Experiment 9

Student Name: Tanmaya Kumar Pani **UID: 22BCS12986**

Branch: BE-CSE **Section/Group: IOT-613B**

Semester: 5 **Date of Performance: 15/10/2024**

Subject Name: AP Lab **Subject Code: 22CSH-311**

1. **Aim:** Sub string game.

2. **Objective:** The Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string. Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo 109+7.

3. **Algorithm:**

a) Initialization:
- Initialize a variable totalSum = 0 to store the cumulative sum of all substring integers.
- Let currentSum = 0 to store the sum for the substrings ending at each position.
- Set MOD = 10^9 + 7 to handle large numbers.

b) Iterate Through the String:
- For each character at position i in the string (from left to right):
  - Convert the character to its integer value.
  - Update the currentSum as: currentSum=(currentSum×10+(i+1)×digit)%MODcurrentSum = (currentSum \times 10 + (i+1) \times \text{digit}) \% MODcurrentSum=(currentSum×10+(i+1)×digit)%MOD
  - Add currentSum to totalSum and take modulo MOD.

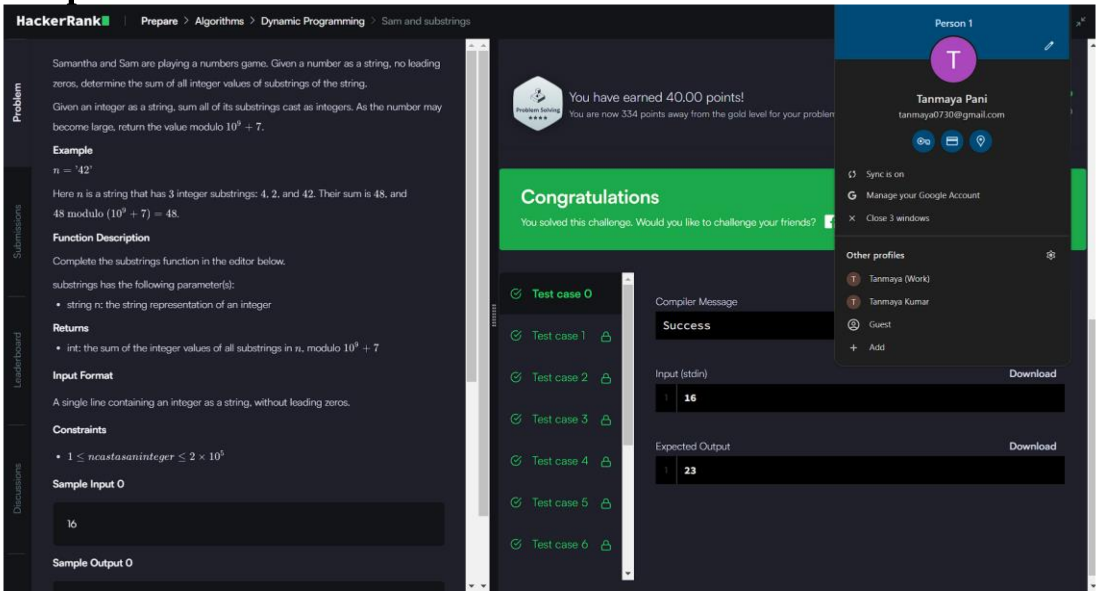
c) Return the Result:
- After processing all characters, return totalSum \% MOD.

4. **Code :**

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

const long long MOD = 1000000007;

long long sumOfSubstrings(string number) {
   int n = number.length();
   long long totalSum = 0;
   long long currentSum = 0;
```

```
    for (int i = 0; i < n; i++) {
        int digit = number[i] - '0';  // Convert character to integer
        currentSum = (currentSum * 10 + (i + 1) * digit) % MOD;
        totalSum = (totalSum + currentSum) % MOD;
    }

    return totalSum;
}

int main() {
    string number;
    cin >> number;

    cout << sumOfSubstrings(number) << endl;
    return 0;
}
```

5. Output:



Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Example

$n =$ '42'

Here n is a string that has 3 integer substrings: 4, 2, and 42. Their sum is 48, and $48 \text{ modulo } (10^9 + 7) = 48$.

Function Description

Complete the substrings function in the editor below.

substrings has the following parameter(s):

- string n: the string representation of an integer

Returns

- int: the sum of the integer values of all substrings in n, modulo $10^9 + 7$

Input Format

A single line containing an integer as a string, without leading zeros.

Constraints

- $1 \leq ncastasaninteger \leq 2 \times 10^5$

Sample Input 0

```
16
```

Sample Output 0

You have earned 40.00 points!
You are now 334 points away from the gold level for your problem

Congratulations
You solved this challenge. Would you like to challenge your friends?

Tanmaya Pani
tanmaya0730@gmail.com

Compiler Message

```
Success
```

Input (stdin)

```
16
```

Expected Output

```
23
```

6. (a) Time Complexity : O(n)
(b) Space Complexity : O(1)

7. Learning Outcomes :

a) Understood how to work with large numbers using the modulo operator to avoid overflow issues, especially in competitive programming.

b) Learnt an optimized way to calculate sums of all substrings of a number string by iterating over each digit once, updating the sum progressively.

c) Practiced managing large integers and input strings in Java.

Problem -2

1. **Aim :** Kingdom Division

2. **Objective :** King Arthur has a large kingdom that can be represented as a tree, where nodes correspond to cities and edges correspond to the roads between cities. The kingdom has a total of n cities numbered from 1 to n. The King wants to divide his kingdom between his two children, Reggie and Betty, by giving each of them 0 or more cities; however, they don't get along so he must divide the kingdom in such a way that they will not invade each other's cities. The first sibling will invade the second sibling's city if the second sibling has no other cities directly connected to it.

3. **Algorithm :**

a) Input:
- n: Number of cities.
- An array of edges representing the roads between the cities.

b) Tree Construction:
- Construct an adjacency list representing the tree using the given edges.

c) Subtree Sizes:
- Perform a DFS (Depth-First Search) traversal to calculate the size of the subtree for each node. The subtree size of a node is the number of cities (nodes) in the subtree rooted at that node.

d) Optimal Edge Removal:
- For each edge, calculate the size of the subtree it would create if the edge were cut.
- Calculate the difference between the sizes of the two resulting subtrees.
- Find the edge that minimizes this difference.

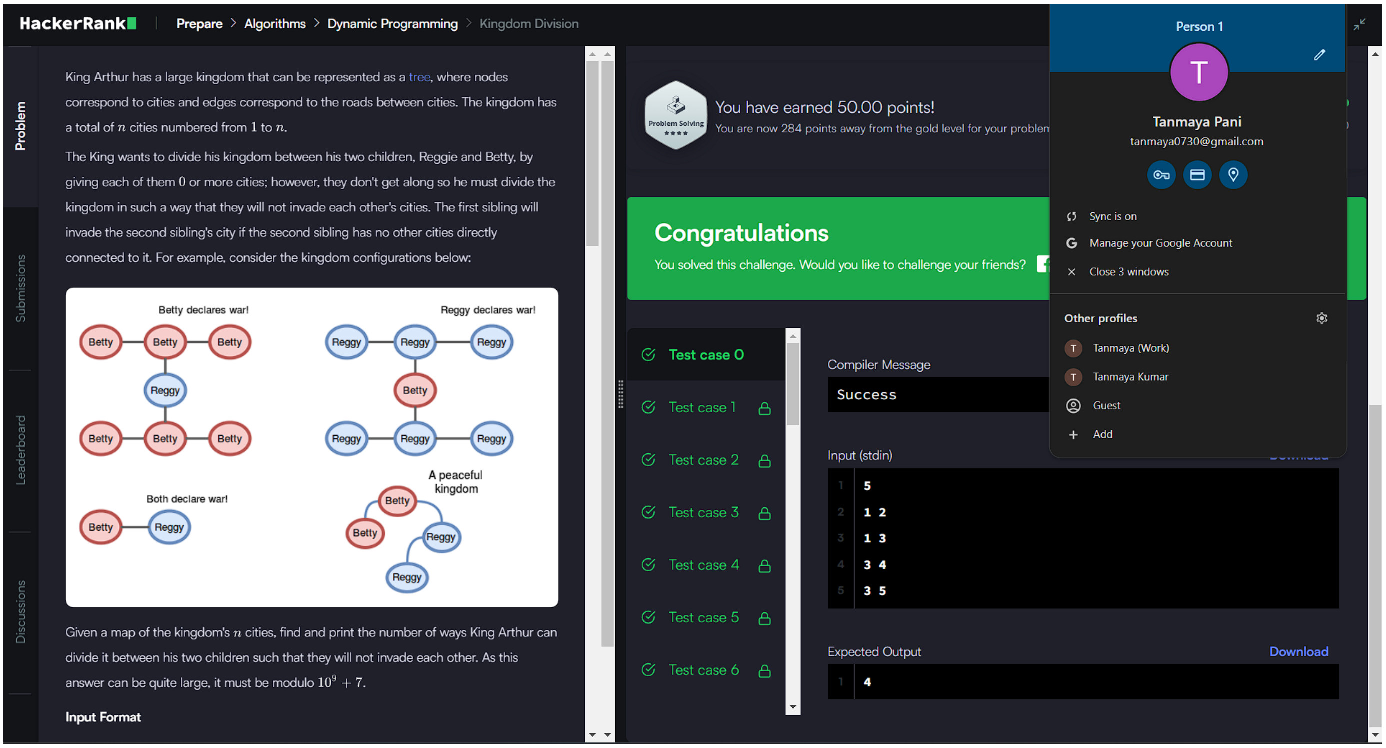
e) Return the result:
- Return the best edge to cut.

4. **Code :**

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

const int MOD = 1000000007;
vector<vector<int>> tree;
vector<bool> visited;

// Function to calculate the number of ways for the subtree
pair<long long, long long> ways(int node, int parent) {
    long long lonely_ways = 1;  // Ways to assign subtree such that all children have opposite assignments
    long long total_ways = 1;   // Total ways to assign the subtree
    for (int child : tree[node]) {
        if (child != parent) {
            pair<long long, long long> child_ways = ways(child, node);
            long long diff = child_ways.first;
            long long same = child_ways.second;

            lonely_ways = (lonely_ways * diff) % MOD;
            total_ways = (total_ways * (same + diff) % MOD) % MOD;
        }
    }

    long long valid_ways = (total_ways - lonely_ways + MOD) % MOD;
    return {valid_ways, total_ways};
}

int kingdomDivision(int n, vector<pair<int, int>>& roads) {
    tree.resize(n + 1);
    visited.resize(n + 1, false);

    // Build the adjacency list for the tree
    for (auto road : roads) {
        int u = road.first, v = road.second;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }

    // Call DFS from node 1 (root) assuming node 1 has no parent (-1)
    pair<long long, long long> result = ways(1, -1);

    // Multiply the result for the root assignment
    return (2 * result.first) % MOD;
}

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

    vector<pair<int, int>> roads(n - 1);
    for (int i = 0; i < n - 1; i++) {
        int u, v;
        cin >> u >> v;
        roads[i] = {u, v};
    }

    cout << kingdomDivision(n, roads) << endl;
    return 0;
}
```

5. **a)Time Complexity :** O(n)
   b)Space Complexity : O(n)

6. Output :



King Arthur has a large kingdom that can be represented as a tree, where nodes correspond to cities and edges correspond to the roads between cities. The kingdom has a total of n cities numbered from 1 to n.

The King wants to divide his kingdom between his two children, Reggie and Betty, by giving each of them 0 or more cities; however, they don't get along so he must divide the kingdom in such a way that they will not invade each other's cities. The first sibling will invade the second sibling's city if the second sibling has no other cities directly connected to it. For example, consider the kingdom configurations below:

Given a map of the kingdom's n cities, find and print the number of ways King Arthur can divide it between his two children such that they will not invade each other. As this answer can be quite large, it must be modulo $10^9 + 7$.

Input Format

You have earned 50.00 points!

You are now 284 points away from the gold level for your problem

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

Input (stdin)

```
5
1 2
1 3
3 4
3 5
```

Expected Output

```
4
```

7. Learning Outcomes :

a) Applying DFS to calculate subtree sizes and find optimal edge removal in a tree structure.

b) Learning to optimize division by minimizing differences in the sizes of divided parts.

c) Applying graph-theoretical concepts (trees, edge cuts) to solve practical problems like kingdom division.