Name: Rollno:

CS345: Design and Analysis of Algorithms (Midsem Exam)

18th September 2024

Total Number of Pages: 7 Time: 2 hr Total Points 80

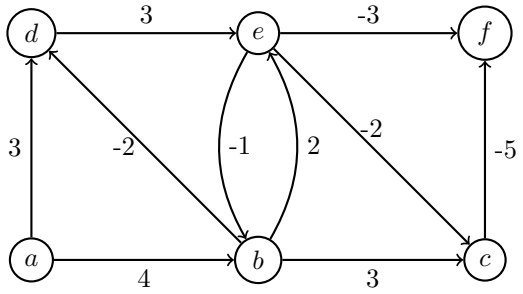
Instructions

1. All questions are compulsory.
2. Answer all the questions in the space provided in the question paper booklet. The space provided is much more than what is necessary to write down the solution.
3. Use the space provided in the paper for rough work.
4. You may cite and use algorithms and their complexity as done in the class.
5. Cheating or resorting to any unfair means will be severely penalized.
6. Superfluous and irrelevant writing will result in negative marking.
7. Using pens (blue/black ink) and not pencils. Do not use red pens. for answering.
8. Please bring your ID cards.
9. For questions involving algorithm design, it is sufficient to give a clear and concise description of your algorithm. You dont need to give the formal pseudocode.
10. If you design an algorithm, that is less efficient than what has been asked, you will get partial credit, provided you clearly mention what is the time complexity of your algorithm.
11. It is advisable to solve a problem first before writing down the solution.
12. The questions are *not* arranged according to the increasing order of difficulty.

Question	Points	Score
1	12	
2	18	
3	12	
4	12	
5	12	
6	14	
Total:	80	

Name: Rollno:

Question 1. Consider the following directed weighted graph G.



(a) (1 point) Does G have any negative weight cycle? **No**

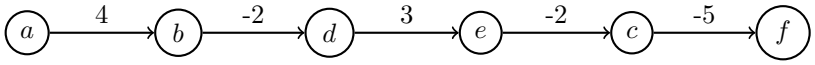
(b) (5 points) Give the matrix obtained by running Bellman Ford algorithm on G starting at vertex a.

Solution:

	1	2	3	4	5
a	0	0	0	0	0
b	4	4	4	4	4
c	∞	7	4	3	3
d	3	2	2	2	2
e	∞	6	5	5	5
f	∞	∞	2	-1	-2

(c) (3 points) What is the corresponding shortest path tree? (If there are multiple such trees, you can give any one)

Solution:

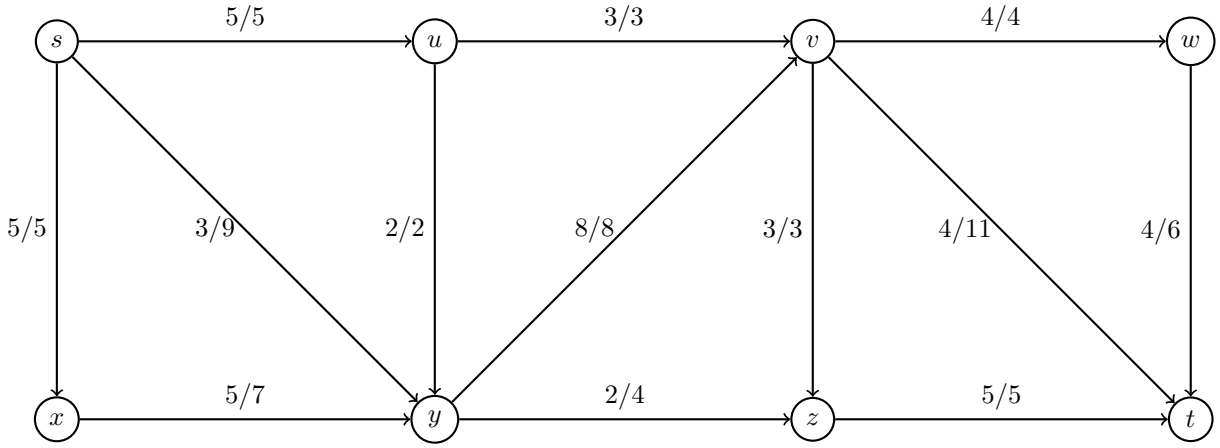


(d) (3 points) What is the distance of each vertex from a?

a	b	c	d	e	f
0	**4**	**3**	**2**	**5**	**-2**

Question 2. Consider the following flow network, where each edge is labeled with the current flow and capacity (that is, a/b means that the capacity of the edge is b and the current flow value is a). The source node is s, and the sink is t.

Name: Rollno:

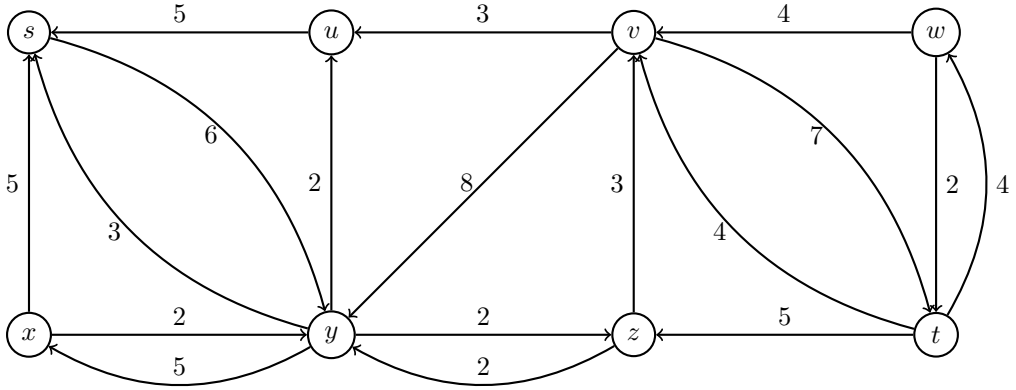


(a) (1 point) Is this a valid flow? **Yes**

(b) (1 point) What is the value of the flow? **13**

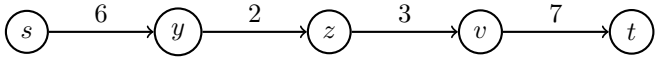
(c) (4 points) Draw the residual network corresponding to the given flow.

Solution:



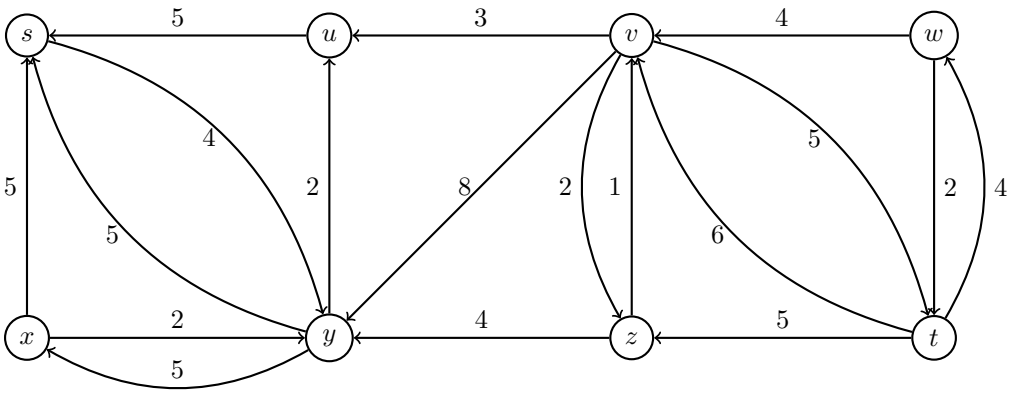
(d) (3 points) Suppose we perform one more iteration of the Ford Fulkerson algorithm, starting with the above flow. Give any one corresponding augmenting path of that iteration.

Solution:



(e) (3 points) Let f' be the flow obtained by augmenting along the path in the previous part. Draw the residual network corresponding to f'.

Solution:



(f) (3 points) What is the value of the maximum flow in this network? **15**

(g) (3 points) List the vertices in the source side of the *st*-cut corresponding to this max flow? What is the capacity of this cut?

Solution: $\{s, u, x, y\}$
Capacity of the cut = 15

Question 3. Let G be a DAG. A *sink path* from a vertex u is a path that ends in a sink. v is a *terminal vertex* for a vertex u, if v is a sink and there is a path from u to v.

(a) (6 points) Design an $O(n + m)$ time algorithm `NumSink`(G) that given a DAG G computes the number of sink paths from every vertex in G. The number of sink paths from a sink is 1.

Solution: We will store the number of sinks from a vertex v in the array $S[v]$.

Algorithm 1: NumSink(G)

Data: Given a DAG G
foreach *sink vertex u in G* **do**
  Set $S[u] = 1$;
end
Perform a topological ordering of G, say $v_1, v_2, \ldots v_n$;
for $i = n - 1$ *to* 1 **do**
  if v_i *is not a sink* **then**
    Set $S[v_i] = \sum_{(v_i, v_j) \in G} S[v_j]$;
  end
end

(b) (6 points) Design an $O(n + m)$ time algorithm `NumTerminus`(G, u) that computes the number of terminals for the vertex u in the DAG G.

Solution:

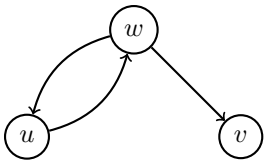
Algorithm 2: NumTerminus(G,u)

Data: Given a DAG G and a vertex u
Set $c = 0$;
Perform DFS(u) on G;
Whenever a sink is reached that is unvisited, increment c by 1;
return c;

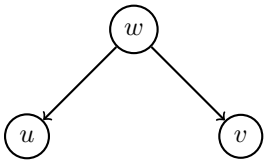
Question 4. Prove or disprove the following statements.

(a) (6 points) If G is a directed graph containing a path from u to v, and $D(u) < D(v)$ in a DFS of G, then v is a descendent of u in the corresponding DFS forest.

Solution: The statement is false. Consider the following graph.



Suppose DFS visits w first, and among the neighbours of w, it visits u followed by v. In this case $D(u) < D(v)$. Also there is a directed path from u to v. However the DFS forest looks as follows.



(b) (6 points) A directed graph G is acyclic if and only if DFS(G) produces no back edges.

Solution: If DFS(G) produces a back edge, say (v, u), then by definition of back edge v is a descendent of u in the DFS forest of G. Therefore, we get a cycle by traversing from u to v in G via the tree edges and then going from v to u via the back edge (v, u).

Suppose G has a cycle, say C. Let u be the first vertex to be discovered by the DFS in C. Since none of the other vertices in C have been discovered till this point, they would be descendants of u in the DFS tree. Since C is a cycle, there is a vertex v in C such that (v, u) is an edge. Now since v is also a descendant of u in the DFS tree, this edge will become a back edge in the DFS.

Question 5. Consider an n node complete binary tree T, where $n = 2^d - 1$ for some d. Each node $v \in T$ is labeled with a real number x_v. You may assume that all the values are distinct. A node $v \in T$ is said to be a **local minima** if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labelling is only specified in the following implicit way: For each node v, you can determine the value x_v by querying the node v.

(a) (6 points) Give an algorithm to find a local minimum of T using only $O(\log n)$ queries to the nodes of T.

Solution:

Algorithm 3: TreeLocalMinima(T)

Data: Given a complete binary tree T
Set u to be the root of T;
while u *is not a local minima* **do**
  Move to a child v of u such that $x_v < x_u$;
  Set $u = v$;
end
return u;

Since the depth of a complete binary tree is $\log n$ the time complexity is $O(\log n)$ as well.

(b) (6 points) State the proof of correctness of your algorithm.

Solution: There always exists a local minima in T, since the global minima is also a local minima. In every iteration of the `while` loop, the value of the current node decreases. Since the number of nodes in T is finite, therefore the algorithm terminates. Also by the condition of the while loop, when the algorithm terminates, it must be at a local minima.

The depth of a complete binary tree in $\log n$. By design, the algorithm traces a path from the root to a leaf in the worst case. Hence the time is $O(\log n)$.

Question 6. Given a string s, an *operation* on s is defined as follows: Either (i) inserting a single character into s, (ii) deleting a single character from s, or (iii) replacing a character in s with another character.

For two strings s_1 and s_2, the *edit distance* of s_1 and s_2, $\text{ED}(s_1, s_2)$ is defined to be the minimum number of operations to convert s_1 to s_2.

For example, $\text{ED}(\text{kitten}, \text{sitting}) = 3$. We can (i) Replace 'k' with 's', (ii) Replace 'e' with 'i', (iii) Insert 'g' at the end.

(a) (3 points) Prove that $\text{ED}(s_1, s_2) = \text{ED}(s_2, s_1)$.

Solution: This is true because, the inverse of the *insert* operation is *delete*, and vice versa. And the inverse of the *replace* operation is itself. Therefore any sequence of operation that converts s_1 to s_2, can be replaced with the corresponding inverses, to convert s_2 to s_1.

(b) (6 points) Give a suitable recursive formulation of $\text{ED}(s_1, s_2)$. (Do not forget the base case)

Solution: Let s_1 be a string of length n and s_2 be a string of length m. Let $s_1[1 \dots i]$ denote the prefix of s_1 consisting of the first i characters. Similarly $s_2[1 \dots i]$ denote the prefix of s_2 consisting of the first i characters.

Let $f(i, j)$ be the minimum number of operations required to convert $s_1[1 \dots i]$ to $s_2[1 \dots j]$. Then in this notation, $\text{ED}(s_1, s_2) = f(n, m)$.

For $i, j \geq 1$ we have:

$$f(i,j) = \begin{cases} f(i-1, j-1) & \text{if } s_1[i] = s_2[j] \\ \min\{1 + f(i, j-1), 1 + f(i-1, j), 1 + f(i-1, j-1)\} & \text{if } s_1[i] \neq s_2[j] \end{cases}$$

The second case corresponds to inserting, deleting or replacing a character.

Base Case: If $i = 0$, then $f(i, j) = j$ and if $j = 0$, then $f(i, j) = i$.

(c) (5 points) Given s_1 and s_2, provide a dynamic programming solution to compute ED(s_1, s_2).

Solution: We create a matrix A of size $(n+1) \times (m+1)$ such that $A[i,j] = f(i,j)$. We then fill the matrix and output the entry $A[n,m]$.

Algorithm 4: EditDistance(s_1, s_2)

```
Data: Given two strings s_1 and s_2
for i = 0 to n do
    A[i, 0] = i;
end
for j = 0 to m do
    A[0, j] = j;
end
for i = 1 to n do
    for j = 1 to m do
        if s_1[i] == s_2[j] then
            A[i, j] = (f(i − 1, j − 1);
        else
            A[i, j] = 1 + min(f(i − 1, j − 1), f(i − 1, j), f(i, j − 1));
        end
    end
end
return A[n, m];
```