TCS NQT 2025

20th march shift 2

Ques 1:

Sum of All Prime Numbers Between 2 and n

Given an integer n, return the sum of all prime numbers between 2 and n (inclusive).

Constraints:

- $2 \leq n \leq 10^6$

Input:

- **A single integer n representing the upper limit.**

Output:

- **Return an integer representing the sum of all prime numbers between 2 and n.**

Example 1:
Input: 10
Output: 17
Explanation: The prime numbers between 2 and 10 are [2, 3, 5, 7]. Their sum is 2 + 3 + 5 + 7 = 17.

Example 2:
Input: 2
Output: 2
Explanation: 2 is the only prime number between 2 and 2.

C++

```cpp
#include <iostream>
#include <cmath>
using namespace std;

bool isPrime(int num) {
    if (num < 2) return false;
    for (int i = 2; i <= sqrt(num); i++) {
        if (num % i == 0) return false;
    }
    return true;
}

int sumOfPrimes(int n) {
    int sum = 0;
    for (int i = 2; i <= n; i++) {
        if (isPrime(i)) {
            sum += i;
        }
    }
    return sum;
}

int main() {
    int n;
    cout << "Enter n: ";
    cin >> n;
    cout << sumOfPrimes(n) << endl;
    return 0;
}
```

PYTHON

```
import math

def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

def sum_of_primes(n):
    return sum(num for num in range(2, n+1) if is_prime(num))

# Example usage
n = int(input("Enter n: "))
print(sum_of_primes(n))
```

JAVA

```
import java.util.*;
public class Main
{

   public static boolean isPrime(int num) {
      if (num < 2) return false;
      for (int i = 2; i <= Math.sqrt(num); i++) {
         if (num % i == 0) return false;
      }
      return true;
   }

   public static int sumOfPrimes(int n) {
      int sum = 0;
      for (int i = 2; i <= n; i++) {
         if (isPrime(i)) {
            sum += i;
         }
      }
      return sum;
   }

 public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter n: ");
      int n = sc.nextInt();
      System.out.println(sumOfPrimes(n));
 }
}
```

MASSIVE SUCCESS RATE

"Transform Your Interview Opportunity into an Offer Letter and Make Your Parents Proud!"

- In-depth Technical Mock
  - Crack coding challenges with real experts.
- HR & Managerial Prep
  - Master behavioral questions and impress Cognizant Interviewer.
- Full Interview Simulation
  - Ace both technical and HR in one session.
- Resume Review
  - Identify and fix weaknesses for a standout CV.
- Personalized Feedback & Expert Guidance
  - Tailored improvement tips to boost success.

www.primecoding.in

Ques 2:

Problem Statement

You are given a Directed Acyclic Graph (DAG) with N vertices and M edges. Each edge has a weight associated with it. Your task is to find the shortest path from a given source node (SRC) to a destination node (DES) using the provided graph data.

Input Format

1. **N M — Two integers representing the number of vertices (N) and the number of edges (M).**
2. **X Y W — Three integers for each edge representing an edge from vertex X to vertex Y with weight W.**
3. **SRC DES — Two integers representing the source node (SRC) and the destination node (DES).**

Output Format

- **Print the shortest path from SRC to DES.**
- **Display the total weight of this path.**

Example Input

3 3
0 1 5
1 2 3
0 2 10
0 2

Example Output

Path: 0 -> 1 -> 2, Total Weight: 8

PYTHON

```
import heapq

def dijkstra(graph, src, dest, n):
    dist = [float('inf')] * n
    visited = [False] * n
    dist[src] = 0
    pq = [(0, src)]

    while pq:
        weight, node = heapq.heappop(pq)

        if node == dest:
            print(f"{node}", end="")
            break
        print(f"{node}->", end="")
        if visited[node]:
            continue
        visited[node] = True

        for neigh, w in graph[node]:
            if dist[neigh] > dist[node] + w:
                dist[neigh] = dist[node] + w
                heapq.heappush(pq, (dist[neigh], neigh))

    return dist[dest]

if __name__ == "__main__":
    n, m = map(int, input().split())
    graph = [[] for _ in range(n)]

    for _ in range(m):
        x, y, w = map(int, input().split())
        graph[x].append((y, w))

    src, dest = map(int, input().split())
    total_weight = dijkstra(graph, src, dest, n)
    print(f", Total Weight: {total_weight}")
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef pair<int, int> pii;

void dijkstra(vector<vector<pii>>& graph, int src, int dest, int n) {
    vector<int> dist(n, INT_MAX);
    vector<bool> visited(n, false);
    priority_queue<pii, vector<pii>, greater<pii>> pq;

    dist[src] = 0;
    pq.push({0, src});

    while (!pq.empty()) {
        int weight = pq.top().first;
        int node = pq.top().second;
        pq.pop();

        if (node == dest) {
            cout << node;
            break;
        }
        cout << node << " -> ";

        if (visited[node]) continue;
        visited[node] = true;

        for (auto& neighbor : graph[node]) {
            int neigh = neighbor.first;
            int w = neighbor.second;

            if (dist[neigh] > dist[node] + w) {
                dist[neigh] = dist[node] + w;
                pq.push({dist[neigh], neigh});
            }
        }
    }

    cout << ", Total Weight: " << dist[dest] << endl;
}

int main() {
    int n, m;
    cin >> n >> m;

    vector<vector<pii>> graph(n);

    for (int i = 0; i < m; i++) {
        int x, y, w;
        cin >> x >> y >> w;
        graph[x].push_back({y, w});
    }

    int src, dest;
    cin >> src >> dest;

    dijkstra(graph, src, dest, n);

    return 0;
}
```

JAVA

```java
import java.util.*;

class Pair {
  int node;
  int distance;

  public Pair(int distance, int node) {
    this.node = node;
    this.distance = distance;
  }
}

public class Main {
  public static void dijkstra(int V, ArrayList<ArrayList<int[]>> adj, int src, int dest) {
    int[] dist = new int[V];
    int[] parent = new int[V];
    Arrays.fill(dist, Integer.MAX_VALUE);
    Arrays.fill(parent, -1);

    PriorityQueue<Pair> pq = new PriorityQueue<>((a, b) -> a.distance - b.distance);

    dist[src] = 0;
    pq.add(new Pair(0, src));

    while (!pq.isEmpty()) {
      Pair current = pq.poll();
      int node = current.node;
      int weight = current.distance;

      if (node == dest) break;

      for (int[] edge : adj.get(node)) {
        int neighbor = edge[0];
        int edgeWeight = edge[1];

        if (dist[node] + edgeWeight < dist[neighbor]) {
          dist[neighbor] = dist[node] + edgeWeight;
          parent[neighbor] = node;
          pq.add(new Pair(dist[neighbor], neighbor));
        }
      }
    }

    printPath(parent, src, dest);
    System.out.println(", Total Weight: " + dist[dest]);
  }

  public static void printPath(int[] parent, int src, int dest) {
    List<Integer> path = new ArrayList<>();
    for (int at = dest; at != -1; at = parent[at]) {
      path.add(at);
    }
    Collections.reverse(path);

    System.out.print("Path: ");
    for (int i = 0; i < path.size(); i++) {
      if (i > 0) System.out.print(" -> ");
      System.out.print(path.get(i));
    }
  }

  public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  int V = sc.nextInt();
  int E = sc.nextInt();

  ArrayList<ArrayList<int[]>> adj = new ArrayList<>();
  for (int i = 0; i < V; i++) adj.add(new ArrayList<>());

  for (int i = 0; i < E; i++) {
  int u = sc.nextInt();
  int v = sc.nextInt();
  int w = sc.nextInt();
  adj.get(u).add(new int[]{v, w});
  }

  int src = sc.nextInt();
  int dest = sc.nextInt();

  dijkstra(V, adj, src, dest);
  }
}
```

leetcode playground: CLICK HERE

Codes are available in all languages