লিংকগুলো

- টিউটোরিয়াল
- প্রোগ্রামিং রিসোর্সেস

শর্টেস্ট পাথ - প্রবলেম নিয়ে বকর বকর

প্যাট-প্যাটানি

> *Prime numbers are what is left when you have taken all the patterns away. I think prime numbers are like life. --- Mark Haddon (The Curious Incident of the Dog at Night Time)*

উপরের কোটেশনটার সাথে শর্টেস্ট পাথের তেমন কোন সম্পর্ক নেই। কিন্তু কোটেশনটা দেখার পর আমার লিখতে ইচ্ছা করলো, সেজন্য আমি কোটেশনটা তুলে দিলাম।

সত্যিকারের পৃথিবীতে তুমি প্রচুর প্যাটার্ন খুঁজে পাবে। ধরো তুমি যদি একটা গ্যাস বেলুন কিনে সেটাকে ছেড়ে দাও, সেটা সবসময়ই আকাশের দিকে উঠে যাবে। তুমি যদি আরেকটা গ্যাস বেলুন কিনে একই কাজ করো, একই জিনিস ঘটবে। তো এটা হচ্ছে একটা প্যাটার্ন। প্যাটার্ন বলতে আমি বুঝাচ্ছি একই ধাঁচের ঘটনা। ধরো একটা জায়গায় আগুন লাগলো, তুমি যদি সেখানে পানি ঠেলে দাও, আগুনটা নিভে যাবে। আরেকটা জায়গায় যদি আগুন লাগে তাহলে একই সলুশন কাজ করবে। কারণ দুইটা একই ধাঁচের ঘটনা। এখন তোমার শুধু একটা বালতি খুঁজে বের করতে হবে আর পানি এনে আগুনটায় ঢালতে হবে। এরপর যখনই তুমি একই প্যাটার্নটা দেখতে পাবে, তুমি অলরেডি এটার সলুশন জানো। বেশ তো, এখন ধরো একটা ছোট্ট আগুন না লেগে একটা আস্ত তিনতালা বিল্ডিং এ আগুন লেগে গেলো । এখন আর আমাদের বালতি সলুশন কাজ করবে না। আমরা পানিই ঢালবো কিন্তু সলুশনটা একটু পাল্টাতে হবে। বালতির বদলে ফায়ার সার্ভিসের গাড়ি লাগবে। কিন্তু মূল সলুশন একই থাকলো, আমরা পানিই ঢালবো।

বেশির ভাগ বাস্তব জগতের প্রবলেমে এই ধরণের প্যাটার্ন পাওয়া যায়। তুমি যদি একটু ভালোভাবে প্রবলেমটা বোঝার চেষ্টা করো, তুমি প্রবলেমের ধাঁচগুলো বুঝতে পারবে। তারপর তুমি এটাও বুঝতে পারবে যে এই প্রবলেমটার জন্য একটা সম্পূর্ণ নতুন সলুশন বের করার কোন দরকার নাই, তোমার শুধু এই প্রবলেমটাকে অন্য একটা প্রবলেমের প্যাটার্নে ফেলতে হবে যেই প্যাটার্নের সলুশন তুমি অলরেডি জানো । তারপর সলুশনটাকে পাল্টাতে হবে যাতে সলুশনটা এই প্রবলেমটার জন্যও কাজ করে।

কি রকম?

ধরো, একটা ট্রাক ড্রাইভার কুমিল্লা থেকে চুয়াডাঙা যাবে। যদি সে কিছু তেল বাঁচাতে পারে, তাইলে সেই তেল বেচে সে নারিকেল কিনতে পারবে। তো সে করবে কি, এমন ভাবে কুমিল্লা থেকে চুয়াডাঙা যাবে যাতে তার পথের দৈর্ঘ্য সবচে কম হয়। তোমার যদি আগের টিউটোরিয়ালটা পড়া থাকে, তুমি নিশ্চই বুঝতে পারছো, এখানে শহরগুলো হচ্ছে আমার node, আর শহরের মাঝের পথগুলো হচ্ছে আমার edge, আর পথের দৈর্ঘ্য হচ্ছে আমার cost। আমরা cost মিনিমাইজ করে দুইটা node এর মধ্যে একটা পথ খুঁজছি। তো এভাবে চিন্তা করলে এই প্রবলেমটা আপনাআপনিই শর্টেস্ট পাথ প্রবলেম হয়ে যাচ্ছে, যেটার সলুশন তুমি এর মধ্যেই জানো।

তারপর ধরো, একটা ট্রাভেল এজেন্ট তোমার জন্য দুবাই এর টিকেট কাটবে। তুমি পয়সা বাঁচাতে চাও, কারণ তুমি খুব গরিব। ট্রাভেল এজেন্ট এখন তোমার জন্য সবচে' সস্তা রুটটা বের করবে। এখন দেখো, যদি দুইটা শহরের মধ্যে একটা ফ্লাইট থাকে, তাইলে সেটা হচ্ছে আমাদের edge(অবশ্যই শহরগুলো হচ্ছে node), আর এই edge এর cost হচ্ছে টিকেট এর দাম। আমরা এখন দুইটা node এর মধ্যে একটা পথ খুঁজে বের করার চেষ্টা করছি যেখানে টিকেটের দাম (cost) সবচে' কম হবে।

ট্রাক ড্রাইভারের প্রবলেম আর দুবাইওয়ালার প্রবলেম, দুইটা শেষ পর্যন্ত ঘুরে ফিরে একই প্রবলেম হয়ে যাচ্ছে যখন তুমি প্যাটার্নটা দেখতে পাচ্ছো।

আরিকটু কঠিন উদাহরণ

(৪ লিটার মানে কিন্তু 4 liters)

তোমাকে তোমার বউ দুইটা বালতি ধরায় দিলো। একটা বালতি লাল আরেকটা বালতি নীল। লাল বালতির সাইজ হচ্ছে ৫ লিটার আর নীল বালতির সাইজ হচ্ছে ৩ লিটার। ওয়াশিং মেশিনে ঠিক ৪ লিটার পানি ঢালতে হবে একবারে একটা বালতি থেকে, কারণ ওয়াশিং মেশিনটা এত মোটকা যে ওটা কিছুতেই বাথরুমে ঢুকানো যাচ্ছা না। তুমি ট্যাপ থেকে ইচ্ছা মতো পানি নিতে পারো বালতিতে। এখন একটা বালতিতে ৪ লিটার পানি ক্যামনে বানানো যায়?

তুমি তিন ধরণের কাজ করতে পারো
১. একটা বালতিতে পুরোপুরি পানি ভরতে পারো
২. বালতি পুরাটা খালি করতে পারো
৩. একটা বালতি থেকে আরেকটা বালতিতে পানি ঢালতে পারো, তবে শর্ত হচ্ছে কোন পানি উপচাতে পারবে না, তাহলে তোমার সাধের কার্পেট নষ্ট হয়ে যাবে।

তোমাকে যদি আমি বলি এখখুনি এই কাজটা করে দাও সবচে' কম সময় নিয়ে, তাহলে প্রবলেমটা একটা শর্টেস্ট পাথ প্রবলেম হয়ে যায়!

এটুকু শোনার পর তোমার একটু ধাক্কা খাওয়া কথা। কারণ সোজাসুজি এখানে শহর দেখা যাচ্ছে না, পথও দেখা যাচ্ছে না, শর্টেস্ট পাথ কোথথেকে আসলো সেটা তো আরো পরের কথা। ঠিক এই জন্যই গ্রাফ থিওরির টার্মগুলো কাজের। সেগুলো তোমাকে সাহায্য করবে প্রবলেমটার একটা অ্যাবস্ট্রাক্ট শেপ তৈরী করতে।

এবার চোখ বন্ধ করে দুইটা বালতি কল্পনা করো। লাল বালতিটা নীলটার চে একটু বড়। দুইটায় পানি ভর্তি। এখন তুমি যদি তিন ঘন্টা ঢালাঢালির পর ক্লান্ত হয়ে আমাকে ফোন করো, তুমি ঠিক কি কি তথ্য দিবা আমাকে? অবশ্যই তুমি আমাকে বালতি দুইটার সাইজ বলবা। তারপর? এবার একটু ভাবো আর কি কি বললে আমি ঠিকঠাক মতো বুঝবো তুমি কোন অবস্থায় আছো।

অবশ্যই সেটা হচ্ছে তিন ঘন্টা ঢালাঢালির পর কোন বালতিতে কতটুকু পানি আছে। এটা বললে আমি পরিষ্কার বুঝতে পারবো তোমার অবস্থা কি। আমার আর তোমার বাসায় আসা লাগবে না। আমরা এটাকে state, সোজা বাংলায় অবস্থা।

ধরো কোন ভাবে আমার লাল বালতিতে ৩ লিটার পানি আছে আর নীল বালতিতে ২ লিটার। এই অবস্থা থেকে আর কোন কোন অবস্থায় যাওয়া যায় ?

১. লাল ৫ লিটার, নীল ০ লিটার (নীল বালতির সব পানি লাল বালতিতে)
২. লাল ২ লিটার, নীল ৩ লিটার (নীল বালতি ভর্তি করলাম লাল বালতি থেকে ঢেলে)
৩. লাল ০ লিটার, নীল ২ লিটার (লাল বালতি খালি করলাম)
৪. লাল ৩ লিটার, নীল ০ লিটার (নীল বালতি খালি করলাম)
৫. লাল ০ লিটার, নীল ০ লিটার (দুইটাই খালি করলাম)

তো এবার দেখো, তোমার একটা অবস্থা থেকে আরেকটা অবস্থায় যাওয়ার জন্য একটা move লাগতেসে। আমরা যদি অবস্থা গুলোকে শহর চিন্তা করি, তাহলে শহর (৩,২) থেকে পাঁচটা শহরে যাওয়া যায় (৫,০), (২,৩), (০,২), (৩,০), (০,০), এবং যেতে একটা move লাগে।

আমার প্রবলেম হচ্ছে আমাকে (০,০) শহর থেকে (x,৪) অথবা (৪,x) শহরে যেতে হবে সবচে' কম সংখ্যক move দিয়ে, যখন x যেকোন সংখ্যা হতে পারে। (একবার ৪ লিটার পানি পাইলে আর বাকি বালতি নিয়ে চিন্তা করার কোন দরকার নাই, তাই না?) হুমম, প্রবলেমটা কি চেনা চেনা লাগতেসে না? আমরা ঘুরে ফিরে আবার শর্টেস্ট পাথ প্রবলেমে ফিরে আসলাম।

এই প্রবলেমটা হচ্ছে হুবহু UVa 571। অবশ্য একই সাথে পথও বের করতে হয়। এই প্রবলেমটার আমার সলুশন হচ্ছে এটা। তুমি যদি কোড দেখে ভয় পাও, তাহলে ভূতরা কষ্ট পাবে।

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cctype>

#include<cmath>
#include<iostream>
#include<fstream>
#include<numeric>

#include<string>
#include<vector>
#include<queue>
#include<map>
#include<algorithm>
#include<set>
#include<sstream>
#include<stack>
#include<list>
#include<iterator>
using namespace std;
```

```
#define REP(i,n) for(__typeof(n) i=0; i<(n); i++)
#define FOR(i,a,b) for(__typeof(b) i=(a); i<=(b); i++)
#define CLEAR(t) memset((t), 0, sizeof(t))

#define sz size()
#define pb push_back
#define pf push_front

#define VI vector<int>
#define VS vector<string>
#define LL long long

#define INF (1<<30)
#define eps 1e-11

#define fillA 0
#define fillB 1
#define emptyA 2
#define emptyB 3
#define pourAB 4
#define pourBA 5

string s[] = { "fill A", "fill B", "empty A", "empty B", "pour A B", "pour B A" };
struct state { int a, b; };
int dist[1005][1005];
state parent[1005][1005];
int work[1005][1005];

void print( int xa, int xb ) {
    if( dist[xa][xb] != 0 ) {
        print( parent[xa][xb].a, parent[xa][xb].b );
        printf("%s\n",s[ work[xa][xb] ].c_str());
    }
}

int main() {
    state ux,u,v;
    int x;
    int n;
    while( scanf("%d%d%d", &ux.a, &ux.b, &n) == 3 ) {


        REP(i,1002) REP(j,1002) dist[i][j] = INF;
        u.a = 0; u.b = 0;
        dist[u.a][u.b] = 0;

        queue<state> q;
        q.push( u );
        while( !q.empty() ) {
            u = q.front();
            q.pop();
            //cout <<"processing "<< u.a <<" "<<u.b << endl;
            if( u.b == n ) {
                //cout <<u.a<<" "<<u.b<<endl;
                break;
            }


            // now the operations
            x = dist[u.a][u.b] + 1;

            //filling A
            v.a = ux.a;
            v.b = u.b;

            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = fillA;
            }

            // filling B
            v.a = u.a;
            v.b = ux.b;

            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = fillB;
            }

            // empty A
            v.a = 0;
            v.b = u.b;

            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = emptyA;
            }

            // empty B
            v.a = u.a;
            v.b = 0;

            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
```

```
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = emptyB;
            }


            //pour A -> B
            if( u.a+u.b >= ux.b ) {
                v.a = u.a - ( ux.b - u.b );
                v.b = ux.b;
            }else {
                v.b = u.b + u.a;
                v.a = 0;
            }



            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = pourAB;
            }

            // pour B -> A

            if( u.a+u.b >= ux.a ) {
                v.b = u.b - ( ux.a - u.a );
                v.a = ux.a;
            }else {
                v.a = u.a + u.b;
                v.b = 0;
            }

            if( dist[v.a][v.b] > x ) {
                dist[v.a][v.b] = x;
                parent[v.a][v.b] = u;
                q.push( v );
                work[v.a][v.b] = pourBA;
            }

        }
        print( u.a, u.b );
        printf("success\n");

    }


    return 0;
}
```

আরেকটা মজার প্রবলেম

সুপার মারিও একটা মনিটরের ভিতর থাকে। সেখানে অনেকগুলো গ্রাম আর অনেকগুলো দূর্গ। সুপার মারিও পুরো জায়গাটা গুগল ম্যাপ খুলে প্রচুর মাপামাপি করসে, তো সে খুব ভালো করে জানে কোন কোন গ্রামগুলো বা দূর্গগুলোর মধ্যে পথ আছে, আর পথ থাকলে সেই পথের দূরত্ব কত। তো মারিও বহুদিন একা একা পথ মাপামাপি করার পর বোরড হয়ে করলো কি, একটা দূর্গে গিয়ে একটা রাজকণ্যাকে উদ্ধার করলো।কিভাবে কিভাবে সেই দূর্গে সে একটা জাদুর বুট খুঁজে পেলো, যেটা পড়লে সে নিমিষে একটা জায়গা থেকে আরেকটা জায়গায় চলে যেতে পারে। শুধু দুইটা সমস্য। বুটটা বেশ পুরনো, তো কিছুতেই ৫০ কিলোমিটারের বেশি টানা বুট পরে দৌড়ানো যায় না, আর বুটটাকে ৩ বারের বেশি ব্যবহার করা যায় না। এর পরের বার ব্যবহার করলে বুটটা সোজা গর্তে গিয়ে হাজির হবে।

আরেকটা বাজে জিনিস হচ্ছে ধাম করে একটা দেয়ালে বাড়ি না খেলে সে থামতে পারে না, তো দৌড় থামানোর জন্য ওর একটা গ্রামে থামতে হবে অথবা একটা দূর্গে থামতে হবে। আর দৌড় শুরু করার জন্য তার একটা জায়গায় বসে বুট পরতে হবে, পথের মাঝখানে বসে সে বুট পরতে পারবে না, সেজন্য তার দৌড় শুরু করতে হবে কোন একটা গ্রাম থেকে অথবা দূর্গ থেকে।

মারিও এখন কোন এক অজানা(!) কারণে খুব দ্রুত রাজকণ্যাকে নিয়ে বাড়ি ফিরতে চায়! খালি ওর মাথায় একটু বুদ্ধি কম, সেজন্য সে বুঝতে পারছে না কিভাবে কিভাবে গেলে বা কিভাবে কিভাবে বুট পড়লে সবচে' দ্রুত বাড়ি ফেরা যাবে। তো মারিওকে বলতে হবে কিভাবে কিভাবে দৌড়ালে সে সবচে' কম সময়ে বাড়ি ফিরতে পারবে।

টিশিয়া

আমাদের আগের প্রবলেমটার পুরো সিচুয়েশন বর্ণনা করা যেতো (লাল বালতিতে পানি, নীল বালতিতে পানি) এই দুইটা জিনিস দিয়ে। আমরা তারপর পুরো প্রবলেমটাকে এমনভাবে মডেল করসিলাম, যাতে জিনিসটাকে শর্টেস্ট পাথ প্রবলেম বানিয়ে ফেলা যায়।

ঠিক একইভাবে চিন্তা করো, যদি সুপার মারিও অনেক দৌড়াদৌড়ির পর (কিংবা দৌড়াতে দৌড়াতে) তোমাকে ফোন করে বুদ্ধি চায়, সে তোমাকে কি কি বললে তুমি পুরো সিচুয়েশনটা বুঝতে পারবা? অবশ্যই সবার আগে তাকে বলতে হবে সে কোন জায়গায়

আছে। তারপর? তারপর তাকে বলতে হবে সে আর কতবার বুটটা ব্যবহার করতে পারবে। এবং যদি সে বুট পরে দৌড়াতে থাকে, তাহলে বলতে হবে সে কতটুকু দৌড়ালো বুট পড়ে। এই তিনটা বললে আমরা পুরো অবস্থাটা হাড়ে হাড়ে বুঝতে পারি, তাই না?

তো এটা হচ্ছে আমাদের state (কোথায়, বুট কয়বার পরতে পারবে, এই বুটে আর কত মাইল দৌড়ানো যাবে)। এবার এটাকে [a][b][c] হিসেবে চিন্তা করো, আর যদি [a][b][c] থেকে [x][y][z] এ যাওয়া যায়, তাহলে তার মানে হচ্ছে এই দুইটা অবস্থার মধ্যে একটা edge আছে, যেটার cost হবে a থেকে x এর দূরত্ব, যেটা মারিও ইতিমধ্যে জানে। আর বুট পরা থাকলে তো কোন কথাই নাই, মারিওর কোন সময়ই লাগবে না যাইতে। তো আমাদের বর্তমান দূর্গ থেকে বাড়ি তে যাবার সবচে' কম সময়ে যাবার পথ খুঁজে বের করতে হবে।

তো এটা আবারও শর্টেস্ট পাথে ফিরে আসলো ঘুরে ফিরে।
এই প্রবলেমটা হচ্ছে UVa 10269 আর এটা হচ্ছে আমার সলুশন।

```
#include<cstdio>
#include<sstream>
#include<cstdlib>
#include<cctype>
#include<cmath>
#include<algorithm>
#include<set>
#include<queue>
#include<stack>
#include<list>
#include<iostream>
#include<fstream>
#include<numeric>
#include<string>
#include<vector>
#include<cstring>
#include<map>
#include<iterator>
using namespace std;

#define FORIT(i, m) for (__typeof((m).begin()) i=(m).begin(); i!=(m).end(); ++i)
#define REP(i,n) for(int i=0; i<(n); i++)
#define FOR(i,a,b) for(__typeof(b) i=(a); i<=(b); i++)

#define sz size()
#define pb push_back
#define ALL(x) x.begin(), x.end()

#define i64 long long
#define SET(t,v) memset((t), (v), sizeof(t))
#define REV(x) reverse( ALL( x ) )

#define IO freopen("","r",stdin); freopen("","w",stdout);
#define debug(x) cerr << __LINE__ <<" "<< #x " = " << x << endl

typedef vector<int> vi;

int A,B,M,L,K;
int memo[105][505][15];
vi edge[105], cost[105];

int main() {
    int t;
    scanf("%d",&t);

    while( t-- ) {
        scanf("%d %d %d %d %d",&A,&B,&M,&L,&K);
        int x,y,z;

        REP(i,105) { edge[i].clear(); cost[i].clear(); }
        SET( memo, 63 );
        int inf = memo[0][0][0];

        while( M-- ) {
            scanf("%d %d %d",&x,&y,&z);
            edge[x].pb( y ); edge[y].pb( x );
            cost[x].pb( z ); cost[y].pb( z );
        }

        memo[A+B][0][K] = 0;
        queue< int > q;
        q.push( A+B ); q.push( 0 ); q.push( K );

        while( !q.empty() ) {
            int a = q.front(); q.pop();
            int l = q.front(); q.pop();
            int k = q.front(); q.pop();
            int bcost = memo[a][l][k];

            REP(i,edge[a].sz) {
                int d = cost[a][i];
                if( a <= A ) {
                    // opt 1
                    if( l >= d ) {
                        int aa = edge[a][i];
                        int ll = l - d;
                        int kk = k;
                        if( memo[aa][ll][kk] > bcost ) {
                            memo[aa][ll][kk] = bcost;
                            q.push( aa ); q.push( ll ); q.push( kk );
                        }
                    }
```

```
                    // opt 2
                    if( k > 0 && L >= d ) {
                        int aa = edge[a][i];
                        int ll = L - d;
                        int kk = k-1;
                        if( memo[aa][ll][kk] > bcost ) {
                            memo[aa][ll][kk] = bcost;
                            q.push( aa ); q.push( ll ); q.push( kk );
                        }
                    }

                    // opt 3
                    int aa = edge[a][i];
                    int ll = 0;
                    int kk = k;
                    if( memo[aa][ll][kk] > bcost + d ) {
                        memo[aa][ll][kk] = bcost + d;
                        q.push( aa ); q.push( ll ); q.push( kk );
                    }
                }else {
                    // opt 2
                    if( k > 0 && L >= d ) {
                        int aa = edge[a][i];
                        int ll = L - d;
                        int kk = k-1;
                        if( memo[aa][ll][kk] > bcost ) {
                            memo[aa][ll][kk] = bcost;
                            q.push( aa ); q.push( ll ); q.push( kk );
                        }
                    }

                    // opt 3
                    int aa = edge[a][i];
                    int ll = 0;
                    int kk = k;
                    if( memo[aa][ll][kk] > bcost + d ) {
                        memo[aa][ll][kk] = bcost + d;
                        q.push( aa ); q.push( ll ); q.push( kk );
                    }
                }
            }

        }

        int ans = inf;
        REP(l,L+1) REP(k,K+1) ans = min( ans, memo[1][l][k] );
        printf("%d\n", ans );


    }

    return 0;
}
```

শেষ কথা মোটেও শেষ কথা নয়

শর্টেস্ট পাথ কিভাবে মডেল করতে হয়, সেটা নিয়ে আমি আরো দুই দিস্তা গল্প লিখতে পারি। কিন্তু আমার মনে হয় তুমি বুঝতে শুরু করেছো, আসলে কি কারিশমাটা করতে হয় শর্টেস্ট পাথ প্রবলেমগুলোকে নিয়ে। তোমাকে প্রবলেমটাকে একটা গ্রাফে কনভার্ট করতে হবে, আর বুঝতে হবে ঠিক ঠিক কি জানলে তুমি একটা অবস্থা পুরোপুরি বর্ণনা করতে পারো। তুমি যদি বড় হয়ে কখনো সিরিয়াস প্রবলেম সলভার হও, কিংবা রিসার্চার হও, কিংবা সত্যিকারের কোন চরম এক্সাইটিং প্রবলেম নিয়ে কাজ করতে করতে বুড়ো হয়ে যাও, ঘুরে ফিরে তুমি একই কাজ করবা। তোমার প্রথম কাজ হবে গুতাই গুতাই দেখা যে প্রবলেমটাকে একটা known solutionওয়ালা প্রবলেম বানানো যায় কিনা। যদি একেবারেই না যায়, তাহলে তোমার দ্বিতীয় কাজ হবে একটা ঝাকানাকা সলুশন বের করে বিখ্যাত হয়ে যাওয়া। তারপর মানুষ জন এখন যেভাবে Dijkstra লিখে ফাংশন লিখে অনলাইন জাজে সাবমিট করে, সেরকমভাবে তোমার নামে ফাংশন লিখে অনলাইন জাজে সাবমিট করবে। তারপর ২৩ বার WA খেয়ে ফেইসবুকে তোমার নাম নিয়ে স্ট্যাটাস দিবে। কি মজা!

> *Hope, it is the quintessential human delusion, simultaneously the source of your greatest strength, and your greatest weakness.*
> *--- The Architect (Matrix Reloaded)*

খাটা মাথানোর জন্য

BFS

www.spoj.pl/problems/ANARC05I
www.spoj.pl/problems/BYTESE1

www.spoj.pl/problems/CERCO7K
www.spoj.pl/problems/CHMAZE
www.spoj.pl/problems/CLEANRBT
www.spoj.pl/problems/CLOCKS
www.spoj.pl/problems/CURSE
www.spoj.pl/problems/DP
www.spoj.pl/problems/ESCJAILA
www.spoj.pl/problems/HIKE
www.spoj.pl/problems/INUMBER
www.spoj.pl/problems/MAWORK
www.spoj.pl/problems/MLASERP
www.spoj.pl/problems/MTWALK
www.spoj.pl/problems/ONEZERO
www.spoj.pl/problems/PPATH
www.spoj.pl/problems/ROADS
www.spoj.pl/problems/SHOP
www.spoj.pl/problems/TOE1
www.spoj.pl/problems/TOE2

Dijkstra

www.spoj.pl/problems/GONDOR
www.spoj.pl/problems/HIGHWAYS
www.spoj.pl/problems/INCARDS
www.spoj.pl/problems/MELE3
www.spoj.pl/problems/NAJKRACI
www.spoj.pl/problems/SHPATH
www.spoj.pl/problems/TRAFFICN

এগুলো শেষ করে পেট না ভরলে http://uvatoolkit.com/ এখানে গিয়ে BFS বা Dijikstra লিখে একটা সার্চ মারো!

শুভ কামনা!

Comments

Commenting disabled due to a network error. Please reload the page.

You do not have permission to add comments.