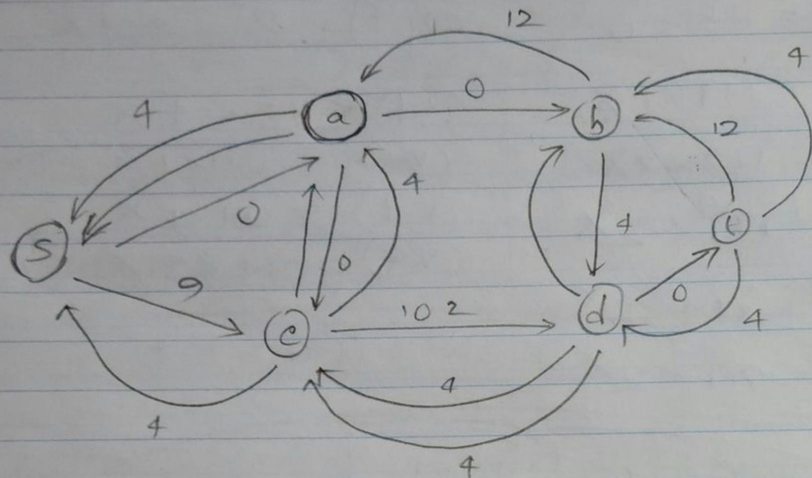
D5:

$$\begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

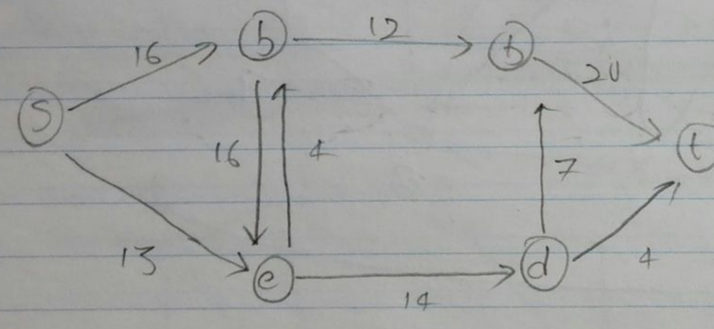
Question 2

How finding min. edge from ie 4 (c→a) The value will

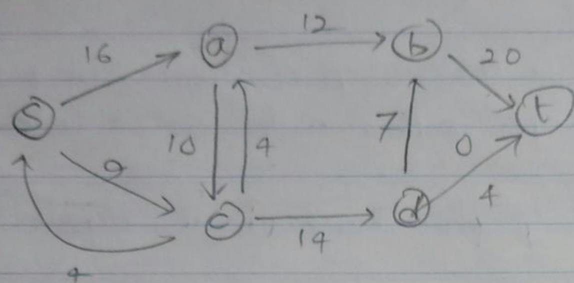
return the flow:



Now we get 4+12+4 = 20 The total flow is 20



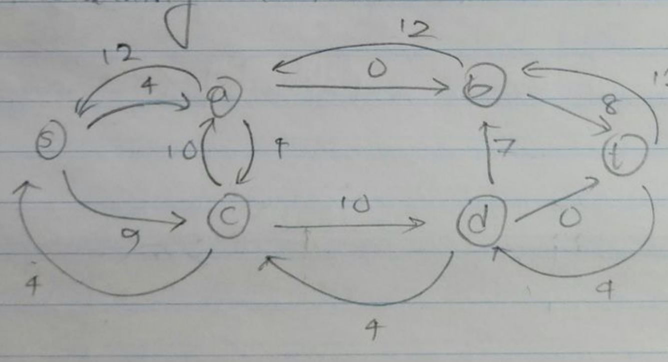
step 1: Finding shortest path using BFS and BFS s→c→d→t



Again use BFS, The path will be

s → a → b → t

Taking the minimum edge = 12 i.e a → b and returning the flow



Question 3

To prove that a given flow f is maximum in a graph G, we can use the concept of augmented paths. An augmented path is a simple path from the source to the sink in the ~~res~~ residual graph (the graph that represents the remaining capacity after the current flow is applied). If there exists an augmented path, it means we can increase the flow along that path and get a larger flow, indicating that current flow is not maximal. If no augmenting path exists, then the current flow is indeed maximal.

~~Initialize the residual graph G' based on the current~~

Here's the algorithm to prove that the flow is maximal which runs

in $O(m)$ time complexity.

1. Initialize the residual graph G' based on the current

flow f and capacities of edges in G.

For edges (u,v) in G, create two edges in G' in (u,v)
with capacities $c(u,v) - f(u,v)$ and (v,u) with capacity $f(u,v)$
Set the capacity of each edge in G' to the remaining
capacity (capacity-flow) in the original graph

2. Perform a DFS of the residual graph G' starting
from the source node s to the sink node t, looking for
an augmented path

3. If an augmented path is found update the flow f
along the path and return to step 1

4. If no augmented path is found, terminate the algorithm

and the current flow is taken to be the maximal.

Constructing the residual graph takes $O(m)$ but each iterations required
to find the maximal flow also takes $O(m)$. So the running time is
$O(m^2)$

Hilroy

4

The edge connectivity of a graph G is the minimum number of edges that need to be deleted, such that the graph G gets disconnected. So, here we are using algorithm in Ford Fullerson theorem with Edmonds-Karps implementation. We iterated over all pairs of vertices and with each flow find ~~longest~~ largest number of disjoint paths. So, we find min-max flow ~~every~~ every two nodes of the graph.

① For each edge (u,v) in E,

ⓐ create a new graph G' by removing the edges
ⓑ Add a new source node S and a new sink node t to G.
ⓒ connect s to u and v to t with infinity capacity edges.

② Run a max flow algorithm on the modified graph

③ The minimum cut value in the max flow algorithm will give us the edge connectivity of the original graph G.

```
int edge (Graph G) {
    int min = infinity
    for (Vertex V : G.m) {
        if (V != m) and (min > max flow) {
            min = max flow;
        }
    }
```

Time Complexity: $O(V^3E^2)$