Indian Institute of Information Technology Allahabad DAA Assignment 06

Debasish Das
IIB2019031

Surya Kant
IIB2019032

Mohammad Monish
IIB2019033

***Abstract*—In this report we have discussed a Algorithm to find the shortest distance between two prime numbers by changing a single digit at a time .**

I. Problem Statement

Shortest path to reach one prime to other by changing single digit at a time. Given two four-digit prime numbers, suppose 1033 and 8179, we need to find the shortest path from 1033 to 8179 by altering only single digit at a time such that every number that we get after changing a digit is prime. For example, a solution is 1033, 1733, 3733, 3739, 3779, 8779, 8179

II. Keywords

- Graph
- BFS
- Sieve of Eratosthenes

III. Introduction

This Problem requires a good understanding of graph traversal (BFS) and Sieve of Eratosthenes.Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. It uses the opposite strategy of depth-first search, which instead explores the node branch as far as possible before being forced to backtrack and expand other nodes.
Sieve of Eratosthenes is an algorithm for finding all the prime numbers in a segment (form 1 to n) using $O(nloglogn)$ operations. The algorithm is very simple: at the beginning we write down all numbers between 2 and n. We mark all proper multiples of 2 (since 2 is the smallest prime number) as composite. A proper multiple of a number x, is a number greater than x and divisible by x. Then we find the next number that hasn't been marked as composite, in this case it is 3. Which means 3 is prime, and we mark all proper multiples of 3 as composite. The next unmarked number is 5, which is the next prime number, and we mark all proper multiples of it. And we continue this procedure until we processed all numbers in the row.

IV. Algorithm Design

Following is step by step approach of algorithm(s) used in our program :

- Find and store all the prime numbers of 4 digit using Sieve of Eratosthenes
- Make a graph using those prime numbers. Vertex will be the prime numbers and two primes u,v will have an edge if they can be converted into one another by changing a single digit. Ex. 1373 and 1303 will have an edge connecting both the nodes in our graph.
- If we have to find the shortest distance between a and b. Then we will root our graph at node a and initialize its distance as 0.
- Then apply bfs on our graph to find the shortest distance between given two prime numbers. The relation used will be distance[child] = distance[parent] + 1, where child and parent have a direct edge and parent has a height smaller than child.

V. Algorithm Analysis

Let N be the largest 4 digit number, P be the number of prime numbers of 4 digits.

Time complexity:
We need three functions to solve this problem,

1) **sieve():** : to generate prime numbers up to N. it take $O(N(log(log(N))))$ time.
2) **makeGraph() :** to add edges in our graph. It takes $O(P^2)$ time.
3) **bfs :** to find shortest distance between given two 4 digit primes. It takes $O(N + P^2)$ time.

Therefore, the time complexity of our program will be : $O(N(log(log(N))) + P^2)$

Space complexity:

1) **sieve():** : to store prime numbers up to N. it take $O(N)$ space.
2) **makeGraph() :** to add edges in our graph. At max we will have $P * (P - 1)/2$ edges. So storing N nodes and P^2 edges Which is $O(N + P^2)$ space.

3) **bfs :** in bfs, our queue can have a maximum of P values at a time. So, It takes $O(P)$ time.

Therefore, the space complexity of our program will be : $O(N + P^2)$

VI. APPLICATIONS

Graphs are widely used to represent the flow of computation . In World Wide Web, web pages are considered to be the vertices. There is an edge from a page u to other page v if there is a link of page v on page u. This is an example of Directed graph. It was the basic idea behind Google Page Ranking Algorithm.Few important real life applications of graph data structures are:

- Facebook: Each user is represented as a vertex and two people are friends when there is an edge between two vertices. Similarly friend suggestion also uses graph theory concept.
- Google Maps: Various locations are represented as vertices and the roads are represented as edges and graph theory is used to find shortest path between two nodes.
- Recommendations on e-commerce websites: The “Recommendations for you” section on various e-commerce websites uses graph theory to recommend items of similar type to user’s choice
- Graph theory is also used to study molecules in chemistry and physics.
- In Operating System, we come across the Resource Allocation Graph where each process and resources are considered to be vertices. Edges are drawn from resources to the allocated process, or from requesting process to the requested resource. If this leads to any formation of a cycle then a deadlock will occur.

VII. Acknowledgement

We are very much grateful to Course instructor Mr. Rahul Kala and our mentor, Mr. Md Meraz, who have provided the great opportunity to do this wondurful work on this subject of Design and Algorithm Analyis specifically on methodologies of Divide and Conquer.

VIII. Conclusion

We have solved our problem using BFS and Sieve of Eratosthenes, which will take over all time of $O(V + E)$.

IX. References

1.https://en.wikipedia.org/wiki/Breadth-first_search
2. https://en.wikipedia.org/wiki/Breadth-first_search
3. Cormen, Leiserson, Rivest, and Stein (2009). Introduction to Algorithms, 3rd edition.

Appendix

Code for implementation of this paper is given below:

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 100009;
vector<int>primes;
vector<vector<int>>graph;
void sieve(){
    vector<bool>isPrime(maxn+1,1);
    for(int i=2;i*i<=maxn;++i){
        if(isPrime[i]){
            for(int j=i*i;j<=maxn;j+=i){
                isPrime[j] = 0;
            }
        }
    }
    for(int i=1000;i<=9999;++i){
        if(isPrime[i]){
            primes.push_back(i);
        }
    }
}
int diffDigits(int a,int b){
    int diff = 0;
    while(a>0){
        int r1 = a%10;
        int r2 = b%10;
        diff+=(r1!=r2);
        a/=10;
        b/=10;
    }
    return (diff);
}
void makeGraph(){
    graph = vector<vector<int>>(maxn+1);
    for(int i=0;i<primes.size();++i){
        for(int j=i+1;j<primes.size();++j){
            int p1 = primes[i];
            int p2 = primes[j];
            if(diffDigits(p1,p2)!=1){
                continue;
            }
            graph[p1].push_back(p2);
            graph[p2].push_back(p1);
        }
    }
}
void shortestDistance(int a,int b){
    queue<int>q;
    q.push(a);
    vector<int>dist(100009,0);
    vector<bool>vis(100009,0);
    vector<int>parent(100009);
    parent[a] = a;
    while(!q.empty()){
        int u = q.front();
        q.pop();
        for(auto & child : graph[u]){
            if(!vis[child]){
                parent[child]=u;
                vis[child] = true;
                dist[child] = dist[u]+1;
                if(child==b){
                    break;
                }
                q.push(child);
            }
        }
    }
    vector<int>path;
    int x = b;
    while(parent[b]!=a){
        path.push_back(b);
        b = parent[b];
    }
    path.push_back(b);
    path.push_back(a);
    reverse(path.begin(),path.end());
    cout<<"Shortest Distance : "<<dist[x]<<"\n";
    cout<<"Path\n";
    cout<<a<<" ";
    for(int i=0;i<path.size();++i)cout<<"-> "<<path[i];
    cout<<'\n';
}
bool isValid(int a,int b){
    int i1 = lower_bound(primes.begin(),primes.end(),a) - primes.begin();
    int i2 = lower_bound(primes.begin(),primes.end(),b) - primes.begin();
    return (i1<primes.size() && i2<primes.size() && primes[i1]==a && primes[i2]==b);
}
int main(){
    sieve();
    makeGraph();
    // 1033 8179
    // 1373 8017
    // 1033 1033
    cout<<"Enter two 4 digit prime numbers\n";
    int a,b;
    cin>>a>>b;
    // a = 1373;
    // b = 8017;
    while(!isValid(a,b)){
        cout<<"Invalid input, Please enter two prime numbers between 1000 and 9999 (ex. 1373 8017)\n";
        cin>>a>>b;
    }
    if(a==b){
        cout<<"0\n";
    }
    else{
        shortestDistance(a,b);
    }
}
```

Listing 1. Code for this paper