

Contents

1 Basic	1	7 Other	18
1.1 Basic codeblock setting	1	7.1 Binary search 三類變化	18
1.2 Basic vim setting	1	7.2 Heap sort	18
1.3 Code Template	1	7.3 Josephus	18
1.4 Python	1	7.4 Largest Multi-interval	18
1.5 Range data	2	7.5 Merge sort	18
1.6 Some Function	2	7.6 N Queen problem	19
1.7 Time	2	7.7 Quick sort	19
2 DP	2	7.8 Sudoku solution	19
2.1 3 維 DP 思路	2	7.9 Weighted Job Scheduling	19
2.2 Knapsack Bounded	2	8 String	20
2.3 Knapsack sample	2	8.1 KMP	20
2.4 Knapsack Unbounded	2	8.2 Min Edit Distance	20
2.5 LCIS	2	8.3 Sliding window	20
2.6 LCS $O(N\log(N))$	3	8.4 Split	20
2.7 LCS	3	9 data structure	20
2.8 LIS $O(N\log(N))$	3	9.1 Bigint	20
2.9 LIS	3	9.2 DisjointSet	22
2.10 LPS	3	9.3 Matirx	22
2.11 Max_subarray	3	9.4 分數	22
2.12 Money problem	4	1 Basic	
2.13 Palindromic Substrings count	4	1.1 Basic codeblock setting	
3 Flow & matching	4	Settings -> Editor -> Keyboard shortcuts -> Plugins -> Source	
3.1 Dinic	4	code formatter (AStyle)	
3.2 Edmonds_karp	4	Settings -> Source Formatter -> Padding	
3.3 Hungarian	5	Delete empty lines within a function or method	
3.4 Independent set	5	Insert space padding around operators	
3.5 Maximum general weighted matching	5	Insert space padding around parentheses on outside	
3.6 Maximum matching	6	Remove extra space padding around parentheses	
3.7 Minimum cut	6	1.2 Basic vim setting	
3.8 Model	6	/*at home directory*/	
3.9 Stable matching	6	/* vi ~/.vimrc */	
4 Geometry	7	syntax enable	
4.1 2D Point	7	set smartindent	
4.2 Circle Intersect	7	set tabstop=4	
4.3 Circumcentre	7	set shiftwidth=4	
4.4 Closest Pair	7	set expandtab	
4.5 Line	7	set relativenumber	
4.6 Max cover rectangle	8	1.3 Code Template	
4.7 Polygon	9	#include <bits/stdc++.h>	
4.8 Smallest Circle	10	using namespace std;	
4.9 Triangle	10	typedef long long ll;	
5 Graph	10	typedef unsigned long long ull;	
5.1 BCC edge	10	#define pb push_back	
5.2 BCC vertex	11	#define len(k) (int)k.length()	
5.3 Bellman Ford	11	#define all(p) p.begin(), p.end()	
5.4 BFS queue	11	#define endl '\n'	
5.5 DFS rec	11	#define x first	
5.6 Dijkstra	12	#define y second	
5.7 Euler circuit	12	#define bug(k) cout << "value of " << #k << " is " << k << endl	
5.8 Floyd warshall	12	;	
5.9 Hamilton cycle	12	#define bugp(k) cout << "pair of " << #k << " is " << k.x << "	
5.10 Kruskal	13	' << k.y << endl;	
5.11 LCA	13	#define bugarr(k) for(auto i : k) cout << i << ' '; cout <<	
5.12 Minimum Weight Cycle	13	endl;	
5.13 Prim	14	int main()	
5.14 Union find	14	{	
6 Mathematics	14	ios::sync_with_stdio(0);	
6.1 Catalan	14	cin.tie(0);	
6.2 Combination	14	return 0;	
6.3 CRT	14	}	
6.4 Extended Euclidean	14	1.4 Python	
6.5 Fermat	15	//輸入	
6.6 Hex to Dec	15	input()	
6.7 Log	15		
6.8 Mod 性質	15	array = [0] * (N) //N個0	
6.9 PI	15	range(0, N) // 0 ~ N-1	
6.10 Pow_Mod	15	line = input().split()	
6.11 Prime table	15	D, R, N = map(int, line) // 分三個 int 變數	
6.12 Prime 判斷	16		
6.13 Round(小數)	16	// 才是 取除數	
6.14 二分逼近法	16	/ 是 小數運算	
6.15 公式	16		
6.16 四則運算	16	pow(a, b, c) // a ^ b % c	
6.17 因數表	16		
6.18 數字乘法組合	16	print(*objects, sep = ' ', end = '\n')	
6.19 數字加法組合	17	// objects -- 可以一次輸出多個對象	
6.20 羅馬數字	17	// sep -- 分開多個objects	
6.21 質因數分解	17	// end -- 默認值是\n	
6.22 質數數量	17		

```
// EOF break
try:
    while True:
        //input something
except EOFError:
    pass
```

1.5 Range data

```
int (-2147483648 to 2147483647)
unsigned int(0 to 4294967295)
long(-2147483648 to 2147483647)
unsigned long(0 to 4294967295)
long long(-9223372036854775808 to 9223372036854775807)
unsigned long long (0 to 18446744073709551615)
```

1.6 Some Function

```
round(double f); // 四捨五入
ceil(double f); // 進入
floor(double f); // 捨去
__builtin_popcount(int n); // 32bit有多少 1
to_string(int s); // int to string

cout << setprecision(位數) // cout 小數位設定
printf 型別
"%lf" // long double
"%lld" // long long int

set_union(all(a), all(b), back_inserter(d)); // 聯集
set_intersection(all(a), all(b), back_inserter(c)); // 交集

/** 全排列要先 sort !!! */
next_permutation(num.begin(), num.end());
prev_permutation(num.begin(), num.end());
//用binary search找第一個大於或等於val的位置
vector<int>::iterator it = lower_bound(v.begin(), v.end(), val)
;
//用binary search找第一個大於val的位置
vector<int>::iterator it = upper_bound(v.begin(), v.end(), val)
;

/*找到範圍裏面的最大元素*/
max_element(n, n + len); // n到n+len範圍內最大值
max_element(v.begin(), v.end()); // vector 中最大值
/*找到範圍裏面的最大元素*/
min_element(n, n + len); // n到n+len範圍內最小值
min_element(v.begin(), v.end()); // vector 中最小值

/*queue*/
queue<datatype> q;
front(); /*取出最前面的值(沒有移除掉)*/
back(); /*取出最後面的值(沒有移除掉)*/
pop(); /*移掉最前面的值*/
push(); /*新增值到最後面*/
empty(); /*回傳bool,檢查是不是空的queue*/
size(); /*queue 的大小*/

/*stack*/
stack<datatype> s;
top(); /*取出最上面的值(沒有移除掉)*/
pop(); /*移掉最上面的值*/
push(); /*新增值到最上面*/
empty(); /*bool 檢查是不是空*/
size(); /*stack 的大小*/

/*unordered_set*/
unordered_set<datatype> s;
unordered_set<datatype> s(arr, arr + n);
/*initial with array*/
insert(); /*插入值*/
erase(); /*刪除值*/
empty(); /*bool 檢查是不是空*/
count(); /*判斷元素存在回傳1 無則回傳0*/

/*tuple*/
tuple<datatype,datatype,datatype> t;
std::get<0>(t) /*Get first element of tuple*/
std::get<1>(t) /*Get second element of tuple*/
std::get<2>(t) /*Get third element of tuple*/
```

1.7 Time

```
cout << 1.0 * clock() / CLOCKS_PER_SEC << endl;
```

2 DP

2.1 3 維 DP 思路

解題思路: $dp[i][j][k]$
 i 跟 j 代表 $range\ i \sim j$ 的 value
 k 在我的理解裡是視题目的要求而定的
像是 Remove Boxes 當中 k 代表的是在 i 之前還有多少個連續的箱子
所以每次區間消去的值就是 $(k+1) * (k+1)$
換言之,我認為可以理解成 k 的意義就是題目今天所關注的重點,就是老師說的題目所規定的運算

2.2 Knapsack Bounded

```
const int N = 100, W = 100000;
int cost[N], weight[N], number[N];
int c[W + 1];
void knapsack(int n, int w)
{
    for (int i = 0; i < n; ++i)
    {
        int num = min(number[i], w / weight[i]);
        for (int k = 1; num > 0; k -= 1)
        {
            if (k > num)
                k = num;
            num -= k;
            for (int j = w; j >= weight[i] * k; --j)
                c[j] = max(c[j], c[j - weight[i] * k] + cost[i] * k);
        }
    }
    cout << "Max Prince" << c[w];
}
```

2.3 Knapsack sample

```
int Knapsack(vector<int> weight, vector<int> value, int
bag_Weight)
{
    // vector<int> weight = {1, 3, 4};
    // vector<int> value = {15, 20, 30};
    // int bagWeight = 4;
    vector<vector<int>> dp(weight.size(), vector<int>(bagWeight
+ 1, 0));
    dp[0][0] = value[0];
    // weight數組的大小就是物品個數
    for (int i = 1; i < weight.size(); i++)
    { // 遍歷物品
        for (int j = 0; j <= bagWeight; j++)
        { // 遍歷背包容量
            if (j < weight[i]) dp[i][j] = dp[i - 1][j];
            else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j -
weight[i]] + value[i]);
        }
    }
    cout << dp[weight.size() - 1][bagWeight] << endl;
}
```

2.4 Knapsack Unbounded

```
const int N = 100, W = 100000;
int cost[N], weight[N];
int c[W + 1];
void knapsack(int n, int w)
{
    memset(c, 0, sizeof(c));
    for (int i = 0; i < n; ++i)
        for (int j = weight[i]; j <= w; ++j)
            c[j] = max(c[j], c[j - weight[i]] + cost[i]);
    cout << "最高的價值為" << c[w];
}
```

2.5 LCIS

```
int LCIS_len(vector<int> arr1, vector<int> arr2)
{
    int n = arr1.size(), m = arr2.size();
    vector<int> table(m, 0);
    for (int j = 0; j < m; j++)
        table[j] = 0;
    for (int i = 0; i < n; i++)
    {
        int current = 0;
        for (int j = 0; j < m; j++)
        {
            if (arr1[i] == arr2[j])
                if (current + 1 > table[j])
                    table[j] = current + 1;

            if (arr1[i] > arr2[j])
                if (table[j] > current)
                    current = table[j];
        }
    }
    int result = 0;
```

```

    for (int i = 0; i < m; i++)
        if (table[i] > result)
            result = table[i];
    return result;
}

```

2.6 LCS $O(N \log(N))$

```

#define LEN 500004
int a[LEN], b[LEN];
int loc[LEN], n;
int LIS()
{
    for (int i = 1; i <= n; i++)
        loc[b[i]] = i;
    for (int i = 1; i <= n; i++)
        b[i] = loc[a[i]];
    int k, l, r, mid;
    a[1] = b[1], k = 1;
    for (int i = 2; i <= n; i++)
    {
        if (a[k] < b[i])
            a[++k] = b[i];
        else
        {
            l = 1;
            r = k;
            while (l <= r)
            {
                mid = (l + r) / 2;
                if (a[mid] < b[i])
                    l = mid + 1;
                else
                    r = mid - 1;
            }
            a[l] = b[i];
        }
    }
    return k;
}

```

2.7 LCS

```

int LCS(vector<string> Ans, vector<string> num)
{
    int N = Ans.size(), M = num.size();
    vector<vector<int>> LCS(N + 1, vector<int>(M + 1, 0));
    for (int i = 1; i <= N; ++i)
    {
        for (int j = 1; j <= M; ++j)
        {
            if (Ans[i - 1] == num[j - 1])
                LCS[i][j] = LCS[i - 1][j - 1] + 1;
            else
                LCS[i][j] = max(LCS[i - 1][j], LCS[i][j - 1]);
        }
    }
    cout << LCS[N][M] << '\n';
    // 列印 LCS
    int n = N, m = M;
    vector<string> k;
    while (n && m)
    {
        if (LCS[n][m] != max(LCS[n - 1][m], LCS[n][m - 1]))
        {
            k.push_back(Ans[n - 1]);
            n--;
            m--;
        }
        else if (LCS[n][m] == LCS[n - 1][m])
            n--;
        else if (LCS[n][m] == LCS[n][m - 1])
            m--;
    }
    reverse(k.begin(), k.end());
    for (auto i : k)
        cout << i << " ";
    cout << endl;
    return LCS[N][M];
}

```

2.8 LIS $O(N \log(N))$

```

int LIS(vector<int> &v) //  $O(n \log(n))$ 
{
    // 需要求 LDS 請把 array reverse 反過來求 LIS
    // 但必須注意 lower_bound or upper_bound
    if (v.size() == 0)
        return 0;
    vector<int> dp(v.size(), 0);
    int length = 1;
    dp[0] = v[0];
    for (int i = 1; i < v.size(); i++)
    {
        auto b = dp.begin(), e = dp.begin() + length;
        // auto it = lower_bound(b, e, v[i]); // 後面 >= 前面

```

```

        auto it = upper_bound(b, e, v[i]); // 後面 > 前面
        if (it == dp.begin() + length)
            dp[length++] = v[i];
        else
            *it = v[i];
    }
    return length;
}

```

2.9 LIS

```

vector<int> ans;
void LIS(vector<int> &arr)
{
    vector<int> dp(arr.size(), 1);
    vector<int> pos(arr.size(), -1);
    int res = INT_MIN, index = 0;
    for (int i = 0; i < arr.size(); ++i)
    {
        for (int j = i + 1; j < arr.size(); ++j)
        {
            if (arr[j] > arr[i])
            {
                if (dp[i] + 1 > dp[j])
                {
                    dp[j] = dp[i] + 1;
                    pos[j] = i;
                }
            }
        }
        if (dp[i] > res)
        {
            res = dp[i];
            index = i;
        }
    }
    cout << res << endl; // length
    printLIS(arr, pos, index);
    for (int i = 0; i < ans.size(); i++)
    {
        cout << ans[i];
        if (i != ans.size() - 1)
            cout << ' ';
    }
    cout << '\n';
}
void printLIS(vector<int> &arr, vector<int> &pos, int index)
{
    if (pos[index] != -1)
        printLIS(arr, pos, pos[index]);
    ans.push_back(arr[index]);
}

```

2.10 LPS

```

// manacher
void LPS(string s)
{
    int maxlen = 0, l, r;
    int n = s.size();
    for (int i = 0; i < n; i++)
    {
        int x = 0;
        while ((s[i - x] == s[i + x]) && (i - x >= 0) && (i + x < n))
            //odd length
            x++;
        x--;
        if (2 * x + 1 > maxlen)
        {
            maxlen = 2 * x + 1;
            l = i - x;
            r = i + x;
        }
        x = 0;
        while ((s[i - x] == s[i + 1 + x]) && (i - x >= 0) && (i + 1 + x < n))
            //even length
            x++;
        if (2 * x > maxlen)
        {
            maxlen = 2 * x;
            l = i - x + 1;
            r = i + x;
        }
    }
    cout << maxlen << '\n';
    cout << l + 1 << ' ' << r + 1 << '\n'; // 最後長度 頭到尾
}

```

2.11 Max_subarray

```

/*Kadane's algorithm*/
int maxSubArray(vector<int> &nums) {
    int local_max = nums[0], global_max = nums[0];
    for (int i = 1; i < nums.size(); i++) {

```

```

        local_max = max(nums[i], nums[i] + local_max);
        global_max = max(local_max, global_max);
    }
    return global_max;
}

```

2.12 Money problem

```

// 能否湊得某個價位
void change(vector<int> price, int limit)
{
    vector<bool> c(limit + 1, 0);
    c[0] = true;
    for (int i = 0; i < price.size(); ++i) // 依序加入各種面額
        for (int j = price[i]; j <= limit; ++j) // 由低價位逐步到高價位
            c[j] = c[j] | c[j - price[i]]; // 湊、湊、湊
    if (c[limit]) cout << "YES\n";
    else cout << "NO\n";
}

// 湊得某個價位的湊法總共幾種
void change(vector<int> price, int limit)
{
    vector<int> c(limit + 1, 0);
    c[0] = true;
    for (int i = 0; i < price.size(); ++i)
        for (int j = price[i]; j <= limit; ++j)
            c[j] += c[j - price[i]];
    cout << c[limit] << '\n';
}

// 湊得某個價位的最少錢幣用量
void change(vector<int> price, int limit)
{
    vector<int> c(limit + 1, 0);
    c[0] = true;
    for (int i = 0; i < price.size(); ++i)
        for (int j = price[i]; j <= limit; ++j)
            c[j] = min(c[j], c[j - price[i]] + 1);
    cout << c[limit] << '\n';
}

// 湊得某個價位的錢幣用量，有哪幾種可能性
void change(vector<int> price, int limit)
{
    vector<int> c(limit + 1, 0);
    c[0] = true;
    for (int i = 0; i < price.size(); ++i)
        for (int j = price[i]; j <= limit; ++j)
            c[j] |= c[j - price[i]] << 1; // 錢幣數量加一，每一種可能性都加一。

    for (int i = 1; i <= 63; ++i)
        if (c[i] & (1 << i))
            cout << "用" << i << "個錢幣可湊得價位" << i;
}

```

2.13 Palindromic Substrings count

```

// 計算 回文字串數量
int countSubstrings(string s)
{
    int n = s.size(), res = 0;
    vector<vector<bool>> dp(n, vector<bool>(n));
    for (int i = n - 1; i >= 0; --i)
    {
        for (int j = i; j < n; ++j)
        {
            dp[i][j] = (s[i] == s[j]) && (j - i <= 2 || dp[i + 1][j - 1]);
            if (dp[i][j]) ++res;
        }
    }
    return res;
}

```

3 Flow & matching

3.1 Dinic

```

const long long INF = 1LL << 60;
struct Dinic { // O(VVE), with minimum cut
    static const int MAXN = 5003;
    struct Edge {
        int u, v;
        long long cap, rest;
    };
    int n, m, s, t, d[MAXN], cur[MAXN];
    vector<Edge> edges;
    vector<int> G[MAXN];
    void init() {
        edges.clear();
    }
}

```

```

    for (int i = 0; i < n; ++i) G[i].clear();
    n = 0;

    // min cut start
    bool side[MAXN];
    void cut(int u) {
        side[u] = 1;
        for (int i : G[u]) {
            if (!side[edges[i].v] && edges[i].rest)
                cut(edges[i].v);
        }
    }

    // min cut end
    int add_node() {
        return n++;
    }

    void add_edge(int u, int v, long long cap) {
        edges.push_back({u, v, cap, cap});
        edges.push_back({v, u, 0, 0});
        m = edges.size();
        G[u].push_back(m - 2);
        G[v].push_back(m - 1);
    }

    bool bfs() {
        fill(d, d + n, -1);
        queue<int> que;
        que.push(s); d[s] = 0;
        while (!que.empty()) {
            int u = que.front(); que.pop();
            for (int ei : G[u]) {
                Edge &e = edges[ei];
                if (d[e.v] < 0 && e.rest > 0) {
                    d[e.v] = d[u] + 1;
                    que.push(e.v);
                }
            }
        }
        return d[t] >= 0;
    }

    long long dfs(int u, long long a) {
        if (u == t || a == 0) return a;
        long long flow = 0, f;
        for (int &i = cur[u]; i < (int)G[u].size(); ++i) {
            Edge &e = edges[G[u][i]];
            if (d[u] + 1 != d[e.v]) continue;
            f = dfs(e.v, min(a, e.rest));
            if (f > 0) {
                e.rest -= f;
                edges[G[u][i]^1].rest += f;
                flow += f;
                a -= f;
                if (a == 0) break;
            }
        }
        return flow;
    }

    long long maxflow(int _s, int _t) {
        s = _s, t = _t;
        long long flow = 0, mf;
        while (bfs()) {
            fill(cur, cur + n, 0);
            while ((mf = dfs(s, INF)) > 0) flow += mf;
        }
        return flow;
    }
} dinic;

```

3.2 Edmonds_karp

```

/* Flow - Edmonds-karp */
/* Based on UVa820 */
#define inf 1000000
int getMaxFlow(vector<vector<int>> &capacity, int s, int t, int n) {
    int ans = 0;
    vector<vector<int>> residual(n + 1, vector<int>(n + 1, 0)); // residual network
    while (true) {
        vector<int> bottleneck(n + 1, 0);
        bottleneck[s] = inf;
        queue<int> q;
        q.push(s);
        vector<int> pre(n + 1, 0);
        while (!q.empty() && bottleneck[t] == 0) {
            int cur = q.front();
            q.pop();
            for (int i = 1; i <= n; ++i) {
                if (bottleneck[i] == 0 && capacity[cur][i] > residual[cur][i]) {
                    cur = i;
                    q.push(i);
                    pre[i] = cur;
                    bottleneck[i] = min(bottleneck[cur], capacity[cur][i] - residual[cur][i]);
                }
            }
        }
        if (bottleneck[t] == 0) break;
    }
}

```

```

    for(int cur = t; cur != s; cur = pre[cur]){
        residual[pre[cur]][cur] += bottleneck[t];
        residual[cur][pre[cur]] -= bottleneck[t];
    }
    ans += bottleneck[t];
}
return ans;
}
int main(){
    int testcase = 1;
    int n;
    while(cin>>n){
        if(n == 0)
            break;
        vector<vector<int>> capacity(n+1, vector<int>(n+1, 0));
        int s, t, c;
        cin >> s >> t >> c;
        int a, b, bandwidth;
        for(int i = 0 ; i < c ; ++i){
            cin >> a >> b >> bandwidth;
            capacity[a][b] += bandwidth;
            capacity[b][a] += bandwidth;
        }
        cout << "Network " << testcase++ << endl;
        cout << "The bandwidth is " << getMaxFlow(capacity, s, t, n)
            << "." << endl;
        cout << endl;
    }
    return 0;
}

```

3.3 Hungarian

```

/*bipartite - hungarian*/
/*Based on 2017 ICPC Taiwan regional final Problem I*/
bool dfs(vector<vector<bool>> mp, vector<bool> pass, vector<int>
    >& pre, int cur){
    for(int i = 0; i < mp[cur].size(); i++){
        if(mp[cur][i] && !pass[i]){
            pass[i] = true;
            if(pre[i] == -1 || dfs(mp, pass, pre, pre[i])){
                pre[i] = cur;
                return true;
            }
        }
    }
    return false;
}
int hungarian(vector<vector<bool>> mp, int n, int m){
    int ans = 0;
    vector<int> pre(m, -1);
    for(int i = 0; i < n; i++){
        vector<bool> pass(m, false);
        if(dfs(mp, pass, pre, i))
            ans += 1;
    }
    return ans;
}
int main(){
    int m, n, e;
    while(cin>>n){
        if(n == 0) break;
        cin>>m>>e;
        int a, b;
        vector<vector<bool>> mp(n, vector<bool>(m, false));
        for(int i = 0; i < e; i++){
            cin>>a>>b;
            mp[a][b] = true;
        }
        cout<<hungarian(mp,n,m)<<endl;
    }
    return 0;
}

```

3.4 Independent set

```

int mp[30][30];
int vis[30];
int n, m;
int dfs(int now){
    for(int i = 0; i < now; i++){
        if(mp[now][i] && (vis[now] == vis[i])) //與now相鄰的結點與
            now有相同的頻率
            return 0;
    }
    if(now == n - 1){ //遍歷結束
        return 1;
    }
    for(int i = 1; i <= 3; i++){ //選顏色
        vis[now+1] = i;
        if(dfs(now+1))
            return 1;
    }
    return 0;
}

```

```

}
int main(){
    int t;
    cin >> t;
    while(t--){
        memset(vis, 0, sizeof(vis));
        memset(mp, 0, sizeof(mp));
        cin >> n >> m;
        while(m--){
            int a, b;
            cin >> a >> b;
            mp[a][b] = 1;
            mp[b][a] = 1;
        }
        vis[0] = 1; //第一個節點任意選一種顏色
        if(dfs(0))
            cout << "Y" << endl;
        else
            cout << "N" << endl;
        //print vis = print result of combinaton
    }
    return 0;
}

```

3.5 Maximum general weighted matching

```

// From NCTU codebook
// Minimum Weight Perfect Matching (Perfect Match)
struct Graph {
    static const int MAXN = 105;
    int n, e[MAXN][MAXN];
    int match[MAXN], d[MAXN], onstk[MAXN];
    vector<int> stk;
    void init(int _n) {
        n = _n;
        for( int i = 0 ; i < n ; i ++ )
            for( int j = 0 ; j < n ; j ++ )
                e[i][j] = 0;
    }
    void add_edge(int u, int v, int w) {
        e[u][v] = e[v][u] = w;
    }
    bool SPFA(int u){
        if (onstk[u]) return true;
        stk.push_back(u);
        onstk[u] = 1;
        for ( int v = 0 ; v < n ; v ++ ) {
            if (u != v && match[u] != v && !onstk[v]) {
                int m = match[v];
                if ( d[m] > d[u] - e[v][m] + e[u][v] ){
                    d[m] = d[u] - e[v][m] + e[u][v];
                    onstk[v] = 1;
                    stk.push_back(v);
                    if (SPFA(m)) return true;
                    stk.pop_back();
                    onstk[v] = 0;
                }
            }
        }
        onstk[u] = 0;
        stk.pop_back();
        return false;
    }
    int solve() {
        for ( int i = 0 ; i < n ; i += 2 ) {
            match[i] = i+1;
            match[i+1] = i;
        }
        while (true){
            int found = 0;
            for ( int i = 0 ; i < n ; i ++ )
                onstk[i] = d[i] = 0;
            for ( int i = 0 ; i < n ; i ++ ) {
                stk.clear();
                if ( !onstk[i] && SPFA(i) ) {
                    found = 1;
                    while ( stk.size() >= 2 ) {
                        int u = stk.back(); stk.
                            pop_back();
                        int v = stk.back(); stk.
                            pop_back();
                        match[u] = v;
                        match[v] = u;
                    }
                }
            }
            if (!found) break;
        }
        int ret = 0;
        for ( int i = 0 ; i < n ; i ++ )
            ret += e[i][match[i]];
        ret /= 2;
        return ret;
    }
} graph;

```

3.6 Maximum matching

```

/*bipartite - maximum matching*/
bool dfs(vector<vector<bool>> res,int node,vector<int>& x,
vector<int>& y, vector<bool> pass){
    for (int i = 0; i < res[0].size(); i++){
        if(res[node][i] && !pass[i]){
            pass[i] = true;
            if(y[i] == -1 || dfs(res,y[i],x,y,pass)){
                x[node] = i;
                y[i] = node;
                return true;
            }
        }
    }
    return false;
}

int main(){
    int n,m,l;
    while(cin>>n>>m>>l){
        vector<vector<bool>> res(n, vector<bool>(m, false));
        for (int i = 0; i < l; i++){
            int a, b;
            cin >> a >> b;
            res[a][b] = true;
        }
        int ans = 0;
        vector<int> x(n, -1);
        vector<int> y(n, -1);
        for (int i = 0; i < n; i++){
            vector<bool> pass(n, false);
            if(dfs(res,i,x,y,pass))
                ans += 1;
        }
        cout << ans << endl;
    }
    return 0;
}

/*
input:
4 3 5 //n matching m, l links
0 0
0 2
1 0
2 1
3 1
answer is 3
*/

```

3.7 Minimum cut

```

/*from 演算法筆記*/
typedef int Graph[9][9]; // adjacency matrix
Graph C, F, R; // 分別是容量上限、流量、剩餘容量
bool visit[9];
void DFS(int i)
{
    visit[i] = true;
    for (int j=0; j<9; ++j)
        if (!visit[j] && F[i][j] < C[i][j])
            DFS(j);
}

void minimum_s_t_cut(int s, int t)
{
    // 求一個最大源匯流，源點為s點，匯點為t點。
    Edmonds_Karp(s, t);

    // 從源點開始遍歷，找出流量瓶頸。
    memset(visit, false, sizeof(visit));
    DFS(s);

    // 找出其中一個最小源匯割，會是源點側點數最少的最小源匯割。
    for (int i=0; i<9; ++i) // 窮舉源點側的點
        if (visit[i])
            for (int j=0; j<9; ++j) // 窮舉匯點側的點
                if (!visit[j])
                    if (C[i][j] > 0) // 要確定有邊
                        cout << "割上的邊有"
                            << "由" << i << "到" << j;
}

```

3.8 Model

- Theorem
 - 最大匹配 + 最小邊覆蓋 = V
 - 最大獨立集 + 最小點覆蓋 = V
 - 最大匹配 = 最小點覆蓋
 - 最小路徑覆蓋數 = V - 最大匹配數
 - maximum flow minimum cut
 - 找到一個最大流 = 至少產生一個最小割
 - 找最大流的bottleneck就可以找到最小割
- Maximum Independent Set
 - General: [NPC] maximum clique of complement of G
 - Bipartite Graph: [P] Maximum Cardinality Bipartite Matching
 - Tree: [P] dp
- Minimum Dominating Set
 - General: [NPC] backtracking
 - Bipartite Graph: [NPC] backtracking
 - Tree: [P] DP
- Minimum Vertex Cover
 - General: [NPC] (?)maximum clique of complement of G
 - Bipartite Graph: [P] Maximum Cardinality Bipartite Matching
 - Tree: [P] Greedy, from leaf to root
- Minimum Edge Cover
 - General: [P] V - Maximum Matching
 - Bipartite Graph: [P] Greedy, strategy: cover small degree node first.
 - (Min/Max)Weighted: [P]: Minimum/Maximum Weight Matching

3.9 Stable matching

```

/*based on UVa1175*/
/*stable marriage problem*/
void engage(vector<int>& bm, vector<int>& gm, int a, int b,
queue<int>& q){
    int tmp = gm[b];
    if(tmp != -1){
        bm[tmp] = -1;
        q.push(tmp);
    }
    bm[a] = b;
    gm[b] = a;
}

int main(){
    int cases;
    bool blank = false;
    cin>>cases;
    while(cases--){
        if(blank) cout<<endl;
        int n,a;
        cin>>n;
        queue<int> q; // proposal
        vector<vector<int>> boy(n+1,vector<int>(n+1,0)),girl(n+1,vector<int>(n+1,0));
        vector<int> p(n+1,1);
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= n; j++){
                cin>>a;
                boy[i][j] = a;
            }
            q.push(i);
        }
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= n; j++){
                cin>>a;
                girl[i][a] = j;
            }
        }
        vector<int> bm(n+1,-1),gm(n+1,-1);
        while(!q.empty()){
            int cur = q.front();
            q.pop();

```



```

    int car = boy[cur][p[cur]++];
    if(gm[car] == -1 || girl[car][cur] < girl[car][gm[car]]) engage(bm, gm, cur, car, q);
    else q.push(cur);
}
/*This is optimal solution for lady*/
/*If want optimal for man change the lady and man on above algorithm*/
for(int i = 1; i <= n; i++){
    cout<<bm[i]<<endl;
}
blank = true;
return 0;
}
}

```

4 Geometry

4.1 2D Point

```

const double PI = atan2(0.0, -1.0);
template <typename T>
struct point
{
    T x, y;
    point() {}
    point(const T &x, const T &y) : x(x), y(y) {}
    point operator+(const point &b) const
    {
        return point(x + b.x, y + b.y);
    }
    point operator-(const point &b) const
    {
        return point(x - b.x, y - b.y);
    }
    point operator*(const T &b) const
    {
        return point(x * b, y * b);
    }
    point operator/(const T &b) const
    {
        return point(x / b, y / b);
    }
    bool operator==(const point &b) const
    {
        return x == b.x && y == b.y;
    }
    T dot(const point &b) const
    {
        return x * b.x + y * b.y;
    }
    T cross(const point &b) const
    {
        return x * b.y - y * b.x;
    }
    point normal() const
    {
        //求法向量
        return point(-y, x);
    }
    T abs() const
    {
        //向量長度的平方
        return sqrt(abs2(p));
    }
    T abs2() const
    {
        //向量長度的平方
        return dot(*this);
    }
    T rad(const point &b) const
    {
        //兩向量的弧度
        return fabs(atan2(fabs(cross(b)), dot(b)));
    }
    T getA() const
    {
        //對x軸的弧度
        T A = atan2(y, x); //超過180度會變負的
        if (A <= -PI / 2)
            A += PI * 2;
        return A;
    }
};

```

4.2 Circle Intersect

```

bool same(double a, double b)
{
    return abs(a - b) < 0;
}
struct P
{
    double x, y;
    P() : x(0), y(0) {}
    P(double x, double y) : x(x), y(y) {}
    P operator+(P b) { return P(x + b.x, y + b.y); }
    P operator-(P b) { return P(x - b.x, y - b.y); }
}

```

```

P operator*(double b) { return P(x * b, y * b); }
P operator/(double b) { return P(x / b, y / b); }
double operator*(P b) { return x * b.x + y * b.y; }
// double operator^(P b) { return x * b.y - y * b.x; }
double abs() { return hypot(x, y); }
P unit() { return *this / abs(); }
P rot(double o)
{
    double c = cos(o), s = sin(o);
    return P(c * x - s * y, s * x + c * y);
}
double angle() { return atan2(y, x); }
};
struct C
{
    P c;
    double r;
    C(P c = P(0, 0), double r = 0) : c(c), r(r) {}
};
vector<P> Intersect(C a, C b)
{
    if (a.r > b.r)
        swap(a, b);
    double d = (a.c - b.c).abs();
    vector<P> p;
    if (same(a.r + b.r, d))
        p.pb(a.c + (b.c - a.c).unit() * a.r);
    else if (a.r + b.r > d && d + a.r >= b.r)
    {
        double o = acos((sqrt(a.r) + sqrt(d) - sqrt(b.r)) / (2 * a.r * d));
        P i = (b.c - a.c).unit();
        p.pb(a.c + i.rot(o) * a.r);
        p.pb(a.c + i.rot(-o) * a.r);
    }
    return p;
}

```

4.3 Circumcentre

```

Point circumcentre(Point &p0, Point &p1, Point &p2)
{
    Point a = p1 - p0;
    Point b = p2 - p0;
    Double c1 = abs2(a) * 0.5;
    Double c2 = abs2(b) * 0.5;
    Double d = a % b;
    Double x = p0.x + (c1 * b.y - c2 * a.y) / d;
    Double y = p0.y + (c2 * a.x - c1 * b.x) / d;
    return {x, y};
}

```

4.4 Closest Pair

```

//最近點對 (距離) //台大
vector<pair<double, double>> p;
double closest_pair(int l, int r)
{
    // p 要對 x 軸做 sort
    if (l == r)
        return 1e9;
    if (r - l == 1)
        return dist(p[l], p[r]); // 兩點距離
    int m = (l + r) >> 1;
    double d = min(closest_pair(l, m), closest_pair(m + 1, r));
    vector<int> vec;
    for (int i = m; i >= l && fabs(p[m].x - p[i].x) < d; --i)
        vec.push_back(i);
    for (int i = m + 1; i <= r && fabs(p[m].x - p[i].x) < d; ++i)
        vec.push_back(i);
    sort(vec.begin(), vec.end(), [&](int a, int b)
        { return p[a].y < p[b].y; });
    for (int i = 0; i < vec.size(); ++i)
        for (int j = i + 1; j < vec.size() && fabs(p[vec[j]].y - p[vec[i]].y) < d; ++j)
            d = min(d, dist(p[vec[i]], p[vec[j]]));
    return d;
}

```

4.5 Line

```

template <typename T>
struct line
{
    line() {}
    point<T> p1, p2;
    T a, b, c; //ax+by+c=0
    line(const point<T> &x, const point<T> &y) : p1(x), p2(y) {}
    void pton()
    {
        // 轉成一般式
        a = p1.y - p2.y;
        b = p2.x - p1.x;
        c = -a * p1.x - b * p1.y;
    }
}

```

```

}
T ori(const point<T> &p) const
{ //點和有向直線的關係 · >0左邊 · =0在線上 <0右邊
    return (p2 - p1).cross(p - p1);
}
T btw(const point<T> &p) const
{ //點投影落在線段上 <=0
    return (p1 - p).dot(p2 - p);
}
bool point_on_segment(const point<T> &p) const
{ //點是否在線段上
    return ori(p) == 0 && btw(p) <= 0;
}
T dis2(const point<T> &p, bool is_segment = 0) const
{ //點跟直線/線段的距離平方
    point<T> v = p2 - p1, v1 = p - p1;
    if (is_segment)
    {
        point<T> v2 = p - p2;
        if (v.dot(v1) <= 0)
            return v1.abs2();
        if (v.dot(v2) >= 0)
            return v2.abs2();
    }
    T tmp = v.cross(v1);
    return tmp * tmp / v.abs2();
}
T seg_dis2(const line<T> &l) const
{ //兩線段距離平方
    return min({dis2(l.p1, 1), dis2(l.p2, 1), l.dis2(p1, 1), l.dis2(p2, 1)});
}
point<T> projection(const point<T> &p) const
{ //點對直線的投影
    point<T> n = (p2 - p1).normal();
    return p - n * (p - p1).dot(n) / n.abs2();
}
point<T> mirror(const point<T> &p) const
{
    //點對直線的鏡射 · 要先呼叫pton轉成一般式
    point<T> R;
    T d = a * a + b * b;
    R.x = (b * b * p.x - a * a * p.x - 2 * a * b * p.y - 2 * a * c) / d;
    R.y = (a * a * p.y - b * b * p.y - 2 * a * b * p.x - 2 * b * c) / d;
    return R;
}
bool equal(const line &l) const
{ //直線相等
    return ori(l.p1) == 0 && ori(l.p2) == 0;
}
bool parallel(const line &l) const
{
    return (p1 - p2).cross(l.p1 - l.p2) == 0;
}
bool cross_seg(const line &l) const
{
    return (p2 - p1).cross(l.p1 - p1) * (p2 - p1).cross(l.p2 - p1) <= 0; //直線是否交線段
}
int line_intersect(const line &l) const
{ //直線相交情況 · -1無限多點 · 1交於一點 · 0不相交
    return parallel(l) ? (ori(l.p1) == 0 ? -1 : 0) : 1;
}
int seg_intersect(const line &l) const
{
    T c1 = ori(l.p1), c2 = ori(l.p2);
    T c3 = l.ori(p1), c4 = l.ori(p2);
    if (c1 == 0 && c2 == 0)
    { //共線
        bool b1 = btw(l.p1) >= 0, b2 = btw(l.p2) >= 0;
        T a3 = l.btw(p1), a4 = l.btw(p2);
        if (b1 && b2 && a3 == 0 && a4 >= 0)
            return 2;
        if (b1 && b2 && a3 >= 0 && a4 == 0)
            return 3;
        if (b1 && b2 && a3 >= 0 && a4 >= 0)
            return 0;
        return -1; //無限交點
    }
    else if (c1 * c2 <= 0 && c3 * c4 <= 0)
        return 1;
    return 0; //不相交
}
point<T> line_intersection(const line &l) const
{ /*直線交點*/
    point<T> a = p2 - p1, b = l.p2 - l.p1, s = l.p1 - p1;
    // if (a.cross(b) == 0)
    //     return INF;
    // return p1 + a * (s.cross(b) / a.cross(b));
}
point<T> seg_intersection(const line &l) const
{ //線段交點

```

```

int res = seg_intersect(l);
if (res <= 0)
    assert(0);
if (res == 2)
    return p1;
if (res == 3)
    return p2;
return line_intersection(l);
}
};

```

4.6 Max cover rectangle

```

const double PI = atan2(0.0, -1.0);
const double eps = 1e-10;
typedef point<double> p; // data type 依照題目更改
int mycmp(double a) { return fabs(a) < eps ? 0 : (a < 0 ? -1 : 1); }
double Length(p a) { return sqrt(a.dot(a)); }
p Rotate(p a, double rad) { return p(a.x * cos(rad) - a.y * sin(rad), a.x * sin(rad) + a.y * cos(rad)); }
double angle(p a) { return atan2(a.y, a.x); }
double angle(p a, p b) { return atan2(a.cross(b), a.dot(b)); }
double turnAngle(p a, p b) { return mycmp(a.dot(b)) == 1 ? angle(a, b) : PI + angle(a, b); }
double distanceOfpAndLine(p a, p b, p c) { return fabs((b - a).cross(c - a) / Length(b - c)); }
double Area(int a, int b, int c, int d, p ab, p cd, polygon<double> po)
{
    double h1 = distanceOfpAndLine(po.p[a], po.p[b], po.p[b] + ab);
    double h2 = distanceOfpAndLine(po.p[c], po.p[d], po.p[d] + cd);
    return h1 * h2;
}
double max_cover_rectangle(polygon<double> po)
{
    po.p.pb(po.p[0]);
    int m = po.p.size();
    if (m < 3)
        return 0; // 沒凸包哪來外包矩形
    double Max = -1;
    double Minx = po.p[0].x, Miny = po.p[0].y, Maxx = po.p[0].x, Maxy = po.p[0].y;
    int p1 = 0, p2 = 0, p3 = 0, p4 = 0;
    p v1, v2, ori;
    ori = v1 = p(1, 0);
    v2 = p(0, 1);
    for (int i = 1; i < m; i++)
    {
        if (mycmp(Minx - po.p[i].x) == 1)
            Minx = po.p[i].x, p3 = i;
        if (mycmp(Maxx - po.p[i].x) == -1)
            Maxx = po.p[i].x, p4 = i;
        if (mycmp(Miny - po.p[i].y) == 1)
            Miny = po.p[i].y, p1 = i;
        if (mycmp(Maxy - po.p[i].y) == -1)
            Maxy = po.p[i].y, p2 = i;
    }
    while (mycmp(ori.cross(v1)) >= 0)
    {
        double minRad = 1e20;
        minRad = min(minRad, turnAngle(v1, po.p[p1 + 1] - po.p[p1]));
        minRad = min(minRad, turnAngle(v1 * (-1), po.p[p2 + 1] - po.p[p2]));
        minRad = min(minRad, turnAngle(v2 * (-1), po.p[p3 + 1] - po.p[p3]));
        minRad = min(minRad, turnAngle(v2, po.p[p4 + 1] - po.p[p4]));
        double l = 0, r = minRad;
        while (mycmp(l - r))
        {
            double len = (r - l) / 3;
            double midl = l + len;
            double midr = r - len;
            if (mycmp(Area(p1, p2, p3, p4, Rotate(v1, midl), Rotate(v2, midl), po) - Area(p1, p2, p3, p4, Rotate(v1, midr), Rotate(v2, midr), po)) == 1)
                r = midr;
            else
                l = midl;
        }
        Max = max(Max, Area(p1, p2, p3, p4, Rotate(v1, 1), Rotate(v2, 1), po));
        v1 = Rotate(v1, minRad);
        v2 = Rotate(v2, minRad);
        if (mycmp(angle(v1, po.p[p1 + 1] - po.p[p1])) == 0)
            p1 = (p1 + 1) % m;
        if (mycmp(angle(v1 * (-1), po.p[p2 + 1] - po.p[p2])) == 0)
            p2 = (p2 + 1) % m;
        if (mycmp(angle(v2 * (-1), po.p[p3 + 1] - po.p[p3])) == 0)

```



```

        p3 = (p3 + 1) % m;
        if (mycmp(angle(v2, po.p[p4 + 1] - po.p[p4])) == 0)
            p4 = (p4 + 1) % m;
    }
    return Max;
}

```

4.7 Polygon

```

template <typename T>
struct polygon
{
    polygon() {}
    vector<point<T>> p; //逆時針順序
    T area() const
    { //面積
        T ans = 0;
        for (int i = p.size() - 1, j = 0; j < (int)p.size(); i = j++)
            ans += p[i].cross(p[j]);
        return ans / 2;
    }
    point<T> center_of_mass() const
    { //重心
        T cx = 0, cy = 0, w = 0;
        for (int i = p.size() - 1, j = 0; j < (int)p.size(); i = j++)
        {
            T a = p[i].cross(p[j]);
            cx += (p[i].x + p[j].x) * a;
            cy += (p[i].y + p[j].y) * a;
            w += a;
        }
        return point<T>(cx / 3 / w, cy / 3 / w);
    }
    char ahas(const point<T> &t) const
    { //點是否在簡單多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
        bool c = 0;
        for (int i = 0, j = p.size() - 1; i < p.size(); j = i++)
            if (line<T>(p[i], p[j]).point_on_segment(t))
                return -1;
            else if ((p[i].y > t.y) != (p[j].y > t.y) &&
                    t.x < (p[j].x - p[i].x) * (t.y - p[i].y) /
                        (p[j].y - p[i].y) + p[i].x)
                c = !c;
        return c;
    }
    char point_in_convex(const point<T> &x) const
    {
        int l = 1, r = (int)p.size() - 2;
        while (l <= r)
        { //點是否在凸多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
            int mid = (l + r) / 2;
            T a1 = (p[mid] - p[0]).cross(x - p[0]);
            T a2 = (p[mid + 1] - p[0]).cross(x - p[0]);
            if (a1 >= 0 && a2 <= 0)
            {
                T res = (p[mid + 1] - p[mid]).cross(x - p[mid]);
                return res > 0 ? 1 : (res >= 0 ? -1 : 0);
            }
            else if (a1 < 0)
                r = mid - 1;
            else
                l = mid + 1;
        }
        return 0;
    }
    vector<T> getA() const
    { //凸包邊對x軸的夾角
        vector<T> res; //一定是遞增的
        for (size_t i = 0; i < p.size(); ++i)
            res.push_back((p[(i + 1) % p.size()] - p[i]).getA());
        return res;
    }
    bool line_intersect(const vector<T> &A, const line<T> &l) const
    { //O(logN)
        int f1 = upper_bound(A.begin(), A.end(), (l.p1 - l.p2).getA()) - A.begin();
        int f2 = upper_bound(A.begin(), A.end(), (l.p2 - l.p1).getA()) - A.begin();
        return l.cross_seg(line<T>(p[f1], p[f2]));
    }
    polygon cut(const line<T> &l) const
    { //凸包對直線切割，得到直線l左側的凸包
        polygon ans;
        for (int n = p.size(), i = n - 1, j = 0; j < n; i = j++)
        {
            if (l.ori(p[i]) >= 0)

```

```

        {
            ans.p.push_back(p[i]);
            if (l.ori(p[j]) < 0)
                ans.p.push_back(l.line_intersection(line<T>(p[i], p[j])));
        }
        else if (l.ori(p[j]) > 0)
            ans.p.push_back(l.line_intersection(line<T>(p[i], p[j])));
        }
        return ans;
    }
    static bool Andrew_Monotone_Chain_angle(const point<T> &a, const point<T> &b)
    { //凸包排序函數 // 起始點不同
        return (a.y < b.y) || (a.y == b.y && a.x < b.x); //Y最小開始
    }
    void Andrew_Monotone_Chain(vector<point<T>> &s)
    { //凸包 Convexhull 2D
        sort(s.begin(), s.end(), Andrew_Monotone_Chain_angle);
        p.resize(s.size() + 1);
        int m = 0;
        // cross >= 0 順時針，cross <= 0 逆時針旋轉
        for (size_t i = 0; i < s.size(); ++i)
        {
            while (m >= 2 && (p[m - 1] - p[m - 2]).cross(s[i] - p[m - 2]) <= 0)
                --m;
            p[m++] = s[i];
        }
        for (int i = s.size() - 2, t = m + 1; i >= 0; --i)
        {
            while (m >= t && (p[m - 1] - p[m - 2]).cross(s[i] - p[m - 2]) <= 0)
                --m;
            p[m++] = s[i];
        }
        if (s.size() > 1) // 重複頭一次需扣掉
            --m;
        p.resize(m);
        // p.pb(s[0]); // 需要頭在 pb 回去!!
    }
    T diam()
    { //直徑
        int n = p.size(), t = 1;
        T ans = 0;
        p.push_back(p[0]);
        for (int i = 0; i < n; i++)
        {
            point<T> now = p[i + 1] - p[i];
            while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
                t = (t + 1) % n;
            ans = max(ans, (p[i] - p[t]).abs2());
        }
        return p.pop_back(), ans;
    }
    T min_cover_rectangle()
    { // 先做凸包 //最小覆蓋矩形
        int n = p.size(), t = 1, r = 1, l;
        if (n < 3)
            return 0; //也可以做最小周長矩形
        T ans = 1e99;
        p.push_back(p[0]);
        for (int i = 0; i < n; i++)
        {
            point<T> now = p[i + 1] - p[i];
            while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
                t = (t + 1) % n;
            while (now.dot(p[r + 1] - p[i]) > now.dot(p[r] - p[i]))
                r = (r + 1) % n;
            if (l)
                l = r;
            while (now.dot(p[l + 1] - p[i]) <= now.dot(p[l] - p[i]))
                l = (l + 1) % n;
            T d = now.abs2();
            T tmp = now.cross(p[t] - p[i]) * (now.dot(p[r] - p[i]) - now.dot(p[l] - p[i])) / d;
            ans = min(ans, tmp);
        }
        return p.pop_back(), ans;
    }
    T dis2(polygon &p1)
    { //凸包最近距離平方
        vector<point<T>> &P = p, &Q = p1.p;
        int n = P.size(), m = Q.size(), l = 0, r = 0;
        for (int i = 0; i < n; ++i)
            if (P[i].y < P[l].y)
                l = i;
        for (int i = 0; i < m; ++i)
            if (Q[i].y < Q[r].y)

```

```

        r = i;
        P.push_back(P[0]), Q.push_back(Q[0]);
        T ans = 1e99;
        for (int i = 0; i < n; ++i)
        {
            while ((P[l] - P[l + 1]).cross(Q[r + 1] - Q[r]) < 0)
            {
                r = (r + 1) % m;
                ans = min(ans, line<T>(P[l], P[l + 1]).seg_dis2(
                    line<T>(Q[r], Q[r + 1]));
                l = (l + 1) % n;
            }
            return P.pop_back(), Q.pop_back(), ans;
        }
        static char sign(const point<T> &t)
        {
            return (t.y == 0 ? t.x : t.y) < 0;
        }
        static bool angle_cmp(const line<T> &A, const line<T> &B)
        {
            point<T> a = A.p2 - A.p1, b = B.p2 - B.p1;
            return sign(a) < sign(b) || (sign(a) == sign(b) && a.
                cross(b) > 0);
        }
        int halfplane_intersection(vector<line<T>> &s)
        {
            // 半平面交
            sort(s.begin(), s.end(), angle_cmp); // 線段左側為該線段
            // 半平面
            int L, R, n = s.size();
            vector<point<T>> px(n);
            vector<line<T>> q(n);
            q[L = R = 0] = s[0];
            for (int i = 1; i < n; ++i)
            {
                while (L < R && s[i].ori(px[R - 1]) <= 0)
                    --R;
                while (L < R && s[i].ori(px[L]) <= 0)
                    ++L;
                q[++R] = s[i];
                if (q[R].parallel(q[R - 1]))
                {
                    --R;
                    if (q[R].ori(s[i].p1) > 0)
                        q[R] = s[i];
                }
                if (L < R)
                    px[R - 1] = q[R - 1].line_intersection(q[R]);
            }
            while (L < R && q[L].ori(px[R - 1]) <= 0)
                --L;
            p.clear();
            if (R - L <= 1)
                return 0;
            px[R] = q[R].line_intersection(q[L]);
            for (int i = L; i <= R; ++i)
                p.push_back(px[i]);
            return R - L + 1;
        }
    };

```

4.8 Smallest Circle

```

pair<Point, Double> SmallestCircle(int n, Point _p[])
{
    Point *p = new Point[n];
    memcpy(p, _p, sizeof(Point) * n);
    random_shuffle(p, p + n);
    Double r2 = 0;
    Point cen;
    for (int i = 0; i < n; i++)
    {
        if (abs2(cen - p[i]) <= r2)
            continue;
        cen = p[i], r2 = 0;
        for (int j = 0; j < i; j++)
        {
            if (abs2(cen - p[j]) <= r2)
                continue;
            cen = (p[i] + p[j]) * 0.5;
            r2 = abs2(cen - p[i]);
            for (int k = 0; k < j; k++)
            {
                if (abs2(cen - p[k]) <= r2)
                    continue;
                cen = circumcentre(p[i], p[j], p[k]);
                r2 = abs2(cen - p[k]);
            }
        }
    }
    delete[] p;
    return {cen, r2};
}

```

4.9 Triangle

```

template <typename T>
struct triangle
{
    point<T> a, b, c;
    triangle() {}
    triangle(const point<T> &a, const point<T> &b, const point<
        T> &c) : a(a), b(b), c(c) {}
    T area() const
    {
        T t = (b - a).cross(c - a) / 2;
        return t > 0 ? t : -t;
    }
    point<T> barycenter() const
    {
        // 重心
        return (a + b + c) / 3;
    }
    point<T> circumcenter() const
    {
        // 外心
        static line<T> u, v;
        u.p1 = (a + b) / 2;
        u.p2 = point<T>(u.p1.x - a.y + b.y, u.p1.y + a.x - b.x);
        v.p1 = (a + c) / 2;
        v.p2 = point<T>(v.p1.x - a.y + c.y, v.p1.y + a.x - c.x);
        return u.line_intersection(v);
    }
    point<T> incenter() const
    {
        // 內心
        T A = sqrt((b - c).abs2()), B = sqrt((a - c).abs2()), C
            = sqrt((a - b).abs2());
        return point<T>(A * a.x + B * b.x + C * c.x, A * a.y +
            B * b.y + C * c.y) / (A + B + C);
    }
    point<T> perpcenter() const
    {
        // 垂心
        return barycenter() * 3 - circumcenter() * 2;
    }
};

```

5 Graph

5.1 BCC edge

```

struct BccEdge {
    static const int MXN = 100005;
    struct Edge {
        int v, eid;
    };
    int n, m, step, par[MXN], dfn[MXN], low[MXN];
    vector<Edge> E[MXN];
    DisjointSet djs;
    void init(int _n) {
        n = _n;
        m = 0;
        for (int i = 0; i < n; i++)
            E[i].clear();
        djs.init(n);
    }
    void add_edge(int u, int v) {
        E[u].PB({v, m});
        E[v].PB({u, m});
        m++;
    }
    void DFS(int u, int f, int f_eid) {
        par[u] = f;
        dfn[u] = low[u] = step++;
        for (auto it : E[u]) {
            if (it.eid == f_eid)
                continue;
            int v = it.v;
            if (dfn[v] == -1) {
                DFS(v, u, it.eid);
                low[u] = min(low[u], low[v]);
            }
            else {
                low[u] = min(low[u], dfn[v]);
            }
        }
    }
    void solve() {
        step = 0;
        memset(dfn, -1, sizeof(int) * n);
        for (int i = 0; i < n; i++) {
            if (dfn[i] == -1)
                DFS(i, i, -1);
            djs.init(n);
            for (int i = 0; i < n; i++) {
                if (low[i] < dfn[i])
                    djs.uni(i, par[i]);
            }
        }
    }
};
graph;

```

5.2 BCC vertex

```
struct BccVertex
{
    int n, nBcc, step, dfn[MXN], low[MXN], top, stk[MXN];
    vector<int> E[MXN], bccv[MXN];
    void init(int _n)
    {
        n = _n;
        nBcc = step = 0;
        for (int i = 0; i < n; i++)
            E[i].clear();
    }
    void addEdge(int u, int v)
    {
        E[u].push_back(v);
        E[v].push_back(u);
    }
    void DFS(int u, int f)
    {
        dfn[u] = low[u] = step++;
        stk[top++] = u;
        for (auto v : E[u])
        {
            if (v == f)
                continue;
            if (dfn[v] == -1)
            {
                DFS(v, u);
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u])
                {
                    int z;
                    bccv[nBcc].clear();
                    do
                    {
                        z = stk[--top];
                        bccv[nBcc].push_back(z);
                    } while (z != v);
                    bccv[nBcc++].push_back(u);
                }
            }
            else
                low[u] = min(low[u], dfn[v]);
        }
    }
    vector<vector<int>> solve()
    {
        vector<vector<int>> res;
        for (int i = 0; i < n; i++)
            dfn[i] = low[i] = -1;
        for (int i = 0; i < n; i++)
            if (dfn[i] == -1)
            {
                top = 0;
                DFS(i, i);
            }
        for (int i = 0; i < nBcc; i++)
            res.push_back(bccv[i]);
        return res;
    }
} graph;
```

5.3 Bellman Ford

```
/*SPA - Bellman-Ford*/
#define inf 99999 //define by you maximum edges weight
vector<vector<int>> edges;
vector<int> dist;
vector<int> ancestor;
void BellmanFord(int start, int node)
{
    dist[start] = 0;
    for (int it = 0; it < node-1; it++)
    {
        for (int i = 0; i < node; i++)
        {
            for (int j = 0; j < node; j++)
            {
                if (edges[i][j] != -1)
                {
                    if (dist[i] + edges[i][j] < dist[j])
                    {
                        dist[j] = dist[i] + edges[i][j];
                        ancestor[j] = i;
                    }
                }
            }
        }
    }
}

for (int i = 0; i < node; i++) //negative cycle detection
    for (int j = 0; j < node; j++)
        if (dist[i] + edges[i][j] < dist[j])
        {
            cout << "Negative cycle!" << endl;
            return;
        }
}

int main()
{
    int node;
    cin >> node;
    edges.resize(node, vector<int>(node, inf));
```

```
dist.resize(node, inf);
ancestor.resize(node, -1);
int a, b, d;
while (cin >> a >> b >> d)
{
    /*input: source destination weight*/
    if (a == -1 && b == -1 && d == -1)
        break;
    edges[a][b] = d;
}
int start;
cin >> start;
BellmanFord(start, node);
return 0;
```

5.4 BFS queue

```
/*BFS - queue version*/
void BFS(vector<int> &result, vector<pair<int, int>> edges, int
node, int start)
{
    vector<int> pass(node, 0);
    queue<int> q;
    queue<int> p;
    q.push(start);
    int count = 1;
    vector<pair<int, int>> newedges;
    while (!q.empty())
    {
        pass[q.front()] = 1;
        for (int i = 0; i < edges.size(); i++)
        {
            if (edges[i].first == q.front() && pass[edges[i].
                second] == 0)
            {
                p.push(edges[i].second);
                result[edges[i].second] = count;
            }
            else if (edges[i].second == q.front() && pass[edges
                [i].first] == 0)
            {
                p.push(edges[i].first);
                result[edges[i].first] = count;
            }
            else
                newedges.push_back(edges[i]);
        }
        edges = newedges;
        newedges.clear();
        q.pop();
        if (q.empty() == true)
        {
            q = p;
            queue<int> tmp;
            p = tmp;
            count++;
        }
    }
}

int main()
{
    int node;
    cin >> node;
    vector<pair<int, int>> edges;
    int a, b;
    while (cin >> a >> b)
    {
        /*a = b = -1 means input edges ended*/
        if (a == -1 && b == -1)
            break;
        edges.push_back(pair<int, int>(a, b));
    }
    vector<int> result(node, -1);
    BFS(result, edges, node, 0);

    return 0;
```

5.5 DFS rec

```
/*DFS - Recursive version*/
map<pair<int, int>, int> edges;
vector<int> pass;
vector<int> route;
void DFS(int start)
{
    pass[start] = 1;
    map<pair<int, int>, int>::iterator iter;
    for (iter = edges.begin(); iter != edges.end(); iter++)
    {
        if ((*iter).first.first == start && (*iter).second == 0
            && pass[(*iter).first.second] == 0)
        {
            route.push_back((*iter).first.second);
            DFS((*iter).first.second);
        }
        else if ((*iter).first.second == start && (*iter).second
            == 0 && pass[(*iter).first.first] == 0)
        {
            route.push_back((*iter).first.first);
```

```

        DFS((*iter).first.first);
    }
}
int main(){
    int node;
    cin>>node;
    pass.resize(node,0);
    int a,b;
    while(cin>>a>>b){
        if(a == -1 && b == -1)
            break;
        edges.insert(pair<pair<int,int>,int>(pair<int,int>(a,b),0));
    }
    int start;
    cin>>start;
    route.push_back(start);
    DFS(start);
    return 0;
}

```

5.6 Dijkstra

```

/*SPA - Dijkstra*/
const int MAXN = 1e5 + 3;
const int inf = INT_MAX;
typedef pair<int, int> pii;
vector<vector<pii>> weight(MAXN);
vector<int> isDone(MAXN, false), dist, ancestor;
void dijkstra(int s)
{
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push(pii(0, s));
    ancestor[s] = -1;
    while (!pq.empty())
    {
        int u = pq.top().second;
        pq.pop();

        isDone[u] = true;

        for (auto &pr : weight[u])
        {
            int v = pr.first, w = pr.second;

            if (!isDone[v] && dist[u] + w < dist[v])
            {
                dist[v] = dist[u] + w;
                pq.push(pii(dist[v], v));
                ancestor[v] = u;
            }
        }
    }
    // weight[a - 1].push_back(pii(b - 1, w));
    // weight[b - 1].push_back(pii(a - 1, w));
    // dist.resize(n, inf);
    // ancestor.resize(n, -1);
    // dist[0] = 0;
    // dijkstra(0);
}

```

5.7 Euler circuit

```

/*Euler circuit*/
/*From NTU kiseki*/
/*G is graph, vis is visited, la is path*/
bool vis[N];
size_t la[K];
void dfs(int u, vector<int> &vec)
{
    while (la[u] < G[u].size())
    {
        if (vis[G[u][la[u]].second])
        {
            ++la[u];
            continue;
        }
        int v = G[u][la[u]].first;
        vis[G[u][la[u]].second] = true;
        ++la[u];
        dfs(v, vec);
        vec.push_back(v);
    }
}

```

5.8 Floyd warshall

```

/*SPA - Floyd-Warshall*/
// 有向圖 · 正邊 O(V³)
// 有向圖 · 無負環 O(V³)
// 有向圖 · 有負環 不適用

// 無向圖 · 正邊 O(V³)
// 無向圖 · 無負環 不適用

```

```

// 無向圖 · 有負環 不適用
/*Find min weight cycle*/
#define inf 99999
void floyd_warshall(vector<vector<int>> &distance, vector<vector<int>> &ancestor, int n)
{
    for (int k = 0; k < n; k++)
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (distance[i][k] + distance[k][j] < distance[i][j])
                {
                    distance[i][j] = distance[i][k] + distance[k][j];
                    ancestor[i][j] = ancestor[k][j];
                }
            }
        }
    }
}

vector<vector<int>> distance(n, vector<int>(n, inf));
vector<vector<int>> ancestor(n, vector<int>(n, -1));
distance[a][b] = w;
ancestor[a][b] = w;
floyd_warshall(distance, ancestor, n);
/*Negative cycle detection*/
for (int i = 0; i < n; i++)
{
    if (distance[i][i] < 0)
    {
        cout << "Negative cycle!" << endl;
        break;
    }
}
}

```

5.9 Hamilton cycle

```

/*find hamilton cycle*/
void hamilton(vector<vector<int>> gp, int k, int cur, vector<int> & solution, vector<bool> pass, bool & flag)
{
    if(k == gp.size()-1){
        if(gp[cur][1] == 1){
            cout << 1 << " ";
            while(cur != 1){
                cout << cur << " ";
                cur = solution[cur];
            }
            cout << cur << endl;
            flag = true;
            return;
        }
    }
    for (int i = 0; i < gp[cur].size() && !flag; i++){
        if(gp[cur][i] == 1 && !pass[i]){
            pass[i] = true;
            solution[i] = cur;
            hamilton(gp, k + 1, i, solution, pass, flag);
            pass[i] = false;
        }
    }
}

int main(){
    int n;
    while(cin>>n){
        int a,b;
        bool end = false;
        vector<vector<int>> gp(n+1,vector<int>(n+1,0));
        while(cin>>a>>b){
            if(a == 0 && b == 0)
                break;
            gp[a][b] = 1;
            gp[b][a] = 1;
        }
        vector<int> solution(n + 1, -1);
        vector<bool> pass(n + 1, false);
        solution[1] = 0;
        pass[1] = true;
        bool flag = false;
        hamilton(gp, 1, 1, solution, pass, flag);
        if(!flag)
            cout << "N" << endl;
    }
    return 0;
}

```

```

/*
4
1 2
2 3
2 4
3 4
3 1
0 0
output: 1 3 4 2 1
*/

```

```
*/
```

5.10 Kruskal

```
/*mst - Kruskal*/
struct edges{
    int from;
    int to;
    int weight;
    friend bool operator < (edges a, edges b){
        return a.weight > b.weight;
    }
};

int find(int x, vector<int>& union_set){
    if(x != union_set[x])
        union_set[x] = find(union_set[x], union_set);
    return union_set[x];
}

void merge(int a, int b, vector<int>& union_set){
    int pa = find(a, union_set);
    int pb = find(b, union_set);
    if(pa != pb)
        union_set[pa] = pb;
}

void kruskal(priority_queue<edges> pq, int n){
    vector<int> union_set(n, 0);
    for (int i = 0; i < n; i++)
        union_set[i] = i;
    int edge = 0;
    int cost = 0; //evaluate cost of mst
    while(!pq.empty() && edge < n - 1){
        edges cur = pq.top();
        int from = find(cur.from, union_set);
        int to = find(cur.to, union_set);
        if(from != to){
            merge(from, to, union_set);
            edge += 1;
            cost += cur.weight;
        }
        pq.pop();
    }
    if(edge < n-1)
        cout << "No mst" << endl;
    else
        cout << cost << endl;
}

int main(){
    int n;
    cin >> n;
    int a, b, d;
    priority_queue<edges> pq;
    while(cin >> a >> b >> d){
        if(a == -1 && b == -1 && d == -1)
            break;
        edges tmp;
        tmp.from = a;
        tmp.to = b;
        tmp.weight = d;
        pq.push(tmp);
    }
    kruskal(pq, n);
    return 0;
}
```

5.11 LCA

```
bool adj[9][9]; // adjacency matrix
bool visit[9]; // DFS當下已經拜訪過的點
int lca[9][9]; // 所有兩點之間的LCA
int p[9]; // Disjoint-sets Forest
// 最多兩步
int find(int x)
{
    return x == p[x] ? x : (p[x] = find(p[x]));
}

int DFS(int x)
{
    if (visit[x])
        return;
    visit[x] = true;
    // 計算LCA
    for (int y = 0; y < 9; ++y)
        if (visit[y])
            lca[x][y] = lca[y][x] = find(y);
    // DFS
    for (int y = 0; y < 9; ++y)
        if (adj[x][y])
        {
            DFS(y);
            p[y] = x; // merge(y, x) · 並讓x是樹根。
        }
}

void demo()
{
}
```

```
for (int i = 0; i < 9; ++i)
    p[i] = i;
for (int i = 0; i < 9; ++i)
    visit[i] = false;
DFS(0); // 假設樹根為0
int x, y;
while (cin >> x >> y)
    cout << "x點與y點的LCA是" << lca[x][y];
}
```

5.12 Minimum Weight Cycle

```
// 最小環
// 圖上無負環 !!!!
#define INF 99999
vector<vector<int>> w, d, p;
vector<int> cycle;
int c = 0;
void trace(int i, int j)
{
    cycle[c++] = i;
    if (i != j)
        trace(p[i][j], j);
}

void init(int n)
{
    for (int i = 0; i < n; ++i)
        d[i][i] = 0;
}

void minimum_cycle(int n)
{
    int weight = 1e9;
    for (int k = 0; k < n; ++k)
    {
        for (int i = 0; i < k; ++i)
            for (int j = 0; j < k; ++j)
                if (i != j)
                    if (w[k][i] + d[i][j] + w[j][k] < weight)
                    {
                        weight = w[k][i] + d[i][j] + w[j][k];
                        c = 0;
                        trace(i, j);
                        cycle[c++] = k;
                    }
    }

    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            if (d[i][k] + d[k][j] < d[i][j])
            {
                d[i][j] = d[i][k] + d[k][j];
                p[i][j] = p[i][k];
            }
        }
    }

    if (weight == 1e9)
        cout << "No exist";
    else
    {
        bug(weight);
        bug(c);
        bugarr(cycle);
    }
}

void simple_minimum_cycle(int n) // No use vector p
{
    int weight = INF;
    for (int k = 0; k < n; ++k)
    {
        for (int i = 0; i < k; ++i)
            for (int j = 0; j < k; ++j)
                if (i != j)
                    weight = min(mp[k][i] + d[i][j] + mp[j][k], weight);

        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j)
                d[i][j] = min(d[i][k] + d[k][j], d[i][j]);
    }

    if (weight == INF)
        cout << "Back to jail\n";
    else
        cout << weight << endl;
}

w.resize(n, vector<int>(n, INF));
d.resize(n, vector<int>(n, INF));
p.resize(n, vector<int>(n));
cycle.resize(n);
//Edge input
w[a][b] = w;
d[a][b] = w;
p[a][b] = b;
init(n);
minimum_cycle(n);
}
```

5.13 Prim

```

/*mst - Prim*/
#define inf 99999
struct edges
{
    int from;
    int to;
    int weight;
    friend bool operator<(edges a, edges b)
    {
        return a.weight > b.weight;
    }
};

void Prim(vector<vector<int>> gp, int n, int start)
{
    vector<bool> pass(n, false);
    int edge = 0;
    int cost = 0; //evaluate cost of mst
    priority_queue<edges> pq;
    for (int i = 0; i < n; i++)
    {
        if (gp[start][i] != inf)
        {
            edges tmp;
            tmp.from = start;
            tmp.to = i;
            tmp.weight = gp[start][i];
            pq.push(tmp);
        }
    }
    pass[start] = true;
    while (!pq.empty() && edge < n - 1)
    {
        edges cur = pq.top();
        pq.pop();
        if (!pass[cur.to])
        {
            for (int i = 0; i < n; i++)
            {
                if (gp[cur.to][i] != inf)
                {
                    edges tmp;
                    tmp.from = cur.to;
                    tmp.to = i;
                    tmp.weight = gp[cur.to][i];
                    pq.push(tmp);
                }
            }
            pass[cur.to] = true;
            edge += 1;
            cost += cur.weight;
        }
    }
    if (edge < n - 1)
        cout << "No mst" << endl;
    else
        cout << cost << endl;
}

int main()
{
    int n;
    cin >> n;
    int a, b, d;
    vector<vector<int>> gp(n, vector<int>(n, inf));
    while (cin >> a >> b >> d)
    {
        if (a == -1 && b == -1 && d == -1)
            break;
        if (gp[a][b] > d)
            gp[a][b] = d;
    }
    Prim(gp, n, 0);
    return 0;
}

```

5.14 Union find

```

// union_find from 台大
vector<int> father;
vector<int> people;
void init(int n)
{
    father.clear();
    people.clear();
    father.resize(n);
    people.resize(n);
    for (int i = 0; i < n; i++)
    {
        father[i] = i;
        people[i] = 1;
    }
}

int Find(int x)
{
    if (x != father[x])
        father[x] = Find(father[x]);
}

```

```

return father[x];
}

void Union(int x, int y)
{
    int m = Find(x);
    int n = Find(y);
    if (m != n)
    {
        father[n] = m;
        people[m] += people[n];
    }
}

```

6 Mathematics

6.1 Catalan

Catalan number

- 0~19項的catalan number
 - 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

6.2 Combination

```

/*input type string or vector*/
for (int i = 0; i < (1 << input.size()); ++i)
{
    string testCase = "";
    for (int j = 0; j < input.size(); ++j)
        if (i & (1 << j))
            testCase += input[j];
}

```

6.3 CRT

```

// 中國剩餘定理
template <typename T>
tuple<T, T, T> extgcd(T a, T b)
{
    if (!b)
        return make_tuple(a, 1, 0);
    T d, x, y;
    tie(d, x, y) = extgcd(b, a % b);
    return make_tuple(d, y, x - (a / b) * y);
}

long long crt(vector<int> mod, vector<int> a)
{
    // x % mod[i] = a[i]
    long long mult = mod[0];
    int n = (int)mod.size();
    long long res = a[0];
    for (int i = 1; i < n; ++i)
    {
        long long d, x, y;
        tie(d, x, y) = extgcd(mult, mod[i] * 1ll);
        if ((a[i] - res) % d)
            return -1;
        long long new_mult = mult / __gcd(mult, 1ll * mod[i]) * mod[i];
        res += x * ((a[i] - res) / d) % new_mult * mult % new_mult;
        mult = new_mult;
        ((res %= mult) += mult) %= mult;
    }
    return res;
}

```

6.4 Extended Euclidean

```

// ax + by = gcd(a,b)
pair<long long, long long> extgcd(long long a, long long b)
{
    if (b == 0)
        return {1, 0};
    long long k = a / b;
    pair<long long, long long> p = extgcd(b, a - k * b);
    //cout << p.first << " " << p.second << endl;
    //cout << "商數(k)= " << k << endl << endl;
    return {p.second, p.first - k * p.second};
}

int main()
{
    int a, b;
    cin >> a >> b;
    pair<long long, long long> xy = extgcd(a, b); //(x0,y0)
}

```



```

    cout << xy.first << " " << xy.second << endl;
    cout << xy.first << " * " << a << " + " << xy.second << " *
    " << b << endl;
    return 0;
}
// ax + by = gcd(a,b) * r
/*find |x|+|y| -> min*/
int main()
{
    long long r, p, q; /*px+qy = r*/
    int cases;
    cin >> cases;
    while (cases--)
    {
        cin >> r >> p >> q;
        pair<long long, long long> xy = extgcd(q, p); //(x0,y0)
        long long ans = 0, tmp = 0;
        double k, k1;
        long long s, s1;
        k = 1 - (double)(r * xy.first) / p;
        s = round(k);
        ans = llabs(r * xy.first + s * p) + llabs(r * xy.second
            - s * q);
        k1 = -(double)(r * xy.first) / p;
        s1 = round(k1);
        /*cout << k << endl << k1 << endl;
        cout << s << endl << s1 << endl;*/
        tmp = llabs(r * xy.first + s1 * p) + llabs(r * xy.
            second - s1 * q);
        ans = min(ans, tmp);

        cout << ans << endl;
    }
    return 0;
}

```

6.5 Fermat

- $a^{(p-1)} \equiv 1 \pmod{p} \Leftrightarrow a * a^{(p-2)} \equiv 1$
 - $a^{(p-2)} \equiv 1/a$
- 同餘因數定理
 - $a \equiv b \pmod{p} \Leftrightarrow k|a - b$
- 同餘加法性質
 - $a \equiv b \pmod{p}$ and $c \equiv d \pmod{p}$
 $\Leftrightarrow a + c \equiv b + d \pmod{p}$
- 同餘相乘性質
 - $a \equiv b \pmod{p}$ and $c \equiv d \pmod{p}$
 $\Leftrightarrow ac \equiv bd \pmod{p}$
- 同餘次方性質
 - $a \equiv b \pmod{p} \Leftrightarrow a^n \equiv b^n \pmod{p}$
- 同餘倍方性質
 - $a \equiv b \pmod{p} \Leftrightarrow am \equiv bm \pmod{p}$

6.6 Hex to Dec

```

int HextoDec(string num) //16 to 10
{
    int base = 1;
    int temp = 0;
    for (int i = num.length() - 1; i >= 0; i--)
    {
        if (num[i] >= '0' && num[i] <= '9')
        {
            temp += (num[i] - 48) * base;
            base = base * 16;
        }
        else if (num[i] >= 'A' && num[i] <= 'F')
        {
            temp += (num[i] - 55) * base;
            base = base * 16;
        }
    }
    return temp;
}
void DecToHex(int p) //10 to 16
{
    char *l = new (char);
    sprintf(l, "%X", p);
    //int l_intResult = stoi(l);
    cout << l << "\n";
    //return l_intResult;
}

```

6.7 Log

```

double mylog(double a, double base)
{
    //a 的對數底數 b = 自然對數 (a) / 自然對數 (b) *
    return log(a) / log(base);
}

```

6.8 Mod 性質

加法： $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$
 減法： $(a - b) \bmod p = (a \bmod p - b \bmod p + p) \bmod p$
 乘法： $(a * b) \bmod p = (a \bmod p * b \bmod p) \bmod p$
 次方： $(a^b) \bmod p = ((a \bmod p)^b) \bmod p$
 加法結合律： $((a + b) \bmod p + c) \bmod p = (a + (b + c)) \bmod p$
 乘法結合律： $((a * b) \bmod p * c) \bmod p = (a * (b * c)) \bmod p$
 加法交換律： $(a + b) \bmod p = (b + a) \bmod p$
 乘法交換律： $(a * b) \bmod p = (b * a) \bmod p$
 結合律： $((a + b) \bmod p * c) \bmod p = ((a * c) \bmod p + (b * c) \bmod p) \bmod p$

如果 $a \equiv b \pmod{m}$ ，我們會說 a, b 在模 m 下同餘。

以下為性質：

- 整除性： $a \equiv b \pmod{m} \Rightarrow c * m = a - b, c \in \mathbb{Z}$
 $\Rightarrow a \equiv b \pmod{m} \Rightarrow m | a - b$
- 遞移性：若 $a \equiv b \pmod{c}, b \equiv d \pmod{c}$
 則 $a \equiv d \pmod{c}$
- 保持基本運算：

$$\begin{cases} a \equiv b \pmod{m} \\ c \equiv d \pmod{m} \end{cases} \Rightarrow \begin{cases} a \pm c \equiv b \pm d \pmod{m} \\ a * c \equiv b * d \pmod{m} \end{cases}$$

- 放大縮小模數：

$$k \in \mathbb{Z}^+, a \equiv b \pmod{m} \Leftrightarrow k * a \equiv k * b \pmod{k * m}$$

模逆元是取模下的反元素，即為找到 a^{-1} 使得 $aa^{-1} \equiv 1 \pmod{c}$ 。

整數 a 在模 c 下要有模反元素的充分必要條件為 a, c 互質。

模逆元如果存在會有無限個，任意兩相鄰模逆元相差 c 。

費馬小定理

給定一個質數 p 及一個整數 a ，那麼： $a^p \equiv a \pmod{p}$ 如果 $\gcd(a, p) = 1$ ，則： $a^{p-1} \equiv 1 \pmod{p}$

歐拉定理

歐拉定理是比較 general 版本的費馬小定理。給定兩個整數 n 和 a ，如果 $\gcd(a, n) = 1$ ，則： $a^{\Phi(n)} \equiv 1 \pmod{n}$ 如果 n 是質數， $\Phi(n) = n - 1$ ，也就是費馬小定理。

Wilson's theorem

給定一個質數 p ，則： $(p - 1)! \equiv -1 \pmod{p}$

6.9 PI

```

#define PI acos(-1)
#define PI M_PI

```

6.10 Pow_Mod

```

int pow_mod(int a, int n, int m) // a ^ n mod m;
{
    // a, n, m < 10 ^ 9
    if (n == 0)
        return 1;
    int x = pow_mid(a, n / 2, m);
    long long ans = (long long)x * x % m;
    if (n % 2 == 1)
        ans = ans * a % m;
    return (int)ans;
}
int inv(int a, int n, int p) // n = p-2
{
    long long res = 1;
    for (; n >= 1, (a *= a) %= p)
        if (n & 1)
            (res *= a) %= p;
    return res;
}

```

6.11 Prime table

```

const int maxn = 10e9;
vector<int> p;
bitset<maxn> is_notp;
void PrimeTable()
{
    is_notp.reset();
    is_notp[0] = is_notp[1] = 1;
    for (int i = 2; i <= maxn; ++i)
    {
        if (!is_notp[i])

```

```

        p.push_back(i);
    for (int j = 0; j < (int)p.size(); ++j)
    {
        if (i * p[j] > maxn)
            break;
        is_notp[i * p[j]] = 1;
        if (i % p[j] == 0)
            break;
    }
}
}

```

6.12 Prime 判斷

```

typedef long long ll;
ll modmul(ll a, ll b, ll mod)
{
    ll ret = 0;
    for (; b >= 1, a = (a + a) % mod;
        if (b & 1)
            ret = (ret + a) % mod;
    return ret;
}
ll qpow(ll x, ll u, ll mod)
{
    ll ret = 1ll;
    for (; u >= 1, x = modmul(x, x, mod);
        if (u & 1)
            ret = modmul(ret, x, mod);
    return ret;
}
ll gcd(ll a, ll b)
{
    return b ? gcd(b, a % b) : a;
}
ll Pollard_Rho(ll n, ll c)
{
    ll i = 1, j = 2, x = rand() % (n - 1) + 1, y = x;
    while (1)
    {
        i++;
        x = (modmul(x, x, n) + c) % n;
        ll p = gcd((y - x + n) % n, n);
        if (p != 1 && p != n)
            return p;
        if (y == x)
            return n;
        if (i == j)
        {
            y = x;
            j <<= 1;
        }
    }
}
bool Miller_Rabin(ll n)
{
    ll x, pre, u = n - 1;
    int i, j, k = 0;
    if (n == 2 || n == 3 || n == 5 || n == 7 || n == 11)
        return 1;
    if (n == 1 || !(n % 2) || !(n % 3) || !(n % 5) || !(n % 7) || !(n % 11))
        return 0;
    while (!(u & 1))
    {
        k++;
        u >>= 1;
    }
    srand((long long)12234336);
    for (i = 1; i <= 50; i++)
    {
        x = rand() % (n - 2) + 2;
        if (!(n % x))
            return 0;
        x = qpow(x, u, n);
        pre = x;
        for (j = 1; j <= k; j++)
        {
            x = modmul(x, x, n);
            if (x == 1 && pre != 1 && pre != n - 1)
                return 0;
            pre = x;
        }
        if (x != 1)
            return 0;
    }
    return 1;
}
// if (Miller_Rabin(n)) puts("Prime");

```

6.13 Round(小數)

```

double myround(double number, unsigned int bits)
{
    LL integerPart = number;
    number -= integerPart;

```

```

    for (unsigned int i = 0; i < bits; ++i)
        number *= 10;
    number = (LL)(number + 0.5);
    for (unsigned int i = 0; i < bits; ++i)
        number /= 10;
    return integerPart + number;
}
//printf("%.1f\n", round(3.4515239, 1));

```

6.14 二分逼近法

```

#define eps 1e-14
void half_interval()
{
    double L = 0, R = /*區間*/, M;
    while (R - L >= eps)
    {
        M = (R + L) / 2;
        if (/*函數*/ > /*方程式目標*/)
            L = M;
        else
            R = M;
    }
    printf("%.3lf\n", R);
}

```

6.15 公式

$$S_n = \frac{a(1-r^n)}{1-r} \quad a_n = \frac{a_1 + a_n}{2} \sum_{k=1}^n k = \frac{n(n+1)}{2} \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^n k^3 = \left[\frac{n(n+1)}{2} \right]^2$$

6.16 四則運算

```

string s = ""; //開頭是負號要補0
long long int DFS(int le, int ri) // (0, string final index)
{
    int c = 0;
    for (int i = ri; i >= le; i--)
    {
        if (s[i] == '(')
            c++;
        if (s[i] == '(')
            c--;
        if (s[i] == '+' && c == 0)
            return DFS(le, i - 1) + DFS(i + 1, ri);
        if (s[i] == '-' && c == 0)
            return DFS(le, i - 1) - DFS(i + 1, ri);
    }
    for (int i = ri; i >= le; i--)
    {
        if (s[i] == '(')
            c++;
        if (s[i] == '(')
            c--;
        if (s[i] == '*' && c == 0)
            return DFS(le, i - 1) * DFS(i + 1, ri);
        if (s[i] == '/' && c == 0)
            return DFS(le, i - 1) / DFS(i + 1, ri);
        if (s[i] == '%' && c == 0)
            return DFS(le, i - 1) % DFS(i + 1, ri);
    }
    if ((s[le] == '(' && (s[ri] == ')'))
        return DFS(le + 1, ri - 1); //去除刮號
    if (s[le] == ' ' && s[ri] == ' ')
        return DFS(le + 1, ri - 1); //去除左右兩邊空格
    if (s[le] == ' ')
        return DFS(le + 1, ri); //去除左邊空格
    if (s[ri] == ' ')
        return DFS(le, ri - 1); //去除右邊空格
    long long int num = 0;
    for (int i = le; i <= ri; i++)
        num = num * 10 + s[i] - '0';
    return num;
}

```

6.17 因數表

```

const int limit = 10000000;
vector<vector<int>> arr(limit);
for (int i = 1; i <= limit; i++)
{
    for (int j = i; j <= limit; j += i)
        arr[j].pb(i); // i 為因數
}

```

6.18 數字乘法組合

```

void dfs(int j, int old, int num, vector<int> com, vector<
vector<int>> &ans)
{
    for (int i = j; i <= sqrt(num); i++)
    {
        if (old == num)
            com.clear();
        if (num % i == 0)
        {
            vector<int> a;
            a = com;
            a.push_back(i);
            finds(i, old, num / i, a, ans);
            a.push_back(num / i);
            ans.push_back(a);
        }
    }
}
vector<vector<int>> ans;
vector<int> zero;
dfs(2, num, num, zero, ans);
/*num 為 input 數字*/
for (int i = 0; i < ans.size(); i++)
{
    for (int j = 0; j < ans[i].size() - 1; j++)
        cout << ans[i][j] << " ";
    cout << ans[i][ans[i].size() - 1] << endl;
}

```

6.19 數字加法組合

```

void recur(int i, int n, int m, vector<int> &out, vector<vector<
int>> &ans)
{
    if (n == 0)
    {
        for (int i : out)
            if (i > m)
                return;
        ans.push_back(out);
    }
    for (int j = i; j <= n; j++)
    {
        out.push_back(j);
        recur(j, n - j, m, out, ans);
        out.pop_back();
    }
}
vector<vector<int>> ans;
vector<int> zero;
recur(1, num, num, zero, ans);
// num 為 input 數字
for (int i = 0; i < ans.size(); i++)
{
    for (int j = 0; j < ans[i].size() - 1; j++)
        cout << ans[i][j] << " ";
    cout << ans[i][ans[i].size() - 1] << endl;
}

```

6.20 羅馬數字

```

int romanToInt(string s)
{
    unordered_map<char, int> T;
    T['I'] = 1;
    T['V'] = 5;
    T['X'] = 10;
    T['L'] = 50;
    T['C'] = 100;
    T['D'] = 500;
    T['M'] = 1000;

    int sum = T[s.back()];
    for (int i = s.length() - 2; i >= 0; --i)
    {
        if (T[s[i]] < T[s[i + 1]])
            sum -= T[s[i]];
        else
            sum += T[s[i]];
    }
    return sum;
}

```

6.21 質因數分解

```

vector<int> primeFactorization(int tn) // 配合質數表
{
    vector<int> f;
    f.clear();
    int n = tn;
    for (int i = 0; i < (int)p.size(); ++i)
    {
        if (p[i] * p[i] > n)
            break;
        if (n % p[i])

```

```

            continue;
            // f.pb(p[i]); //不重複
            while (n % p[i] == 0)
            {
                n /= p[i];
                // f.pb(p[i]); //重複
            }
        }
        if (n != 1)
            f.pb(n);
        return f;
    }
}
vector<int> factorcount(int tn) // 質因數個數
{
    // ex tn = 2 * 2 * 3 => {2, 1}
    // "2" 個 2, "1" 個 3
    vector<int> fac;
    for (auto pr : p)
    {
        if (pr * pr > tn)
            break;
        if (tn % pr == 0)
        {
            int cc = 0;
            while (tn % pr == 0)
            {
                cc++;
                tn /= pr;
            }
            fac.push_back(cc);
        }
    }
    if (tn > 1)
        fac.push_back(1);
    return fac;
}

```

6.22 質數數量

```

// 10 ^ 11 左右
#define LL long long
const int N = 5e6 + 2;
bool np[N];
int prime[N], pi[N];
int getprime()
{
    int cnt = 0;
    np[0] = np[1] = true;
    pi[0] = pi[1] = 0;
    for (int i = 2; i < N; ++i)
    {
        if (!np[i])
            prime[++cnt] = i;
        pi[i] = cnt;
        for (int j = 1; j <= cnt && i * prime[j] < N; ++j)
        {
            np[i * prime[j]] = true;
            if (i % prime[j] == 0)
                break;
        }
    }
    return cnt;
}
const int M = 7;
const int PM = 2 * 3 * 5 * 7 * 11 * 13 * 17;
int phi[PM + 1][M + 1], sz[M + 1];
void init()
{
    getprime();
    sz[0] = 1;
    for (int i = 0; i <= PM; ++i)
        phi[i][0] = i;
    for (int i = 1; i <= M; ++i)
    {
        sz[i] = prime[i] * sz[i - 1];
        for (int j = 1; j <= PM; ++j)
            phi[j][i] = phi[j][i - 1] - phi[j / prime[i]][i - 1];
    }
}
int sqrt2(LL x)
{
    LL r = (LL)sqrt(x - 0.1);
    while (r * r <= x)
        ++r;
    return int(r - 1);
}
int sqrt3(LL x)
{
    LL r = (LL)cbrt(x - 0.1);
    while (r * r * r <= x)
        ++r;
    return int(r - 1);
}
LL getphi(LL x, int s)
{
    if (s == 0)

```

```

    return x;
if (s <= M)
    return phi[x % sz[s]][s] + (x / sz[s]) * phi[sz[s]][s];
if (x <= prime[s] * prime[s])
    return pi[x] - s + 1;
if (x <= prime[s] * prime[s] * prime[s] && x < N)
{
    int s2x = pi[sqrt2(x)];
    LL ans = pi[x] - (s2x + s - 2) * (s2x - s + 1) / 2;
    for (int i = s + 1; i <= s2x; ++i)
        ans += pi[x / prime[i]];
    return ans;
}
return getphi(x, s - 1) - getphi(x / prime[s], s - 1);
}
LL getpi(LL x)
{
    if (x < N)
        return pi[x];
    LL ans = getphi(x, pi[sqrt3(x)]) + pi[sqrt3(x)] - 1;
    for (int i = pi[sqrt3(x)] + 1, ed = pi[sqrt2(x)]; i <= ed; ++i)
        ans -= getpi(x / prime[i]) - i + 1;
    return ans;
}
LL lehmer_pi(LL x)
{
    if (x < N)
        return pi[x];
    int a = (int)lehmer_pi(sqrt2(sqrt2(x)));
    int b = (int)lehmer_pi(sqrt2(x));
    int c = (int)lehmer_pi(sqrt3(x));
    LL sum = getphi(x, a) + (LL)(b + a - 2) * (b - a + 1) / 2;
    for (int i = a + 1; i <= b; i++)
    {
        LL w = x / prime[i];
        sum -= lehmer_pi(w);
        if (i > c)
            continue;
        LL lim = lehmer_pi(sqrt2(w));
        for (int j = i; j <= lim; j++)
            sum -= lehmer_pi(w / prime[j]) - (j - 1);
    }
    return sum;
}
// lehmer_pi(n)

```

7 Other

7.1 Binary search 三類變化

```

// 查找和目標值完全相等的數
int find(vector<int> &nums, int target)
{
    int left = 0, right = nums.size() - 1;
    while (left < right)
    {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target)
            return mid;
        else if (nums[mid] < target)
            left = mid + 1;
        else
            right = mid;
    }
    return -1;
}
// 找第一個不小於目標值的數 == 找最後一個小於目標值的數
/*(lower_bound)*/
int find(vector<int> &nums, int target)
{
    int left = 0, right = nums.size() - 1;
    while (left < right)
    {
        int mid = left + (right - left) / 2;
        if (nums[mid] < target)
            left = mid + 1;
        else
            right = mid;
    }
    return right;
}
// 找第一個大於目標值的數 == 找最後一個不大於目標值的數
/*(upper_bound)*/
int find(vector<int> &nums, int target)
{
    int left = 0, right = nums.size() - 1;
    while (left < right)
    {
        int mid = left + (right - left) / 2;
        if (nums[mid] <= target)
            left = mid + 1;
        else
            right = mid;
    }
    return right;
}

```

```

}
return right;
}

```

7.2 Heap sort

```

void MaxHeapify(vector<int> &array, int root, int length)
{
    int left = 2 * root, right = 2 * root + 1, largest;
    if (left <= length && array[left] > array[root])
        largest = left;
    else
        largest = root;
    if (right <= length && array[right] > array[largest])
        largest = right;
    if (largest != root)
    {
        swap(array[largest], array[root]);
        MaxHeapify(array, largest, length);
    }
}
void HeapSort(vector<int> &array)
{
    array.insert(array.begin(), 0);
    for (int i = (int)array.size() / 2; i >= 1; i--)
        MaxHeapify(array, i, (int)array.size() - 1);
    int size = (int)array.size() - 1;
    for (int i = (int)array.size() - 1; i >= 2; i--)
    {
        swap(array[1], array[i]);
        size--;
        MaxHeapify(array, 1, size);
    }
    array.erase(array.begin());
}

```

7.3 Josephus

```

/*n people kill k for each turn*/
int josephus(int n, int k)
{
    int s = 0;
    for (int i = 2; i <= n; i++)
    {
        s = (s + k) % i;
    }
    /*index start from 1 -> s+1*/
    return s + 1;
}
/*died at kth*/
int kth(int n, int m, int k)
{
    if (m == 1)
        return n - 1;
    for (k = k * m + m - 1; k >= n; k = k - n + (k - n) / (m - 1))
        ;
    return k;
}

```

7.4 Largest Multi-interval

```

/*多區間算最大值*/
bool name(pii a, pii b)
{ return b.first > a.first; }
vector<pii> data;
data.pb(pii(a, c)); // 區間 a 到 c
sort(data.begin(), data.end(), name); // pair first 從小到大
int l = data[0].x, r = data[0].y, res = 0;
for (int i = 1; i < data.size(); i++)
{
    if (data[i].x <= r)
    {
        if (r < data[i].y)
            r = data[i].y;
    }
    else
    {
        res += r - l;
        l = data[i].x;
        r = data[i].y;
    }
}
res += r - l;

```

7.5 Merge sort

```

long long merge(vector<int> &arr, int left, int mid, int right)
{
    int *tmp = new int[right - left + 1];
    long long sum = 0;
    int l = left, r = mid + 1, m = 0;
    while (l <= mid && r <= right)
    {
        if (arr[l] <= arr[r])

```

```

        tmp[m++] = arr[l++];
    else
    {
        tmp[m++] = arr[r++];
        sum += mid - l + 1;
    }
}
while (l <= mid)
    tmp[m++] = arr[l++];
while (r <= right)
    tmp[m++] = arr[r++];
for (int i = left; i <= right; ++i)
    arr[i] = tmp[i - left];
delete[] tmp;
return sum;
}
long long mergesort(vector<int> &arr, int left, int right)
{
    long long sum = 0;
    // left = 0, right = P.size() - 1
    if (left < right)
    {
        int mid = (left + right) / 2;
        sum = mergesort(arr, left, mid);
        sum += mergesort(arr, mid + 1, right);
        sum += merge(arr, left, mid, right);
    }
    return sum; // 回傳為 swap 次數
}

```

7.6 N Queen problem

```

#define N 18
void printSolution(vector<vector<int>> &board)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            printf(" %d ", board[i][j]);
        printf("\n");
    }
}
bool isSafe(vector<vector<int>> &board, int row, int col)
{
    int i, j;
    /* Check this row on left side */
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;
    /* Check upper diagonal on left side */
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;
    /* Check lower diagonal on left side */
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;
    return true;
}
bool solveNQUtil(vector<vector<int>> &board, int col)
{
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++)
    {
        if (isSafe(board, i, col))
        {
            board[i][col] = 1;
            /* recur to place rest of the queens */
            if (solveNQUtil(board, col + 1))
                return true;
            board[i][col] = 0; // BACKTRACK
        }
    }
    return false;
}
bool solveNQ()
{
    vector<vector<int>> board(N, vector<int>(N, 0));
    if (solveNQUtil(board, 0) == false)
    {
        printf("Solution does not exist");
        return false;
    }
    printSolution(board);
    return true;
}

```

7.7 Quick sort

```

int Partition(vector<int> &arr, int front, int end)
{
    int pivot = arr[end];
    int i = front - 1;
    for (int j = front; j < end; j++)
    {

```

```

        if (arr[j] < pivot)
        {
            i++;
            swap(arr[i], arr[j]);
        }
        i++;
        swap(arr[i], arr[end]);
        return i;
    }
}
void QuickSort(vector<int> &arr, int front, int end)
{
    // front = 0, end = arr.size() - 1
    if (front < end)
    {
        int pivot = Partition(arr, front, end);
        QuickSort(arr, front, pivot - 1);
        QuickSort(arr, pivot + 1, end);
    }
}

```

7.8 Sudoku solution

```

/*數獨解法*/
int getSquareIndex(int row, int column, int n)
{
    return row / n * n + column / n;
}

bool backtracking(vector<vector<int>> &board, vector<vector<
    bool>> &rows, vector<vector<bool>> &cols,
    vector<vector<bool>> &boxes, int index, int n)
{
    int n2 = n * n;
    int rowNum = index / n2, colNum = index % n2;
    if (index >= n2 * n2)
        return true;

    if (board[rowNum][colNum] != 0)
        return backtracking(board, rows, cols, boxes, index + 1,
            n);

    for (int i = 1; i <= n2; i++)
    {
        if (!rows[rowNum][i] && !cols[colNum][i] && !boxes[
            getSquareIndex(rowNum, colNum, n)][i])
        {
            rows[rowNum][i] = true;
            cols[colNum][i] = true;
            boxes[getSquareIndex(rowNum, colNum, n)][i] = true;
            board[rowNum][colNum] = i;
            if (backtracking(board, rows, cols, boxes, index +
                1, n))
                return true;
            board[rowNum][colNum] = 0;
            rows[rowNum][i] = false;
            cols[colNum][i] = false;
            boxes[getSquareIndex(rowNum, colNum, n)][i] = false;
        }
    }
    return false;
}

/*用法 main*/
int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
vector<vector<int>> board(n * n + 1, vector<int>(n * n + 1, 0))
;
vector<vector<bool>> isRow(n * n + 1, vector<bool>(n * n + 1,
    false));
vector<vector<bool>> isColumn(n * n + 1, vector<bool>(n * n +
    1, false));
vector<vector<bool>> isSquare(n * n + 1, vector<bool>(n * n +
    1, false));

for (int i = 0; i < n * n; ++i)
{
    for (int j = 0; j < n * n; ++j)
    {
        int number;
        cin >> number;
        board[i][j] = number;
        if (number == 0)
            continue;
        isRow[i][number] = true;
        isColumn[j][number] = true;
        isSquare[getSquareIndex(i, j, n)][number] = true;
    }
}
if (backtracking(board, isRow, isColumn, isSquare, 0, n))
    /*有解答*/
else
    /*無解答*/

```

7.9 Weighted Job Scheduling

```

struct Job

```

```

{
    int start, finish, profit;
};
bool jobComparataor(Job s1, Job s2)
{
    return (s1.finish < s2.finish);
}
int latestNonConflict(Job arr[], int i)
{
    for (int j = i - 1; j >= 0; j--)
    {
        if (arr[j].finish <= arr[i].start)
            return j;
    }
    return -1;
}
int findMaxProfit(Job arr[], int n)
{
    sort(arr, arr + n, jobComparataor);
    int *table = new int[n];
    table[0] = arr[0].profit;
    for (int i = 1; i < n; i++)
    {
        int inclProf = arr[i].profit;
        int l = latestNonConflict(arr, i);
        if (l != -1)
            inclProf += table[l];
        table[i] = max(inclProf, table[i - 1]);
    }
    int result = table[n - 1];
    delete[] table;

    return result;
}

```

8 String

8.1 KMP

```

// 用在在一個 S 內查找一個詞 W 的出現位置
void ComputePrefix(string s, int next[])
{
    int n = s.length();
    int q, k;
    next[0] = 0;
    for (k = 0, q = 1; q < n; q++)
    {
        while (k > 0 && s[k] != s[q])
            k = next[k];
        if (s[k] == s[q])
            k++;
        next[q] = k;
    }
}
void KMPMatcher(string text, string pattern)
{
    int n = text.length();
    int m = pattern.length();
    int next[pattern.length()];
    ComputePrefix(pattern, next);

    for (int i = 0, q = 0; i < n; i++)
    {
        while (q > 0 && pattern[q] != text[i])
            q = next[q];
        if (pattern[q] == text[i])
            q++;
        if (q == m)
        {
            cout << "Pattern occurs with shift " << i - m + 1
                  << endl;
            q = 0;
        }
    }
}
// string s = "abcdabcdebcd";
// string p = "bcd";
// KMPMatcher(s, p);
// cout << endl;

```

8.2 Min Edit Distance

```

int EditDistance(string a, string b)
{
    vector<vector<int>> dp(a.size() + 1, vector<int>(b.size() + 1, 0));
    int m = a.length(), n = b.length();
    for (int i = 0; i < m + 1; i++)
    {
        for (int j = 0; j < n + 1; j++)
        {
            if (i == 0)
                dp[i][j] = j;
            else if (j == 0)

```

```

                dp[i][j] = i;
            else if (a[i - 1] == b[j - 1])
                dp[i][j] = dp[i - 1][j - 1];
            else
                dp[i][j] = 1 + min(min(dp[i - 1][j], dp[i][j - 1]), dp[i - 1][j - 1]);
        }
    }
    return dp[m][n];
}

```

8.3 Sliding window

```

string minWindow(string s, string t)
{
    unordered_map<char, int> letterCnt;
    for (int i = 0; i < t.length(); i++)
        letterCnt[t[i]]++;
    int minLength = INT_MAX, minStart = -1;
    int left = 0, matchCnt = 0;
    for (int i = 0; i < s.length(); i++)
    {
        if (--letterCnt[s[i]] >= 0)
            matchCnt++;
        while (matchCnt == t.length())
        {
            if (i - left + 1 < minLength)
            {
                minLength = i - left + 1;
                minStart = left;
            }
            if (++letterCnt[s[left]] > 0)
                matchCnt--;
            left++;
        }
    }
    return minLength == INT_MAX ? "" : s.substr(minStart,
        minLength);
}

```

8.4 Split

```

vector<string> mysplit(const string& str, const string& delim)
{
    vector<string> res;
    if (" " == str)
        return res;

    char *strs = new char[str.length() + 1];
    strcpy(strs, str.c_str());

    char *d = new char[delim.length() + 1];
    strcpy(d, delim.c_str());

    char *p = strtok(strs, d);
    while (p)
    {
        string s = p;
        res.push_back(s);
        p = strtok(NULL, d);
    }
    return res;
}

```

9 data structure

9.1 Bigint

```

//台大 //非必要請用python
struct Bigint
{
    static const int LEN = 60; // maxLEN
    static const int BIGMOD = 10000; //10為正常位數
    int s;
    int vl, v[LEN];
    // vector<int> v;
    Bigint() : s(1) { vl = 0; }
    Bigint(long long a)
    {
        s = 1;
        vl = 0;
        if (a < 0)
        {
            s = -1;
            a = -a;
        }
        while (a)
        {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    Bigint(string str)

```



```

{
    s = 1;
    v1 = 0;
    int stPos = 0, num = 0;
    if (!str.empty() && str[0] == '-')
    {
        stPos = 1;
        s = -1;
    }
    for (int i = str.length() - 1, q = 1; i >= stPos; i--)
    {
        num += (str[i] - '0') * q;
        if ((q *= 10) >= BIGMOD)
        {
            push_back(num);
            num = 0;
            q = 1;
        }
    }
    if (num)
        push_back(num);
    n();
}
int len() const
{
    return v1; //return SZ(v);
}
bool empty() const { return len() == 0; }
void push_back(int x)
{
    v[v1++] = x; //v.PB(x);
}
void pop_back()
{
    v1--; //v.pop_back();
}
int back() const
{
    return v[v1 - 1]; //return v.back();
}
void n()
{
    while (!empty() && !back())
        pop_back();
}
void resize(int nl)
{
    v1 = nl; //v.resize(nl);
    fill(v, v + v1, 0); //fill(ALL(v), 0);
}
void print() const
{
    if (empty())
    {
        putchar('0');
        return;
    }
    if (s == -1)
        putchar('-');
    printf("%d", back());
    for (int i = len() - 2; i >= 0; i--)
        printf("%.4d", v[i]);
}
friend std::ostream &operator<<(std::ostream &out, const
    Bigint &a)
{
    if (a.empty())
    {
        out << "0";
        return out;
    }
    if (a.s == -1)
        out << "-";
    out << a.back();
    for (int i = a.len() - 2; i >= 0; i--)
    {
        char str[10];
        sprintf(str, 5, "%.4d", a.v[i]);
        out << str;
    }
    return out;
}
int cp3(const Bigint &b) const
{
    if (s != b.s)
        return s - b.s;
    if (s == -1)
        return -(*this).cp3(-b);
    if (len() != b.len())
        return len() - b.len(); //int
    for (int i = len() - 1; i >= 0; i--)
        if (v[i] != b.v[i])
            return v[i] - b.v[i];
    return 0;
}
bool operator<(const Bigint &b) const
{

```

```

        return cp3(b) < 0;
    }
    bool operator<=(const Bigint &b) const
    {
        return cp3(b) <= 0;
    }
    bool operator==(const Bigint &b) const
    {
        return cp3(b) == 0;
    }
    bool operator!=(const Bigint &b) const
    {
        return cp3(b) != 0;
    }
    bool operator>(const Bigint &b) const
    {
        return cp3(b) > 0;
    }
    bool operator>=(const Bigint &b) const
    {
        return cp3(b) >= 0;
    }
}
Bigint operator-() const
{
    Bigint r = (*this);
    r.s = -r.s;
    return r;
}
Bigint operator+(const Bigint &b) const
{
    if (s == -1)
        return -(*this) + (-b);
    if (b.s == -1)
        return (*this) - (-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i = 0; i < nl; i++)
    {
        if (i < len())
            r.v[i] += v[i];
        if (i < b.len())
            r.v[i] += b.v[i];
        if (r.v[i] >= BIGMOD)
        {
            r.v[i + 1] += r.v[i] / BIGMOD;
            r.v[i] %= BIGMOD;
        }
    }
    r.n();
    return r;
}
Bigint operator-(const Bigint &b) const
{
    if (s == -1)
        return -(*this) - (-b);
    if (b.s == -1)
        return (*this) + (-b);
    if ((*this) < b)
        return -(b - (*this));
    Bigint r;
    r.resize(len());
    for (int i = 0; i < len(); i++)
    {
        r.v[i] += v[i];
        if (i < b.len())
            r.v[i] -= b.v[i];
        if (r.v[i] < 0)
        {
            r.v[i] += BIGMOD;
            r.v[i + 1]--;
        }
    }
    r.n();
    return r;
}
Bigint operator*(const Bigint &b)
{
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i = 0; i < len(); i++)
    {
        for (int j = 0; j < b.len(); j++)
        {
            r.v[i + j] += v[i] * b.v[j];
            if (r.v[i + j] >= BIGMOD)
            {
                r.v[i + j + 1] += r.v[i + j] / BIGMOD;
                r.v[i + j] %= BIGMOD;
            }
        }
    }
    r.n();
    return r;
}
Bigint operator/(const Bigint &b)

```

```

{
    Bigint r;
    r.resize(max(1, len() - b.len() + 1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i = r.len() - 1; i >= 0; i--)
    {
        int d = 0, u = BIGMOD - 1;
        while (d < u)
        {
            int m = (d + u + 1) >> 1;
            r.v[i] = m;
            if ((r * b2) > (*this))
                u = m - 1;
            else
                d = m;
        }
        r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
}
Bigint operator%(const Bigint &b)
{
    return (*this) - (*this) / b * b;
}
};

```

9.2 DisjointSet

```

struct DisjointSet {
    int p[maxn], sz[maxn], n, cc;
    vector<pair<int*, int>> his;
    vector<int> sh;
    void init(int _n) {
        n = _n; cc = n;
        for (int i = 0; i < n; ++i) sz[i] = 1, p[i] = i;
        sh.clear(); his.clear();
    }
    void assign(int *k, int v) {
        his.emplace_back(k, *k);
        *k = v;
    }
    void save() {
        sh.push_back((int)his.size());
    }
    void undo() {
        int last = sh.back(); sh.pop_back();
        while (his.size() != last) {
            int *k, v;
            tie(k, v) = his.back(); his.pop_back();
            *k = v;
        }
    }
    int find(int x) {
        if (x == p[x]) return x;
        return find(p[x]);
    }
    void merge(int x, int y) {
        x = find(x); y = find(y);
        if (x == y) return;
        if (sz[x] > sz[y]) swap(x, y);
        assign(&sz[y], sz[x] + sz[y]);
        assign(&p[x], y);
        assign(&cc, cc - 1);
    }
}
};

```

9.3 Matirx

```

template <typename T>
struct Matrix
{
    using rt = std::vector<T>;
    using mt = std::vector<rt>;
    using matrix = Matrix<T>;
    int r, c; // [r][c]
    mt m;
    Matrix(int r, int c) : r(r), c(c), m(r, rt(c)) {}
    Matrix(mt a) { m = a; r = a.size(), c = a[0].size(); }
    rt &operator[](int i) { return m[i]; }
    matrix operator+(const matrix &a)
    {
        matrix rev(r, c);
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                rev[i][j] = m[i][j] + a.m[i][j];
        return rev;
    }
    matrix operator-(const matrix &a)
    {
        matrix rev(r, c);
        for (int i = 0; i < r; ++i)

```

```

        for (int j = 0; j < c; ++j)
            rev[i][j] = m[i][j] - a.m[i][j];
        return rev;
    }
    matrix operator*(const matrix &a)
    {
        matrix rev(r, a.c);
        matrix tmp(a.c, a.r);
        for (int i = 0; i < a.r; ++i)
            for (int j = 0; j < a.c; ++j)
                tmp[j][i] = a.m[i][j];
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < a.c; ++j)
                for (int k = 0; k < c; ++k)
                    rev.m[i][j] += m[i][k] * tmp[j][k];
        return rev;
    }
    bool inverse() //逆矩陣判斷
    {
        Matrix t(r, r + c);
        for (int y = 0; y < r; y++)
        {
            t.m[y][c + y] = 1;
            for (int x = 0; x < c; ++x)
                t.m[y][x] = m[y][x];
        }
        if (!t.gas())
            return false;
        for (int y = 0; y < r; y++)
            for (int x = 0; x < c; ++x)
                m[y][x] = t.m[y][c + x] / t.m[y][y];
        return true;
    }
    T gas() //行列式
    {
        vector<T> lazy(r, 1);
        bool sign = false;
        for (int i = 0; i < r; ++i)
        {
            if (m[i][i] == 0)
            {
                int j = i + 1;
                while (j < r && !m[j][i])
                    j++;
                if (j == r)
                    continue;
                m[i].swap(m[j]);
                sign = !sign;
            }
            for (int j = 0; j < r; ++j)
            {
                if (i == j)
                    continue;
                lazy[j] = lazy[j] * m[i][i];
                T mx = m[j][i];
                for (int k = 0; k < c; ++k)
                    m[j][k] = m[j][k] * m[i][i] - m[i][k] * mx;
            }
        }
        T det = sign ? -1 : 1;
        for (int i = 0; i < r; ++i)
        {
            det = det * m[i][i];
            det = det / lazy[i];
            for (auto &j : m[i])
                j /= lazy[i];
        }
        return det;
    }
};

```

9.4 分數

```

typedef long long ll;
struct fraction
{
    ll n, d;
    fraction(const ll &n = 0, const ll &d = 1) : n(_n), d(_d)
    {
        ll t = __gcd(n, d);
        n /= t, d /= t;
        if (d < 0)
            n = -n, d = -d;
    }
    fraction operator-() const
    {
        return fraction(-n, d);
    }
    fraction operator+(const fraction &b) const
    {
        return fraction(n * b.d + b.n * d, d * b.d);
    }
    fraction operator-(const fraction &b) const
    {
        return fraction(n * b.d - b.n * d, d * b.d);
    }
}

```

```
fraction operator*(const fraction &b) const
{
    return fraction(n * b.n, d * b.d);
}
fraction operator/(const fraction &b) const
{
    return fraction(n * b.d, d * b.n);
}
void print()
{
    cout << n;
    if (d != 1)
        cout << "/" << d;
}
};
```