Find all the stepping numbers in a given range.

AMIT JAIN (IIT2019150), AYUSH TIWARI (IIT2019151), MRINAL BHAVE(IIT2019152)

IV Semester, B.Tech, Department of Information Technology,

Indian Institute of Information Technology Allahabad, Prayagraj, India

Abstract— **In this paper we will be using brute force approach and breadth first search to find the stepping numbers in a given range.**

Keywords— graph, node, bfs, brute force, stepping numbers, range.

I. INTRODUCTION

Stepping numbers are those numbers in which all the adjacent digits have an absolute difference of 1, i.e. if in a number there is even one digit whose absolute difference with its adjacent digits is not equal to 1, it is not a stepping number.

An efficient solution for finding stepping numbers in a given range is to use breadth first search on a graph. A Graph is a non-linear data structure which consists of a finite set of vertices (or nodes) and a set of edges which connect a pair of nodes. Graphs can be traversed using techniques like Breadth First Search (BFS) and Depth First Search (DFS).

BFS Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration. It systematically explores the edges of a graph to "discover" every vertex that is reachable from the source vertex. BFS works both on directed and undirected graphs.

Illustration

Let's consider that the value of numbers n and m is 25 and 47 respectively. Then we will find stepping numbers in the range [25, 47].
Value of n and m : 25, 47
Range : [25, 47]

All the stepping numbers in this range are 32, 34, 43 and 45.
Thus, the answer would be 4.

II. ALGORITHM DESIGN

1.Brute Force Approach

Approach: To find all stepping numbers between n and m, we use a loop and traverse in the range n to m. For each number in the loop, we check if the number is a stepping number or not by determining the absolute difference between its adjacent digits.

Algorithm:

Step.1) Run a loop from n to m and for each number (i) in between n and m we call the check function and pass it.If check function returns true we print the number otherwise we move to the next number.

Step.2) In check function we extract all digits of the number and check if the difference between all adjacent digits is one or not .If we find two adjacent digits whose absolute difference is not 1 we return false otherwise we return true.

2.Using Breadth First Search

Approach: We will construct a graph, where each node represents a stepping number. Consider nodes A and B. There will be a directed edge from node A to B, if a node B can be transformed from node A.
Since every single digit number is a stepping number, the BFS traversal on the graph consisting of nodes 0 - 9 will give us all the stepping numbers starting from a particular digit. So we will do BFS traversal for all digits from [0, 9] which will give us the desired stepping numbers.

Algorithm:

Step.1) As all numbers from 0 to 9 are stepping number call bfs for every number from 1 to 9.For node 0, no need to explore neighbors during BFS traversal since it will lead to 01, 012, 010 and these will be covered by the BFS traversal starting from node 1.

Step.2) For each node U in graph do bfs traversal for its adjacent nodes if they are in range from n to m.Adjacent node V is created using last digit of U.lastDigit refers to last digit of U.Adjacent number can be U*10+lastDigit-1 or U*10+lastDigit+1.If lastDigit is 9 we do bfs traversal only for U*10+lastDigit-1 or if lastDigit is 0 we do bfs traversal only for U*10+lastDigit+1 otherwise we do bfs traversal for both the adjacent nodes.

Illustration to find all stepping numbers between 0 and 21 using BFS approach

Step.1)0 is a stepping Number and it is in the range so display it.

Step.2)1 is a Stepping Number, find neighbors of 1 i.e.,10 and 12 and push them into the queue.
To find 10 and 12 here U is 1 and last Digit is also 1
V = 10 + 0 = 10 (Adding lastDigit - 1)
V = 10 + 2 = 12 (Adding lastDigit + 1)

Step.3)Then we do the same for 10 and 12 this will result in 101, 123, 121 but these numbers are out of range.
Now any number transformed from 10 and 12 will result into a number greater than 21 so no need to explore their neighbors.

Step.4) 2 is a Stepping Number, find neighbors of 2 i.e. 21, 23.23 is out of range so it is not considered as a Stepping Number (Or a neighbor of 2).The other stepping numbers will be 3, 4, 5, 6, 7, 8, 9.

Pseudo code to find all stepping numbers in the given range

Pseudo code for brute force approach

function to check if x is a stepping number or not.

```
bool isStepping(x)
{

   we will initialize the preDigit variable  as x%10;
  Then iterate through the all digits of x
     while (x)
  {
         we will initialize curDigit as x%10;

            if the absolute diff of preDigit and curDigit is not equal to 1
           then
           return false
         else

      prevDigit = curDigit;
      and
      x=x/10;

  }

  return true;
}

for(i n to m)
{
 if(isStepping(i) )
  print(i);
}
```

Pseudo code for B.F.S.

```
bfs( n, m, num)
 where n is starting point and m is last point and num is root node for every bfs graph.

 queue to store stepping no.
   queue q

   q.enqueue(num);
    insert num in queue until all its neighbour vertices are marked.

   while (queue is not empty)
   {
      Get the front element of the queue
      front = q.front();

          Removing that vertex from queue , whose neighbour will be visited now.
      q.pop();

        If the Stepping Number is in the range n, m then print it.
      Print(front);

            If Stepping Number is 0 or greater than m,  no need to explore the neighbors so break the loop


       Get the last digit of the currently visited Stepping Number
      lastDigit = front % 10;

            There can be 2 stepping no. which can be derived from a stepping no.
          A = front * 10 + (lastDigit- 1);
          B = front * 10 + (lastDigit + 1);

       If lastDigit is 0 then only possible digit after 0 can be 1
      if (lastDigit == 0)
          q.push(B);

       If lastDigit is 9 then only possible digit after 9 can be 8
       if (lastDigit == 9)
```

```
        q.push(A);

    else in the case of any other digit A and
B both will be stepping number
        q.push(A);
        q.push(B);



void display(n,m)
{
   for every i find the stepping no. associated
to it using B.F.S.
  for (i 0 to 9)
     bfs(n, m, i);
}
```

III. TIME COMPLEXITY ANALYSIS AND COMPARISON

Time Complexity :

Brute Force Approach-O((m-n)*log(m)) as we traverse from n to m which takes O(m-n) time and for every number i we find the difference between all adjacent digits which takes O(log(i)) time. So overall time complexity is O((m-n)*log(m)).

Breadth First Search Approach-O(m) as we do bfs traversal so time complexity depends on maximum number of nodes in graph which can be O(m).

Space Complexity :

Brute Force Approach-O(1) as we don't take any extra space to store so space complexity is O(1).

Breadth First Search Approach - O(log m) as we use a queue to store numbers and do bfs traversal using this queue.This queue can have maximum size of O(2*log(m)) which is of order log m.

IV. EXPERIMENTAL COMPLEXITY ANALYSIS

	Time(in micro sec.)				
	1	2	3	4	5
n, m	0 ,21	4,45	5,60	6,700	100,1000
time(Brute force)	3634	3871	3998	4199	5297
time(B.F.S)	2562	2333	3000	3655	4025

V. CONCLUSION

In this paper, we have concluded that we can solve the given problem using brute force approach in O((M-N)*log(M)) time and using B.F.S. time complexity is O(log(m)).

VI. REFERENCES

- T.H. Cormen, C.E. Leiserson, R.A. Riveset. Introduction to Algorithms, PHI/EEE.
- Jeffrey J. Mc Connel. Analysis of algorithms: An active learning approach, Narosa.
- R.G. Dromey. How to solve it by Computer.
- Anany levitin. Design and Analysis of Algorithms, Pearson Education.
- geeksforgeeks.org