COMP 3711 Design and Analysis of Algorithms

2023 Fall Semester - Homework 4 solution

Question 1: (a) **Solution 1:**

The sum of vertex degrees is twice the number of edges. We have n edges so the sum of vertex degrees is $2n$. By assumption, every vertex has degree 2 or above. Therefore, if some vertex has degree three or above, the sum of vertex degrees is at least 2n + 1, which is a contradiction.

Solution 2:

When we have n edges, it means the sum of vertex degrees is equal to $2n$. Now if every vertex has a degree of 2 (which is the claim itself) we are fine. We Also know there are no vertex with degree 0 or 1. Now consider there exists at least one vertex with degree more than 2. This means there must exists a vertex with degree less than 2 (As the average degree for each vertex is 2) which is a contradiction. Notice that 2 can be seen as the "Average" degree of vertex so it is obvious if there exists a vertex with higher degree than 2, there must exists a vertex with lower degree to maintain the same average.

(b) **Solution 1:**

Suppose that there are $n \geq 2$ vertices. There are two cases. Suppose that every vertex has degree at least 1. If every vertex has a different degree, the sum of vertex degrees is at least $1 + 2 + 3 + \ + n = n(n + 1)/2$. But this is greater than the number of possible edges which is $n(n - 1)/2$, a contradiction. The remaining case is that some vertex has degree 0. If every vertex has a different degree, the other $n - 1$ vertices must have degree at least 1. Using the previous argument, the sum of vertex degrees of these $n-1$ vertices is at least $1+2+ +n-1 = n(n-1)/2$. But this is greater than the number of possible edges among these $n - 1$ vertices which is $(n - 1)(n - 2)/2$, a contradiction.

Solution 2:

If there are n vertices, the possible values for degrees are $\{0, 1, 2, \cdots, n - 1\}$. There are n values here. However, degrees 0 and $n-1$ can not happen at the same time. Because 0 means there exists a vertex that is not connected to any other vertex and $n - 1$ means there exists a vertex that is connected to every other vertex. So there are $n-1$ possible values for the degrees and there are n vertices. Based on Pigeonhole Principle, at least there are two vertices with the same degree. This proves that it is not possible for every vertex to have a different degree.

(c) **Solution 1:**

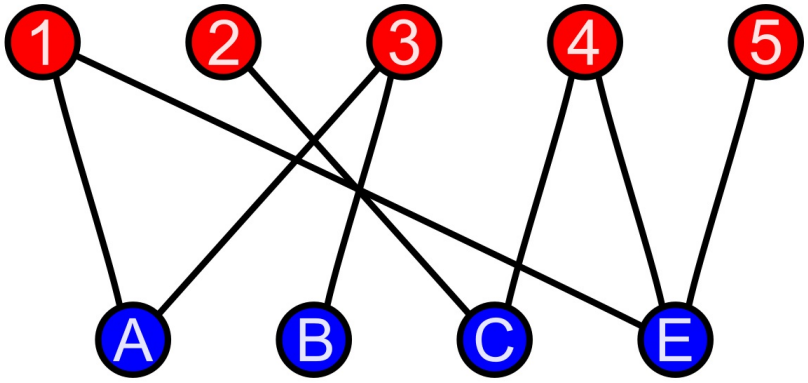
Label the 10 people by v_i for $i \in [1, 10]$. Let G be an undirected graph with vertices $v_1, v_2, \cdots, v_{10}$ such that two vertices v_i and v_j are connected by an edge if and only if v_i and v_j are friends. For every v_i, let N_i denote the set of vertices that are neighbors of v_i. Without loss of generality, assume that $N_1 = v_2, v_3, \cdots, v_8$. Since $|N_2| = 7$ by assumption, we have $|N_1 \cap N_2| \geq 4$ because v_9 and v_{10} may belong to N_2. Note that $v_1 \in N_2$ but $v_1 \notin N_1$. Without loss of generality, assume that $v_3 \in N_1 \cap N_2$. We have $|N3| = 7$ by assumption. It is possible that v_9 and v_{10} belong to N_3. It is also possible that N_1 N_2 is a subset of N_3, but we know that $|N_1 \setminus N_2| \leq 3$ as $|N_1| = 7$ and $|N_1 \cap N_2| \geq 4$. As a result $|N_1 \setminus (v_2 \cup N_2)| \leq 2$. It follows that if we discount v_9, v_{10}, v_1, v_2, and $N_1 \setminus (v_2 \cup N_2)$ from N_3, we still have at least 1 person left. This person must belong to $N_1 \cap N_2 \cap N_3$. This person must form the desired subset of 4 people with v_1, v_2 and v_3.

Solution 2:

Consider an edge between two people means friendship between them. Every node is connected to 7 other nodes. Consider two nodes A, B that are already friend of each other. Out of the remaining 8 people, A, B are each connected to 6 of them. So based on pigeonhole principle, They have at least 4 mutual friends. Consider this 4 mutual friends to be x_1, x_2, x_3, x_4. If there exists any edge between any pair of them we are done because we will have 4 people who are pairwise friend of each other. So there must be no edge among any pair of them. Now consider any of those nodes. It has to be connected to 7 nodes. However, Now maximum degree of that node is 6 as it can not be connected to 3 nodes (and itself), which is a contradiction. So it means there exists 4 people that are pairwise friend of each other.

Question 2: We first explain the idea behind this question and then provide the psudo-code and analysis of it. In the figure below you can see a bipartite graph. In this graph nodes can be split into two groups such that the nodes of each group does not have any edge among them. So because of that, any edge will go from a node in one group to a node in the other group. This allows us to color the nodes of this graph with two different colors like Blue and Red such that no two adjacent node (nodes that are connected with an edge) have the same color. Now consider performing DFS. If we have a bipartite graph, with every move from one node to next node, the color of the node has to change. So the idea is to start DFS from any random initial node that we can color it by color X for example. Then, If we go to an unvisited node, we color it by the other color and if we go to a visited node, we need to make sure its color is different from the current node color. Otherwise this will not be a bipartite graph. We continue this until all nodes are visited.

Cearly the complexity here is the same as DFS as there is one extra check or color assignment at every step. So this algorithm runs in $O(E+V)$.



Here is a psudocode for this problem:

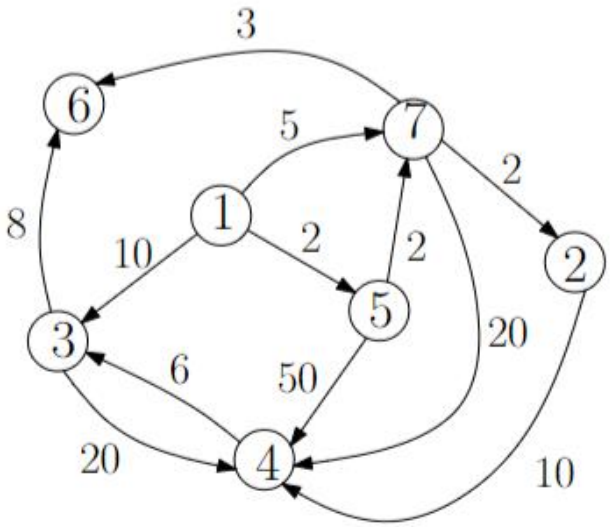
DFS(G,u):
u.visited = TRUE
for each $v \in G.Adj[u]$ **do**:
    if (v.visited == True) **then**
        if (v.color == u.color) **then**
            Print: "Not Bipartite!"
            Return
        end if
    else
        v.color= 1− u.color
        DFS(G,v)
    end if
end for

Init():
for each $u \in G$ **do**:
    u.visited = FALSE
    u.color = -1
end for
for each $u \in G$ **do**:
    if (u.visited == FALSE) **then**
        u.color = 1
        DFS(G, u)
    end if
end for
Print: "Bipartite!"
Return

Question 3: Let H be G but with the edge weights negated. We find the minimum spanning tree T of H. We claim that the same tree T in G contains the maximum bottleneck paths between all pairs of vertices. Assume to the contrary that this is false. So there exists a pair of vertices u and v and a maximum bottleneck path Q between u and v such that $wt(Q) > wt(P)$, where P is the path between u and v in T. Let e_P be the edge of minimum weight in P. So deleting e_P splits T into two subtrees T_u and T_v such that T_u contains u and T_v contains v. Since Q is a path between u and v, some edge e of Q must connect T_u and T_v. Let e_Q be edge of minimum weight in Q. Therefore, $weight(e) \geq weight(e_Q) = wt(Q) > wt(P) = weight(e_P)$. It follows that $-weight(e) < -weight(e_P)$. Therefore, if we replace e_P by e, we would get a spanning tree of H that has a smaller weight than T, which is an impossibility.

We first form H in $O(n + m)$ time. Then, we run Kruskal's algorithm on H in $O(m \log n)$ to obtain a minimum spanning tree T. Then, for each vertex v, we can run a BFS of T from v in $O(n)$ time. This sets up the parent points that we can then traverse to report the maximum bottleneck paths from v to all other vertices in $O(n^2)$ time. Thus, the total running time is $O(n^3)$.

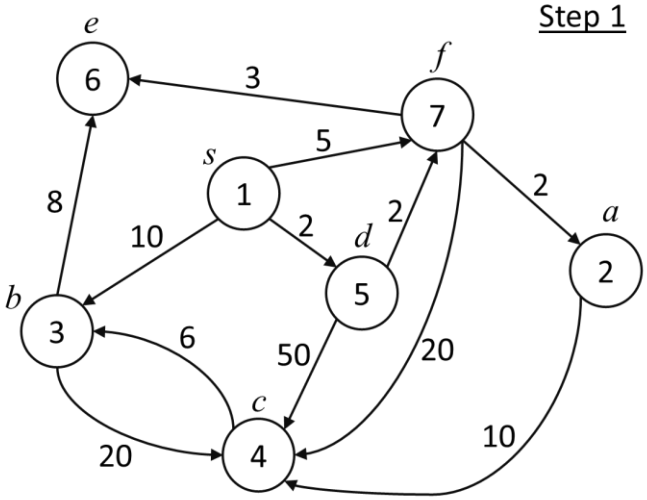
Question 4 (20 points)



Q4a. (10 points) Run Dijkstra's single-source shortest path algorithm on the following directed graph G. Use vertex 1 as the source. Show the graph G and the values of $u.d$ and $u.p$ for every vertex u after removing and processing each vertex from the min heap Q as in the lecture notes. Use the same convention and notation as in the lecture notes to show your intermediate steps.

Solution:

Step 1

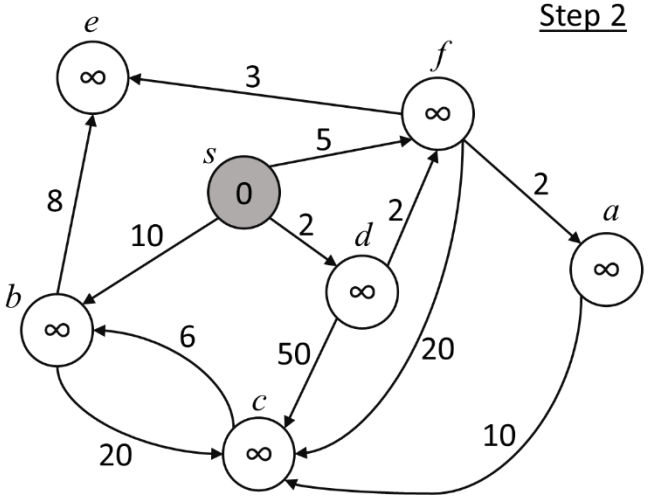


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]							
p[u]							

Step 2

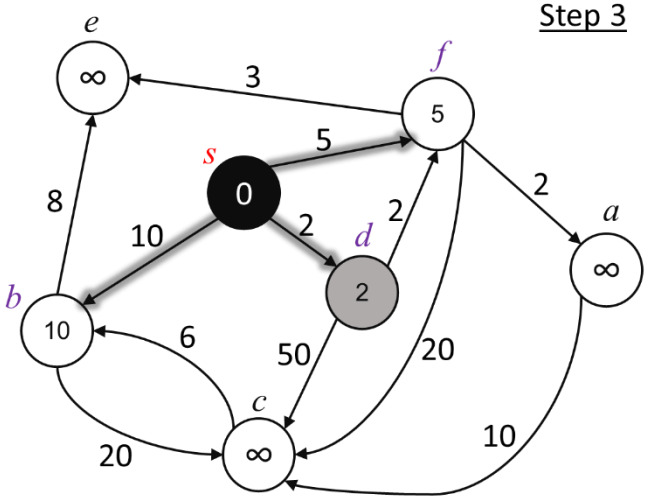


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	∞	∞	∞	∞	∞	∞
p[u]	-						

Step 3

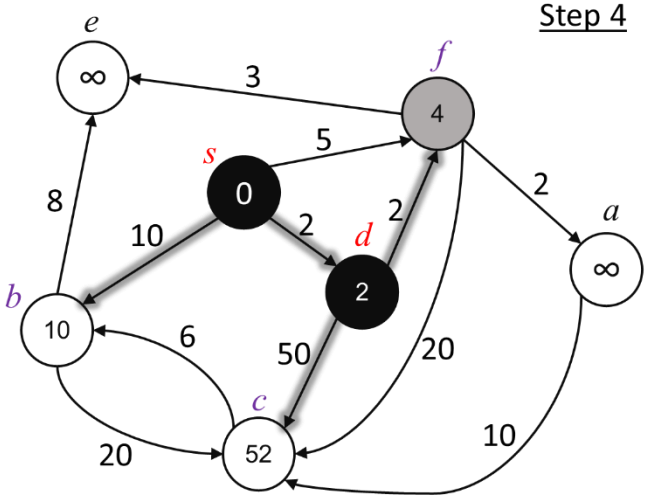


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	∞	10	∞	2	∞	5
p[u]	-		*s*		*s*		*s*

Step 4

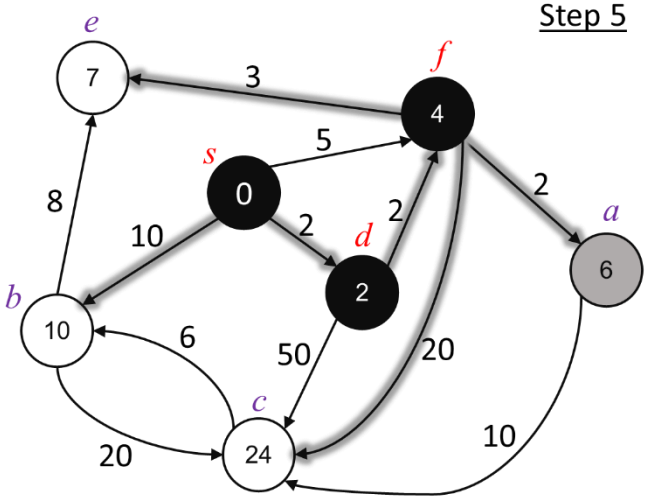


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	∞	10	52	2	∞	4
p[u]	-		*s*	*d*	*s*		*d*

Step 5

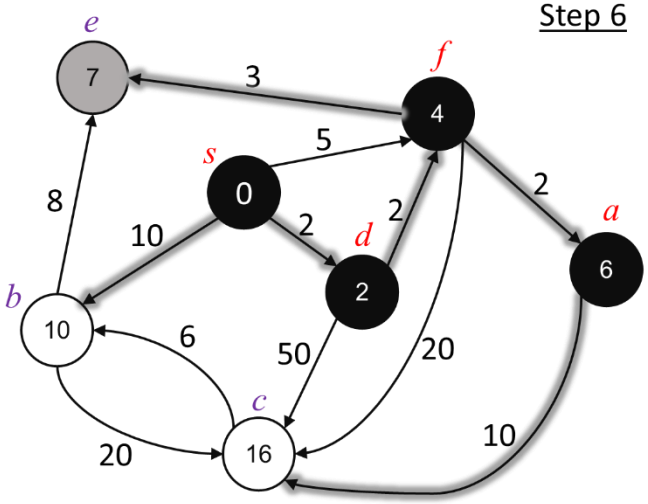


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	6	10	24	2	7	4
p[u]	-	*f*	*s*	*f*	*s*	*f*	*d*

Step 6

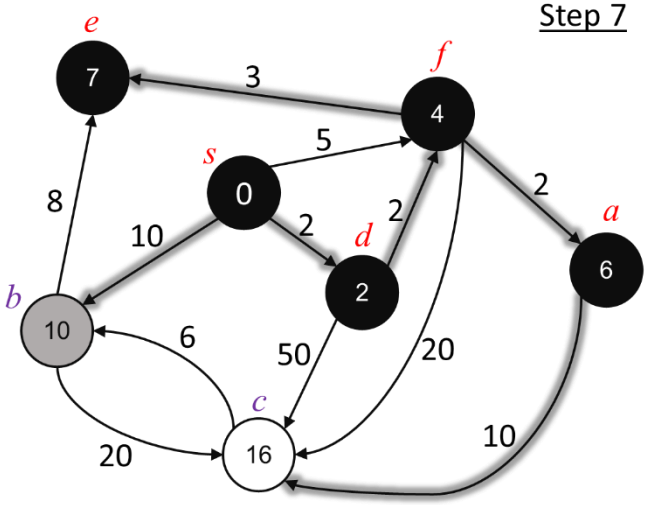


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	6	10	16	2	7	4
p[u]	-	*f*	*s*	*a*	*s*	*f*	*d*

Step 7

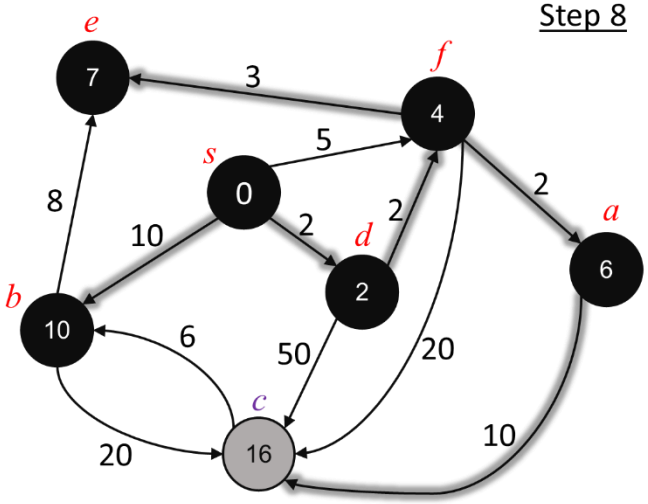


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	6	10	16	2	7	4
p[u]	-	*f*	*s*	*a*	*s*	*f*	*d*

Step 8

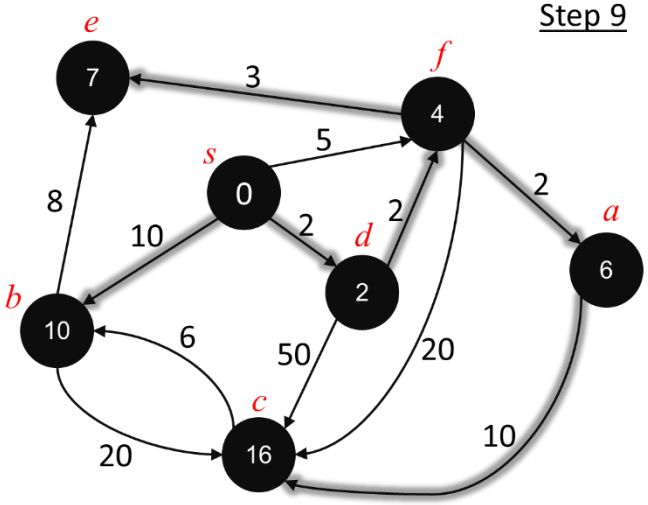


Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	6	10	16	2	7	4
p[u]	-	*f*	*s*	*a*	*s*	*f*	*d*

Step 9



Adjacency Matrix
s: *b*, *d*, *f*
a: *c*
b: *c*, *e*
c: *b*
d: *c*, *f*
e: /
f: *a*, *c*, *e*

min-heap

u	*s*	*a*	*b*	*c*	*d*	*e*	*f*
d[u]	0	6	10	16	2	7	4
p[u]	-	*f*	*s*	*a*	*s*	*f*	*d*

Q4b. (10 points) Run Floyd-Warshall algorithm on the above graph. For each $k = 0, 1, 2, 3, 4, 5, 6, 7$ in this order, show d_{ij}^{k} as a 7×7 matrix.

Solution:

Step 1 (when $k = 0$)

$$d_{ij}^{k=0} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & \infty & 2 & \infty & 5 \\ \infty & 0 & \infty & 10 & \infty & \infty & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & 0 & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & \infty & 20 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 2 (when $k = 1$)

$$d_{ij}^{k=1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & \infty & 2 & \infty & 5 \\ \infty & 0 & \infty & 10 & \infty & \infty & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & 0 & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & \infty & 20 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 3 (when $k = 2$)

$$d_{ij}^{k=2} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & \infty & 2 & \infty & 5 \\ \infty & 0 & \infty & 10 & \infty & \infty & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 50 & 0 & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & \infty & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 4 (when $k = 3$)

$$d_{ij}^{k=3} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & 30 & 2 & 18 & 5 \\ \infty & 0 & \infty & 10 & \infty & \infty & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & 14 & \infty \\ \infty & \infty & \infty & 50 & 0 & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & \infty & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 5 (when $k = 4$)

$$d_{ij}^{k=4} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & 30 & 2 & 18 & 5 \\ \infty & 0 & 16 & 10 & \infty & 24 & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & 14 & \infty \\ \infty & \infty & 56 & 50 & 0 & 64 & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & 18 & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 6 (when $k = 5$)

$$d_{ij}^{k=5} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & 30 & 2 & 18 & 4 \\ \infty & 0 & 16 & 10 & \infty & 24 & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & 14 & \infty \\ \infty & \infty & 56 & 50 & 0 & 64 & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & 18 & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

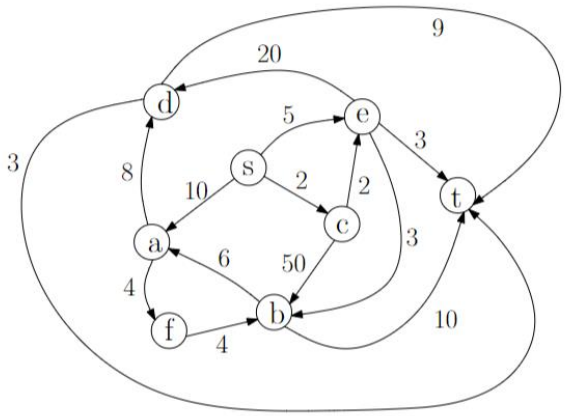
Step 7 (when $k = 6$)

$$d_{ij}^{k=6} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & \infty & 10 & 30 & 2 & 18 & 4 \\ \infty & 0 & 16 & 10 & \infty & 24 & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & 14 & \infty \\ \infty & \infty & 56 & 50 & 0 & 64 & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & 18 & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Step 8 (when $k = 7$)

$$d_{ij}^{k=7} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 6 & 10 & 16 & 2 & 7 & 4 \\ \infty & 0 & 16 & 10 & \infty & 24 & \infty \\ \infty & \infty & 0 & 20 & \infty & 8 & \infty \\ \infty & \infty & 6 & 0 & \infty & 14 & \infty \\ \infty & 4 & 20 & 14 & 0 & 5 & 2 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & 2 & 18 & 12 & \infty & 3 & 0 \end{bmatrix} \end{array}$$

Question 5 (20 Points)

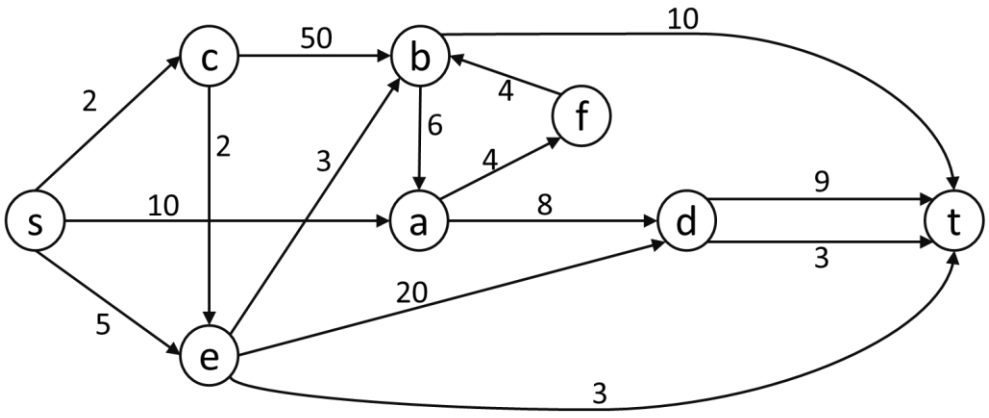


Run Ford-Fulkerson's maximum flow algorithm on the following directed graph G. Use s as the source and t as the sink. Use the same convention and notation as in the lecture notes to show:

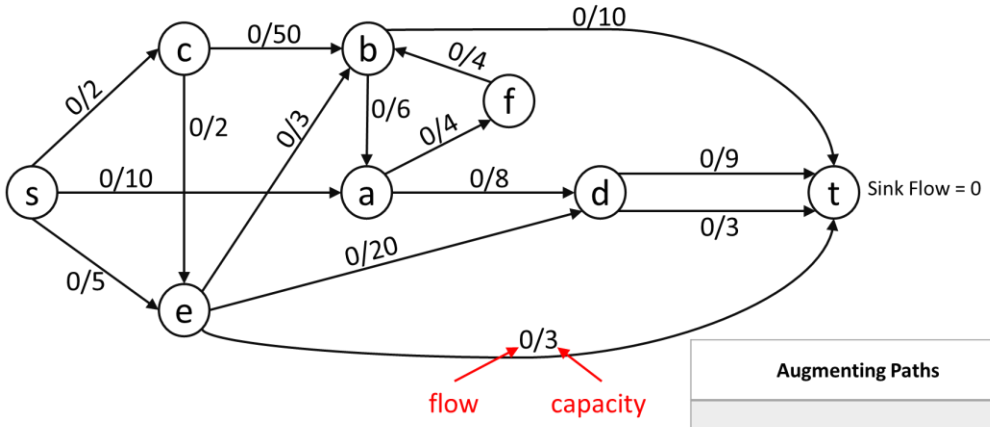
- The residual graph G_f and the augmenting path selected in G_f.
- The flow values on the edges of G after using the augmenting path selected to update the flow.

Solution:

Redraw the graph for better presentation/understanding

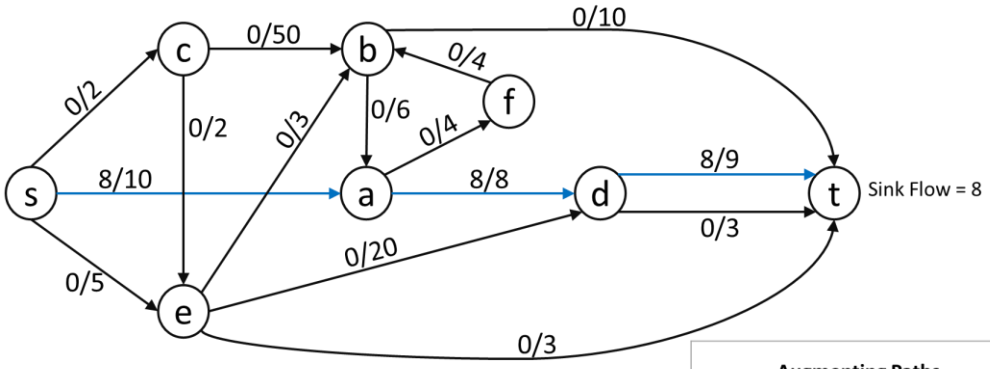


Iteration: 0



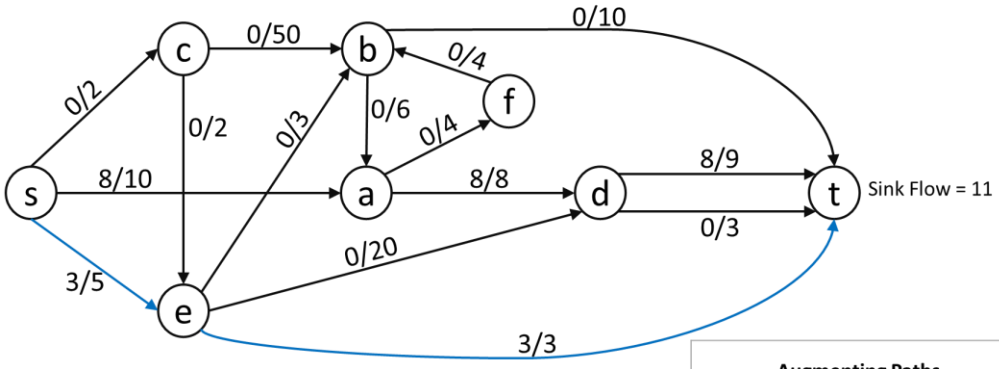
Augmenting Paths	Bottle neck capacity

Iteration: 1



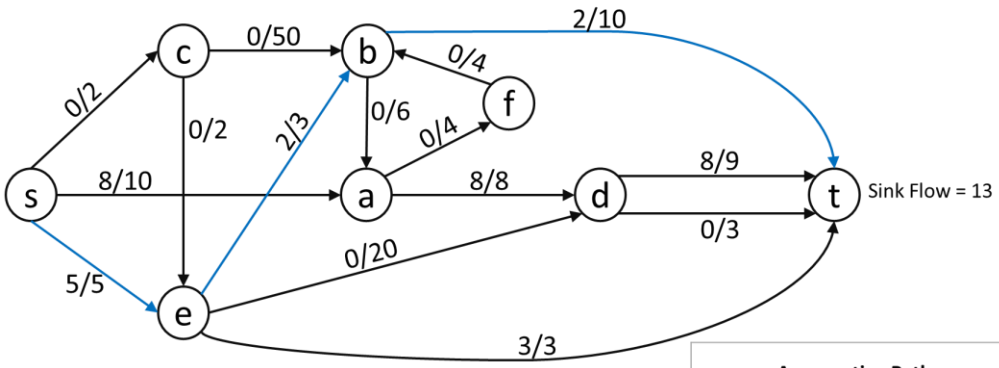
Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8

Iteration: 2



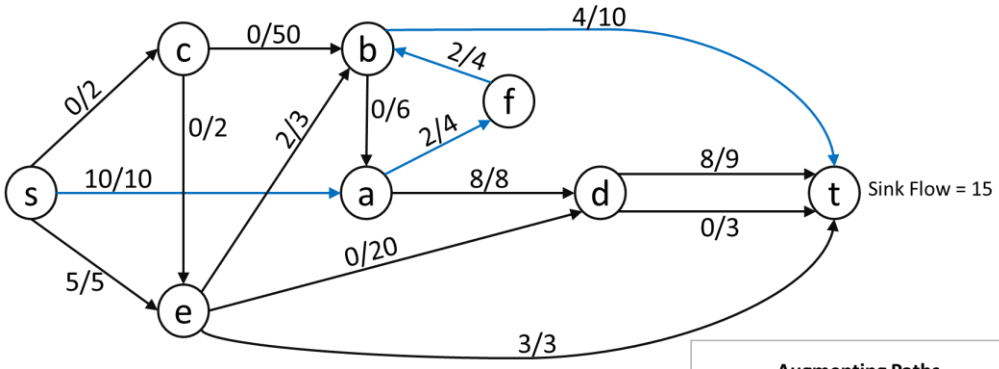
Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8
s -> e -> t	3

Iteration: 3



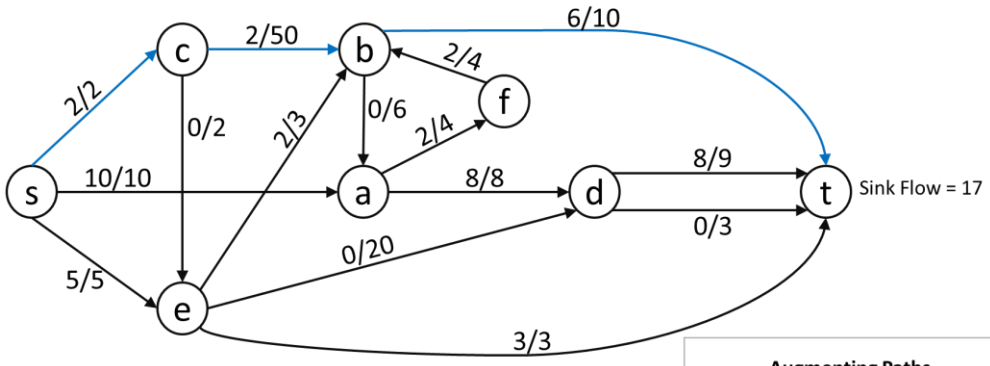
Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8
s -> e -> t	3
s -> e -> b -> t	2

Iteration: 4



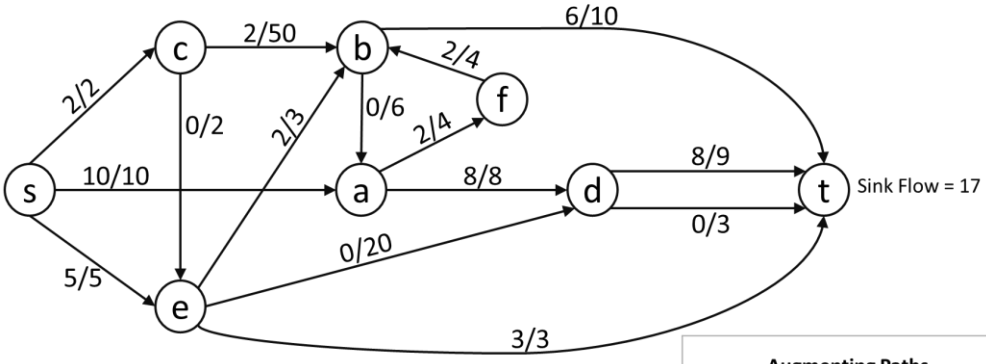
Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8
s -> e -> t	3
s -> e -> b -> t	2
s -> a -> f -> b -> t	2

Iteration: 5



Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8
s -> e -> t	3
s -> e -> b -> t	2
s -> a -> f -> b -> t	2
s -> b -> t	2

Final result



Augmenting Paths	Bottle neck capacity
s -> a -> d -> t	8
s -> e -> t	3
s -> e -> b -> t	2
s -> a -> f -> b -> t	2
s -> b -> t	2