**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SRI CITY**
**END EXAMINATION – MONSOON 2025**

**ADVANCE DATA STRUCTURES AND ALGORITHMS**

**CSE :UG 2 (PC)** **Date: 20-11-2025**

**Duration: 180 Mins (02:30-05:30 PM)** **Max. Marks: 50**

---

**Instructions** **Roll No:_____**

1. This question paper consists of six pages.
2. All questions are compulsory. All sub-parts of a question should be written together.
3. Attach the question paper to the answer sheet.

---

## PART A MULTIPLE CHOICE QUESTIONS
**(Each question carries one mark)**

1. Suppose a BST is converted into an AVL tree. Which of the following statements is correct?
   a. The in-order traversal of the AVL tree and the BST will be the same.
   b. The pre-order traversal of the AVL tree and the BST will be the same.
   c. The post-order traversal of the AVL tree and the BST will be the same.
   d. None of the above.
2. Which of the following statement is/are correct?
   P1: Every tree will always be a graph
   P2: Every graph will always be trees.
   P3: Every tree will be a graph, but every graph will not be a tree
   P4: Every graph will be a tree, but every tree will not be a graph.
   a. P1 and P2 b. P1 and P3 c. Only P1 d. Only P3
3. ***G*** is a is an undirected graph with vertex set and edge set {***V1, V2, V3, V4, V5, V6, and V7***} and an edge set of ***{V1V2, V1V3, V1V4, V2V4, V2V5,V3V4, V4V5, V4V6, V5V6, V6V7}***. A breadth first search of the graph is performed with ***V1*** as the root node. Which of the following is a tree edge?
   a. ***V2V4*** b. ***V1V4*** c. ***V4V5*** d. ***V3V4***
4. When developing a dynamic programming algorithm, the sequence of steps followed is:
   1. Construct an optimal solution from computed information.
   2. Recursively define the value of an optimal solution.
   3. Characterize the structure of an optimal solution.
   4. Compute the value of an optimal solution, typically in a bottom-up fashion.
   a. 2-3-1-4 b. 2-1-3-4 c. 3-2-1-4 d. 3-2-4-1
5. Consider the strings “PQRSTPQRS” and “PRATPBRQRPS”. What is the length of the longest common subsequence?
   a. 9 b. 8 c. 7 d. 6
6. Red-black trees are one of many search tree schemes that are “balanced” in order to guarantee that basic dynamic-set operations take _____ time in the worst case.
   a. O(1) b. O(log n) c. O(n) d. O(n log n)
7. What is the primary goal of network flow problems?
   a. Maximizing flow b. Minimizing cost
   c. Finding shortest path d. Balancing loads
8. Bellmann ford algorithm provides solution for _____ problems.
   a. Minimum Spanning Tree b. All Pairs Shortest path
   c. Max Flow Problem d. Single Source Shortest path

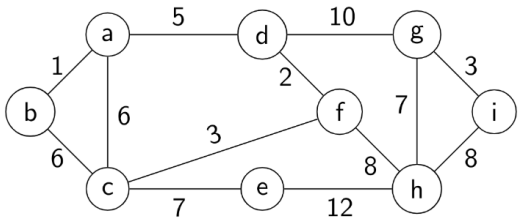
9. Consider the following table

| | |
|---|---|
| (P) Kruskal Algorithm | (i) Monte Carlo |
| (Q) Flyod Warshall | (ii) Greedy |
| (R) Approximating Circle Radius | (iii) Las Vegas |
| (S) Quicksort | (iv) Dynamic Programming |

Match the algorithms to the design paradigms they are based on.

a. P - iv, Q - ii, R - i, S - iii
b. P - ii, Q - iv, R - i, S - iii
c. P - iv, Q - ii, R - iii, S - i
d. P - ii, Q - iv, R - iii, S - i

10. For the undirected, weighted graph given below, which of the following sequences of edges represents a correct execution of Prim's algorithm to construct a Minimum Spanning Tree?

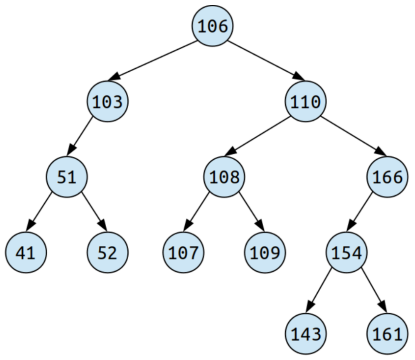



a. (a, b), (d, f), (f, c), (g, i), (d, a), (g, h), (c, e), (f, h)
b. (c, e), (c, f), (f, d), (d, a), (a, b), (g, h), (h, f), (g, i)
c. (d, f), (f, c), (d, a), (a, b), (c, e), (f, h), (g, h), (g, i)
d. (h, g), (g, i), (h, f), (f, c), (f, d), (d, a), (a, b), (c, e)

## PART B DESCRIPTIVE QUESTIONS

1. Find the set Longest Common Subsequence (X, Y ) where X = BACDB, Y = BDCB. Show your working (the Table) and justify your method of extracting the longest common subsequences. **[5 Marks]**

2. In a binary search tree, the lower bound of a key is the node in the tree with the smallest value greater than or equal to the key, and the upper bound of a key is the node in the tree with the largest value less than or equal to the key. For example, in the BST shown here, the lower bound of 137 is the node containing 143, and the upper bound of 137 is the node containing 110. **[5 Marks]**




If the key is less than all the values, its upper bound is nullptr, so the upper bound of 15 is nullptr. (The lower bound of 15 is the node containing 41.)

Similarly, if the key is greater than all elements in the BST, its lower bound is nullptr. For example, the lower bound of 261 is nullptr. (The upper bound of 261 is the node containing 166.)

In the event that the key happens to appear inside the BST, then the key itself is its own lower bound and upper bound. For example, the lower and upper bounds of 106 are each the node containing 106.

It is a bit confusing that a key's lower bound is always at least as large as its upper bound, but, alas, that is the naming convention we use.

Your task is to implement a function:

**Bounds boundsOf(Node* root, int key);**

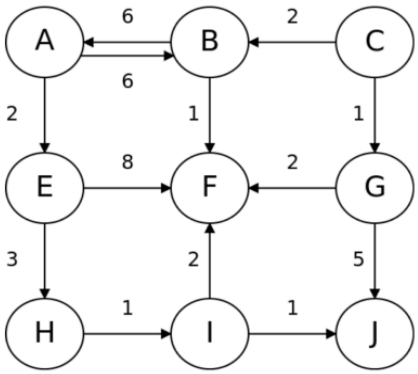
that takes in a pointer to the root of a BST, along with an integer key, then returns the lower and upper bound of that key in the tree. Here, Node and Bounds are structs defined as follows:

```
struct Node {
   int value;
   Node* left;
   Node* right;
};
struct Bounds {
   Node* upperBound;    Node* lowerBound;  };
```

Given below are the recursive solution to the above problem which is incomplete. Your job is to complete it by filling the missing code snippets.

```
Bounds boundsRec(Node* root, int key, Node* upper, Node* lower) {
      if (_____1_____) return { upper, lower };
    if (key == ____2_____) return { _____3___,  _____4_____};
       else if (key < root->value) {
         return boundsRec(____5_____, key, _____6_____, _____7_____ );
                                    }
       else {
     return boundsRec(_____8_____, key, _____9_____, _____10_____ );
           }
  }
Bounds boundsOf(Node* root, int key)      // Driver
{
return boundsRec(root, key, nullptr, nullptr);
}
```

3. Answer the following three questions below **[1 + 2 + 2 Marks]**
   a. You need to determine whether an undirected graph is connected. How could you use BFS to do so? For the next two parts, refer to the graph shown below:



b. Write the order that a depth-first search (DFS) would visit vertexes if it were looking for a path from vertex A to vertex I. Also write the path it would return.

c. Write the order that a breadth-first search (BFS) would visit vertexes if it were looking for a path from vertex C to vertex H. Also write the path it would return.

4. Answer the following two questions **[3 + 2 Marks]**

a. You are given an exam with questions numbered 1, 2, 3, . . . , n. Each question ***i*** is worth ***pi*** points. You must answer the questions in order, but you may choose to skip some questions. The reason you might choose to do this is that even though you can solve any individual question ***i*** and obtain the ***pi*** points, some questions are so frustrating that after solving them you will be unable to solve any of the following ***fi*** questions. Suppose that you are given the ***pi*** and ***fi*** values for all the questions as input. Devise the most efficient algorithm you can for choosing a set of questions to answer that maximizes your total points, and compute its asymptotic worstcase running time as a function of ***n.***

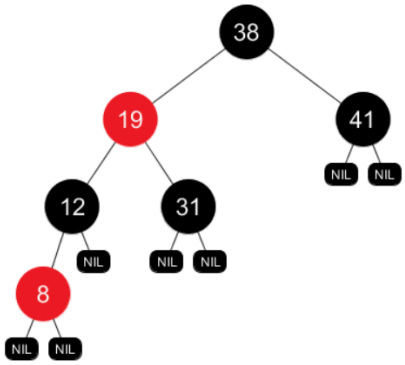
b. Let G=(V,E) be a directed graph where V is the set of vertices and E the set of edges. Then which one of the following graphs has the same strongly connected components as G ? Discuss

i. G1=(V,E1) where E1={(u,v)|(u,v)∉E}

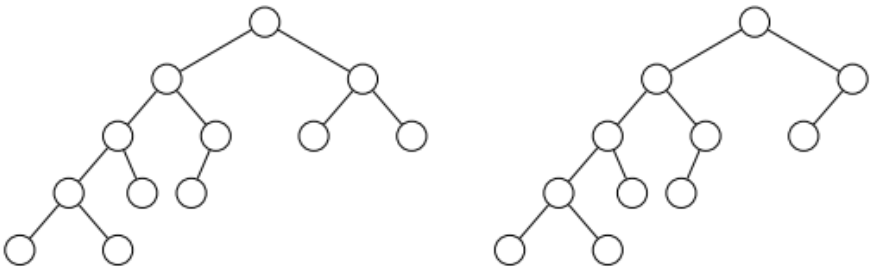
ii. G2=(V,E2 )where E2={(u,v) | (v,u)∈E}

5. Answer the following two questions **[ 3 + 2 Marks ]**

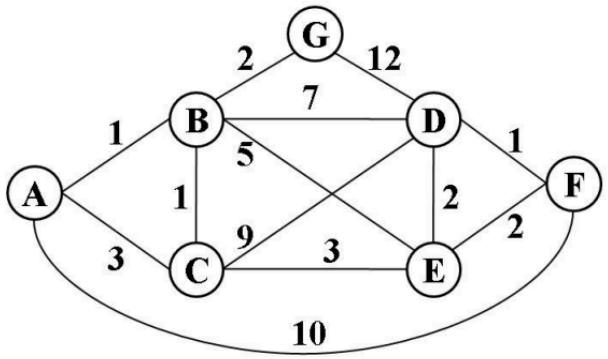
a. Consider the following RED BLACK Tree (red nodes are R and black nodes are B). Now show the red-black trees that result from the successive deletion of the keys in the order 8,12,19,31,38,41.




b. For each of the following examples, if the nodes can be colored red or black to make a legitimate red-black tree, then give such a coloring. If not, then say that they cannot.

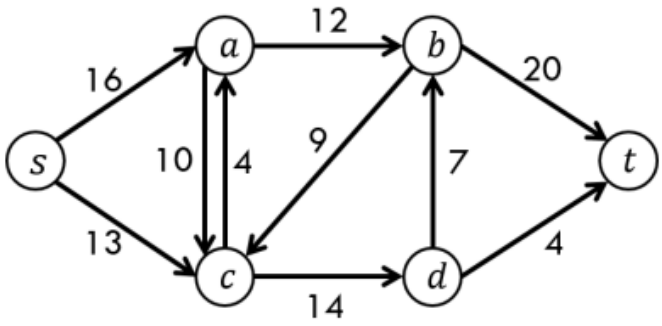


6. Consider the following undirected, weighted graph: **[5 Marks]**




Step through Dijkstra's algorithm to calculate the single-source shortest paths from A to every other vertex. Show your steps by filling in the Distance table/matrix. Also list the vertices in the order in which you marked them known. Finally, indicate the lowest-cast path from node A to node F.

7. Given a directed graph G = (V, E), where each edge e is associated with its capacity c(e) > 0. Two special nodes source **s** and sink **t** are given (s ≠ t) **[ 5 Marks ]**




Use Ford-Fulkerson Algorithm to maximize the total amount of flow from **s** to **t** subject to two constraints
– Flow on edge e doesn't exceed c(e)
– For every node v ∉ s, t, incoming flow is equal to outgoing flow.
Task you need to do

1. Write Pseudo code of the Ford-Fulkerson Algorithm
2. Properly draw Network Flow Graph and Residual Graph after each iteration of algorithm .

8. Answer the following two questions **[ 3 + 2 Marks]**
   a. Lucky wants to study for an exam using a deck of **m** flashcards. Lucky is not good at learning many concepts at once, so he only wants to learn **n** of these flashcards, one new concept each day. He will be happy with any subset of **n**; after all, the exam will be open-book and he can spend more time and go through the other **m − n** flashcards during the exam if need be. For this problem, assume that **n ≤ m.** Lucky's algorithm for studying is as follows: Every morning he wakes up and randomly shuffles his flashcards. Then he goes over the flashcards one-by-one in the shuffled order. For every card he has studied before, he spends one hour reviewing it again. He stops when he sees a new flashcard, spends one hour studying it, and then he calls it a day. The next morning, he repeats this procedure until he learns a new flashcard, and so on until **n** days pass. For this

problem, assume that random shuffling of cards takes negligible time. We want to study the runtime **T** which we define to be the total number of hours that Lucky spends studying.

***Example.*** Below is an example scenario with m = 4 and n = 3. Assume that flashcards are numbered 1 through m.

- On the first day, Lucky goes over the flashcards in the (randomly chosen) order: [2, 4, 1, 3]. He stops after the first flashcard 2, and calls it a day.
- On the second day, Lucky goes over the flashcards in the (randomly chosen) order: [3, 1, 2, 4]. He stops after the first flashcard 3, and calls it a day.
- On the third day, Lucky goes over the flashcards in the (randomly chosen) order: [3, 2, 4, 1]. He reviews flashcards 3 and 2 (he has seen them before), and then studies flashcard 4. He stops and calls it a day.
- In this example, T would be 1 + 1 + 3 = 5. In general, T depends on the random shuffles each day; in other words, T is a random variable.

What is the worst case runtime of this algorithm? In other words, what is the worst case value of T for the worst possible sequence of shuffles? Express your answer in big-O notation in terms of n, AND briefly justify your answer.

b. State whether the following statements are True or False
   i. The Bellman-Ford algorithm simply uses repeated edge relaxation.
   ii. Let G = (V, E) be a directed graph with negative-weight edges, but no negative-weight cycles. Then, one can compute all shortest paths from a source $s \in V$ to all $v \in V$ faster than Bellman-Ford using the technique of reweighting.
   iii. If a dynamic-programming problem satisfies the optimal-substructure property, then a locally optimal solution is globally optimal.
   iv. Suppose a decision problem A has a pseudopolynomial-time algorithm to solve A. If $P \neq NP$, then A is not solvable in polynomial time.

**MCQs Answers**

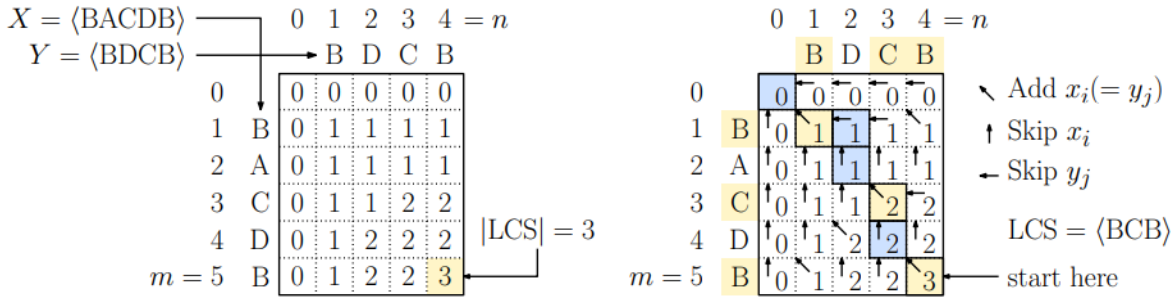
1. a. The in-order traversal of the AVL tree and the BST will be the same.
2. b. P1 and P3
3. b. ***V1V4***
4. d. 3-2-4-1
5. c. 7
6. b. O(log n)
7. a. Maximizing flow
8. d. Single Source Shortest path
9. b. P - ii, Q - iv, R - i, S - iii
10. c. (d, f), (f, c), (d, a), (a, b), (c, e), (f, h), (g, h), (g, i)

# Key of the question paper

# Descriptive Part

## 1. Answer

Answer




**Filling the table / matrix 3 marks (First Figure)**

**Justifying how they find LCS 2 marks (backtracking—- second figure)**

## 2.. Ans

```
Bounds boundsRec(Node* root, int key, Node* upper, Node* lower) {
      if (1. root == nullptr) return { upper, lower };
    if (key ==  2. root -> value) return { 3. root,   4. root};
       else if (key < root->value) {
         return boundsRec(5. root -> left  key, 6. upper,  7. root  );
                                    }
       else {
     return boundsRec(8. root->right ,  key, 9. root,  10. lower);
           }
  }
Bounds boundsOf(Node* root, int key)      // Driver
```

```
{
return boundsRec(root, key, nullptr, nullptr);
}
```

**Each correct answer carries 0.5 marks 0.5 * 10 = 5 Marks**

None

## 3. Answer

a. Run BFS at a node, marking all the nodes that are visited. After BFS is run, check to make sure all nodes are visited. If there are unvisited nodes, the graph is not connected. —---------- **1 mark**

b. DFS: examines A, B, F, (backtracks to A), E, H, I —---------- **1 Mark**

returns path {A, E, H, I} —---------- **1 Mark**

c. BFS: examines C, (neighbors of C) B, G, (neighbors of B) A, F, (neighbors of G) J, (neighbors of A) E, (neighbors of E) H —---------- **1 Mark**

returns path {C, B, A, E, H} —---------- **1 Mark**

## 4. Answer

a. Let **dp[i]** = the **maximum points you can get starting from question i**.

For each question i, you have two choices:

### Case 1 — Skip question i

- Then best score = **dp[i+1]**

### Case 2 — Solve question i

- You get $\mathbf{p_i}$ **points**
- But you cannot solve the next $\mathbf{f_i}$ **questions**
- So the next solvable question is $\mathbf{i + f_i + 1}$
- Score = $\mathbf{p_i}$ + **dp[i +** $\mathbf{f_i}$ **+ 1]**

$$dp[i] = \max(dp[i+1],\ p_i + dp[i + f_i + 1])$$

We compute dp **from right to left**.

The running time is easily seen to be O(n) (possibly with some additional tinkering to catch indices off of the end of the array).

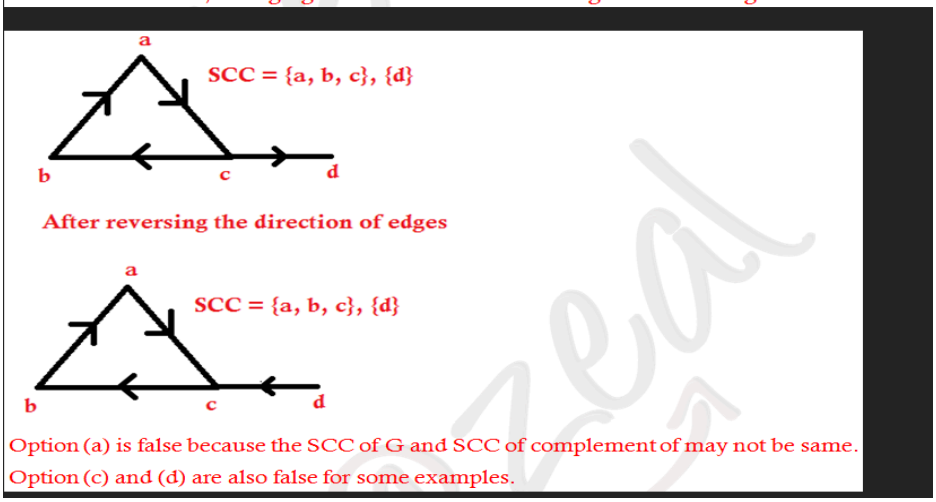
**Properly defining the cases 1.marks  Writing the formula 1.5 marks  Time complexity - 0.5 marks**

**b. Justification 2 marks**

Solution: In a directed graph G Strongly connected will have a path from each vertex to every other vertex. If the direction of the edges is reverse, then also graph is strongly connected components as G.

Option (b): $G_2 = (V, E_2)$ where $E_2 = \{(u, v) \mid (v, u) \in E\}$

In this option $G_2$, edges are reversed and hence it is a strongly connected component as similar to G. So, changing the direction of all the edges won't change the SCC.




Option (a) is false because the SCC of G and SCC of complement of may not be same.

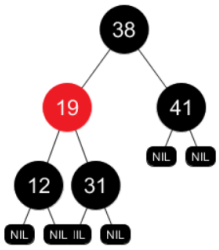
Option (c) and (d) are also false for some examples.

## 5. Answer  Del 8 - 0.5 marks

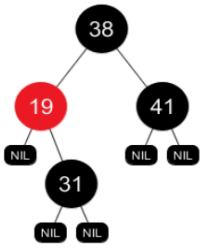
**Del 12 ——---- 1 mark**

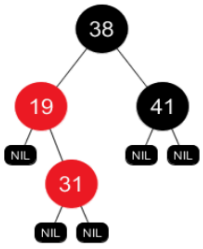
**Del 19, 31,  -  1 mark ,  Delete  38, 41——-- 0.5 marks**

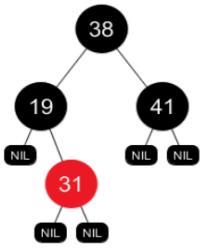
Delete 8:



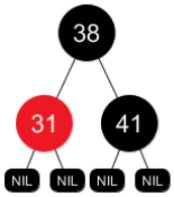

Delete 12:

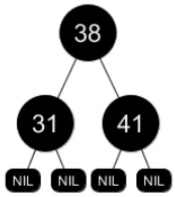


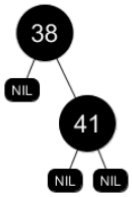



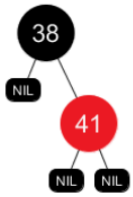
Delete 19:




Delete 31:



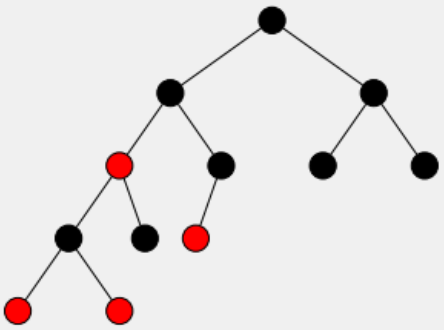

Delete 38:

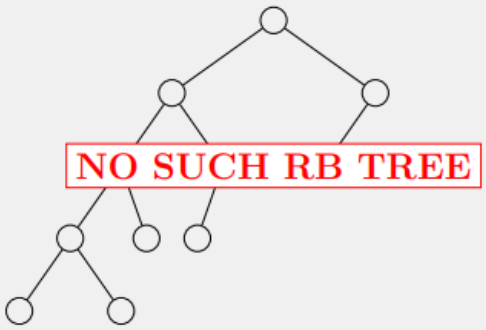


Delete 41.

## 5.b

**SOLUTION:**






## 8. Answers

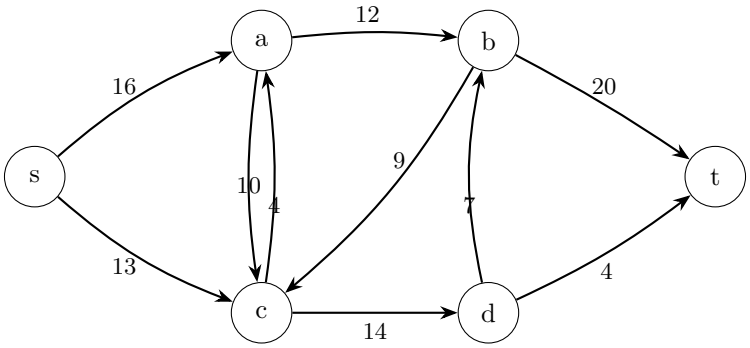
a.

The worst case happens when Lucky comes across every single flashcard he has seen before. In total, he needs to study $1 + 2 + \cdots + n = \frac{n(n+1)}{2} = O(n^2)$ flashcards.

a.

i. True
ii. False
iii. False
iv. False

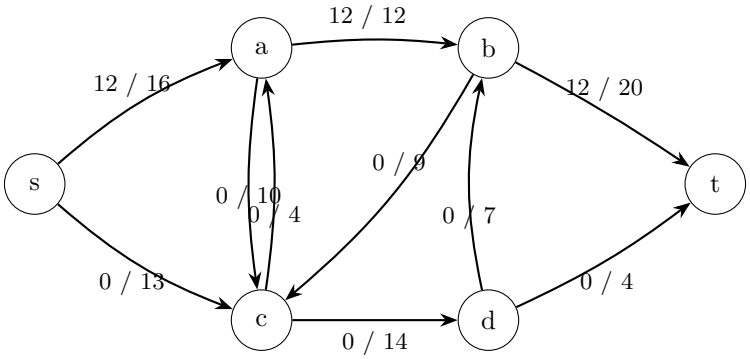
# Question 7 Answer




Original network (capacities)

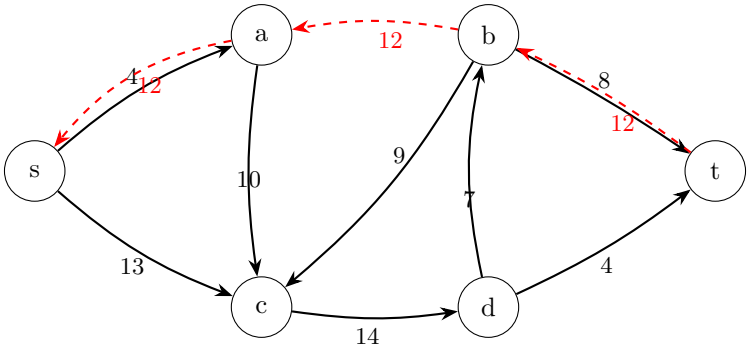
Original network (edge capacities)

## After Iteration 1 (pushed 12 along $s \to a \to b \to t$)




Flow network after Iteration 1 (flows shown as flow / capacity)

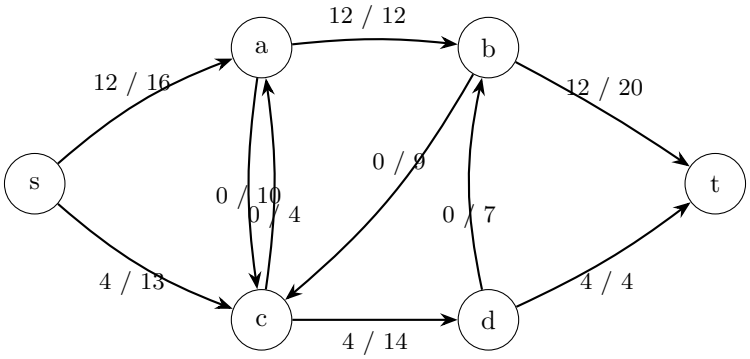
Network after Iteration 1




Residual graph after Iteration 1 (reverse edges in red)

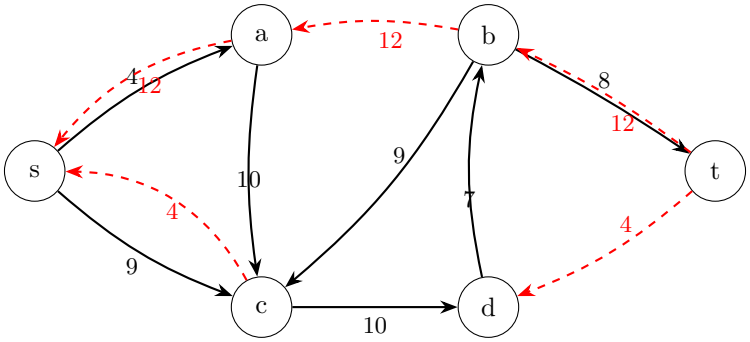
Residual Graph after Iteration 1

## After Iteration 2 (additional push 4 along $s \to c \to d \to t$)




Flow network after Iteration 2 (flows shown as flow / capacity)

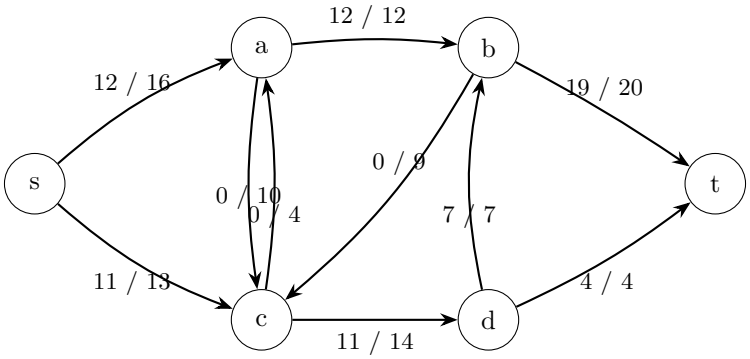
Network after Iteration 2




Residual graph after Iteration 2 (reverse edges in red)

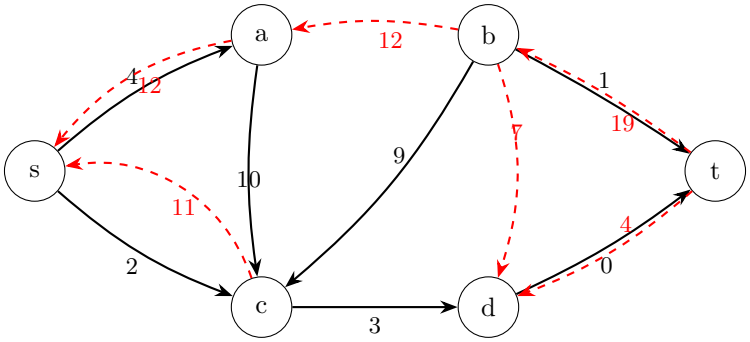
Residual Graph after Iteration 2

## After Iteration 3 (additional push 7 along $s \to c \to d \to b \to t$)




Flow network after Iteration 3 (final flows shown as flow / capacity)

Network after Iteration 3 (final)




Residual graph after Iteration 3 (final). Reverse edges are dashed red. Total max flow = 23.

Residual Graph after Iteration 3 (final)

# 1 Question 6 Answer

| Step | d(A) | d(B) | d(C) | d(G) | d(E) | d(D) | d(F) |
|---|---|---|---|---|---|---|---|
| Init (after A) | 0 (-) | 1 (A) | 3 (A) | $\infty$ | $\infty$ | $\infty$ | 10 (A) |
| After B chosen | 0 | 1 | 2 (B) | 3 (B) | 6 (B) | 8 (B) | 10 (A) |
| After C chosen | 0 | 1 | 2 | 3 | 5 (C) | 8 | 10 |
| After G chosen | 0 | 1 | 2 | 3 | 5 | 8 | 10 |
| After E chosen | 0 | 1 | 2 | 3 | 5 | 7 (E) | 7 (E) |
| After D chosen | 0 | 1 | 2 | 3 | 5 | 7 | 7 |
| After F chosen | 0 | 1 | 2 | 3 | 5 | 7 | 7 |

# Distance Table (summary per iteration)

| Iteration | Known set | d(A) | d(B) | d(C) | d(G) | d(E) | d(D) | d(F) |
|---|---|---|---|---|---|---|---|---|
| Init (after A relax) | {A} | 0 | 1 | 3 | $\infty$ | $\infty$ | $\infty$ | 10 |
| After B chosen | {A,B} | 0 | 1 | 2 | 3 | 6 | 8 | 10 |
| After C chosen | {A,B,C} | 0 | 1 | 2 | 3 | 5 | 8 | 10 |
| After G chosen | {A,B,C,G} | 0 | 1 | 2 | 3 | 5 | 8 | 10 |
| After E chosen | {A,B,C,G,E} | 0 | 1 | 2 | 3 | 5 | 7 | 7 |
| After D chosen | {A,B,C,G,E,D} | 0 | 1 | 2 | 3 | 5 | 7 | 7 |
| Final (F chosen) | {A,B,C,G,E,D,F} | 0 | 1 | 2 | 3 | 5 | 7 | 7 |

Table of tentative distances after each iteration. Distances show the current best known distance from A.

**Marked order (vertices marked known):**
A, B, C, G, E, D, F

**Final shortest path A $\rightarrow$ F:**
$A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$ with total cost $1 + 1 + 3 + 2 = 7$.