

1 Basic

1.1 A function

```
1 round(double f);           // 四捨五入
2 ceil(double f);           // 無條件捨去
3 floor(double f);          // 無條件進入
4 __builtin_popcount(int n); // 32bit有多少 1
5 to_string(int s);          // int to string
6 /** 全排列要先 sort !!! */
7 next_permutation(num.begin(), num.end());
8 prev_permutation(num.begin(), num.end());
9 //用binary search找大於或等於val的最小值的位置
10 vector<int>::iterator it = lower_bound(v.
    begin(), v.end(), val);
11 //用binary search找大於val的最小值的位置
12 vector<int>::iterator it = upper_bound(v.
    begin(), v.end(), val);
13 /**queue*/
14 queue<datatype> q;
15 front(); /*取出最前面的值(沒有移除掉喔!!)*/
16 back(); /*取出最後面的值(沒有移除掉!!)*/
17 pop(); /*移除最前面的值*/
18 push(); /*新增值到最後面*/
19 empty(); /*回傳bool,檢查是不是空的queue*/
20 size(); /*queue 的大小*/
21
22 /**stack*/
23 stack<datatype> s;
24 top(); /*取出最上面的值(沒有移除掉喔!!)*/
25 pop(); /*移除最上面的值*/
26 push(); /*新增值到最上面*/
27 empty(); /*回傳bool,檢查是不是空的stack*/
28 size(); /*stack 的大小*/
29
30 /**unordered_set*/
31 unordered_set<datatype> s;
32 unordered_set<datatype> s(arr, arr + n);
33 /**initial with array*/
34 insert(); /*插入值*/
35 erase(); /*刪除值*/
36 empty(); /*bool 檢查是不是空*/
37 count(); /*判斷元素存在回傳1 無則回傳0*/
```

1.2 Codeblock setting

```
1 Settings -> Editor -> Keyboard shortcuts ->
    Plugins -> Source code formatter (AStyle
    )
2 Settings -> Source Formatter -> Padding
3 Delete empty lines within a function or
    method
4 Insert space padding around operators
5 Insert space padding around parentheses on
    outside
```

```
6 Remove extra space padding around
    parentheses
```

1.3 data range

```
1 int (-2147483648 to 2147483647)
2 unsigned int(0 to 4294967295)
3 long(-2147483648 to 2147483647)
4 unsigned long(0 to 4294967295)
5 long long(-9223372036854775808 to
    9223372036854775807)
6 unsigned long long (0 to
    18446744073709551615)
```

1.4 IO_fast

```
1 void io()
2 {
3     ios::sync_with_stdio(false);
4     cin.tie(nullptr);
5     cout.tie(nullptr);
6 }
```

1.5 Python

```
1 /*輸入1*/
2 import sys
3 line = sys.stdin.readline() // 會讀到換行
4 D, R, N = map(int, line[:-1].split()) // 分
    三個 int 變數
5 pow(a, b, c) // a ^ b % c
6 print(*objects, sep = ' ', end = '\n')
7 objects -- /*可以一次輸出多個對象*/
8 sep -- /*分開多個objects*/
9 end -- /*默認值是\n*/
```

2 DP

2.1 3 維 DP 思路

```
1 解題思路: dp[i][j][k]
2 i 跟 j 代表 range i ~ j 的 value
3 k 在我的理解裡是視題目的要求而定的
4 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
5 所以每次區間消去的值就是(k+1) * (k+1)
6 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

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N], number[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     for (int i = 0; i < n; ++i)
7     {
8         int num = min(number[i], w / weight[
9             i]);
10        for (int k = 1; num > 0; k *= 2)
11        {
12            if (k > num)
13                k = num;
14            num -= k;
15            for (int j = w; j >= weight[i] *
16                k; --j)
17                c[j] = max(c[j], c[j -
18                    weight[i] * k] + cost[i]
19                    * k);
20        }
21    }
22    cout << "Max Prince" << c[w];
23 }
```

2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
    value, int bag_Weight)
2 {
3     // vector<int> weight = {1, 3, 4};
4     // vector<int> value = {15, 20, 30};
5     // int bagWeight = 4;
6     vector<vector<int>> dp(weight.size(),
7         vector<int>(bagWeight + 1, 0));
8     for (int j = weight[0]; j <= bagWeight;
9         j++)
10         dp[0][j] = value[0];
11     // weight數組的大小就是物品個數
12     for (int i = 1; i < weight.size(); i++)
13     { // 遍歷物品
14         for (int j = 0; j <= bagWeight; j++)
15         { // 遍歷背包容量
16             if (j < weight[i]) dp[i][j] = dp
17                 [i - 1][j];
18             else dp[i][j] = max(dp[i - 1][j
19                 ], dp[i - 1][j - weight[i]]
20                 + value[i]);
21         }
22     }
23     cout << dp[weight.size() - 1][bagWeight]
24         << endl;
25 }
```

2.4 Knapsack Unbounded

```
1 const int N = 100, W = 100000;
2 int cost[N], weight[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     memset(c, 0, sizeof(c));
7     for (int i = 0; i < n; ++i)
8         for (int j = weight[i]; j <= w; ++j)
9             c[j] = max(c[j], c[j - weight[i]
10                 ] + cost[i]);
11    cout << "最高的價值為" << c[w];
12 }
```

2.5 LCIS

```
1 int LCIS_len(vector<int> arr1, vector<int>
    arr2)
2 {
3     int n = arr1.size(), m = arr2.size();
4     vector<int> table(m, 0);
5     for (int j = 0; j < m; j++)
6         table[j] = 0;
7     for (int i = 0; i < n; i++)
8     {
9         int current = 0;
10        for (int j = 0; j < m; j++)
11        {
12            if (arr1[i] == arr2[j])
13                if (current + 1 > table[j])
14                    table[j] = current + 1;
15
16            if (arr1[i] > arr2[j])
17                if (table[j] > current)
18                    current = table[j];
19        }
20    }
21    int result = 0;
22    for (int i = 0; i < m; i++)
23        if (table[i] > result)
24            result = table[i];
25    return result;
26 }
```

2.6 LCS

```
1 int LCS(vector<string> Ans, vector<string>
    num)
2 {
3     int N = Ans.size(), M = num.size();
4     vector<vector<int>> LCS(N + 1, vector<
5         int>(M + 1, 0));
6     for (int i = 1; i <= N; ++i)
7     {
8         for (int j = 1; j <= M; ++j)
9         {
10            if (Ans[i - 1] == num[j - 1])
11                LCS[i][j] = LCS[i - 1][j -
12                    1] + 1;
13            else
14                LCS[i][j] = max(LCS[i - 1][j],
15                    LCS[i][j - 1]);
16        }
17    }
18    return LCS[N][M];
19 }
```

```

11     else
12         LCS[i][j] = max(LCS[i - 1][j]
13             , LCS[i][j - 1]);
14     }
15     cout << LCS[N][M] << '\n';
16     //列印 LCS
17     int n = N, m = M;
18     vector<string> k;
19     while (n && m)
20     {
21         if (LCS[n][m] != max(LCS[n - 1][m],
22             LCS[n][m - 1]))
23         {
24             k.push_back(Ans[n - 1]);
25             n--;
26             m--;
27         }
28         else if (LCS[n][m] == LCS[n - 1][m])
29             n--;
30         else if (LCS[n][m] == LCS[n][m - 1])
31             m--;
32     }
33     reverse(k.begin(), k.end());
34     for (auto i : k)
35         cout << i << " ";
36     cout << endl;
37     return LCS[N][M];
38 }

```

2.7 LIS

```

1 void getMaxElementAndPos(vector<int> &LISTbl,
2     vector<int> &LISLen, int tNum, int
3     tlen, int tStart, int &num, int &pos)
4 {
5     int max = numeric_limits<int>::min();
6     int maxPos;
7     for (int i = tStart; i >= 0; i--)
8     {
9         if (LISLen[i] == tlen && LISTbl[i] <
10             tNum)
11         {
12             if (LISTbl[i] > max)
13             {
14                 max = LISTbl[i];
15                 maxPos = i;
16             }
17         }
18     }
19     num = max;
20     pos = maxPos;
21 }
22 int LIS(vector<int> &LISTbl)
23 {
24     if (LISTbl.size() == 0)
25         return 0;
26     vector<int> LISLen(LISTbl.size(), 1);
27     for (int i = 1; i < LISTbl.size(); i++)
28         for (int j = 0; j < i; j++)
29             if (LISTbl[j] < LISTbl[i])
30                 LISLen[i] = max(LISLen[i],
31                     LISLen[j] + 1);
32 }

```

2.8 LPS

```

1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s.size();
5     for (int i = 0; i < n; i++)
6     {
7         int x = 0;
8         while ((s[i - x] == s[i + x]) && (i -
9             x >= 0) && (i + x < n)) //odd
10             x++;
11         x--;
12         if (2 * x + 1 > maxlen)
13         {
14             maxlen = 2 * x + 1;
15             l = i - x;
16             r = i + x;
17         }
18         x = 0;
19         while ((s[i - x] == s[i + 1 + x]) &&
20             (i - x >= 0) && (i + 1 + x < n)) //even length
21             x++;
22         if (2 * x > maxlen)
23         {
24             maxlen = 2 * x;
25             l = i - x + 1;
26             r = i + x;
27         }
28     }
29     cout << maxlen << '\n'; // 最後長度
30 }

```

```

28 int maxlen = *max_element(LISLen.begin()
29     , LISLen.end());
30 int num, pos;
31 vector<int> buf;
32 getMaxElementAndPos(LISTbl, LISLen,
33     numeric_limits<int>::max(), maxlen,
34     LISTbl.size() - 1, num, pos);
35 buf.push_back(num);
36 for (int len = maxlen - 1; len >= 1; len
37     --)
38 {
39     int tnum = num;
40     int tpos = pos;
41     getMaxElementAndPos(LISTbl, LISLen,
42         tnum, len, tpos - 1, num, pos);
43     buf.push_back(tnum);
44 }
45 reverse(buf.begin(), buf.end());
46 for (int k = 0; k < buf.size(); k++) //
47     列印
48 {
49     if (k == buf.size() - 1)
50         cout << buf[k] << endl;
51     else
52         cout << buf[k] << ", ";
53 }
54 return maxlen;
55 }

```

```

28     cout << l + 1 << ' ' << r + 1 << '\n';
29     //頭到尾
30 }

```

2.9 Max_subarray

```

1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int> & nums) {
3     int local_max = nums[0], global_max =
4     nums[0];
5     for (int i = 1; i < nums.size(); i++) {
6         local_max = max(nums[i], local_max +
7             nums[i]);
8         global_max = max(local_max,
9             global_max);
10    }
11    return global_max;
12 }

```

2.10 Money problem

```

1 //能否湊得某個價位
2 void change(vector<int> price, int limit)
3 {
4     vector<bool> c(limit + 1, 0);
5     c[0] = true;
6     for (int i = 0; i < price.size(); ++i)
7         // 依序加入各種面額
8         for (int j = price[i]; j <= limit;
9             ++j) // 由低價位逐步到高價位
10             c[j] = c[j] | c[j - price[i]];
11         // 湊、湊、湊
12         if (c[limit]) cout << "YES\n";
13         else cout << "NO\n";
14 }
15 // 湊得某個價位的湊法總共幾種
16 void change(vector<int> price, int limit)
17 {
18     vector<int> c(limit + 1, 0);
19     c[0] = true;
20     for (int i = 0; i < price.size(); ++i)
21         for (int j = price[i]; j <= limit;
22             ++j)
23             c[j] += c[j - price[i]];
24     cout << c[limit] << '\n';
25 }
26 // 湊得某個價位的最少錢幣用量
27 void add_node() {
28     return n++;
29 }
30 void add_edge(int u, int v, long long
31     cap) {
32     edges.push_back( {u, v, cap, cap} );
33     edges.push_back( {v, u, 0, 0LL} );
34     m = edges.size();
35     G[u].push_back(m-2);
36     G[v].push_back(m-1);
37 }

```

```

32 //湊得某個價位的錢幣用量，有哪幾種可能性
33 void change(vector<int> price, int limit)
34 {
35     vector<int> c(limit + 1, 0);
36     c[0] = true;
37     for (int i = 0; i < price.size(); ++i)
38         for (int j = price[i]; j <= limit;
39             ++j)
40             c[j] |= c[j - price[i]] << 1; //
41             錢幣數量加一，每一種可能性都
42             加一。
43     for (int i = 1; i <= 63; ++i)
44         if (c[i] & (1 << i))
45             cout << "用" << i << "個錢幣可湊
46                 得價位" << m;
47 }

```

3 Flow & matching

3.1 Dinic

```

1 const long long INF = 1LL<<60;
2 struct Dinic { //O(VVE), with minimum cut
3     static const int MAXN = 5003;
4     struct Edge {
5         int u, v;
6         long long cap, rest;
7     };
8     int n, m, s, t, d[MAXN], cur[MAXN];
9     vector<Edge> edges;
10    vector<int> G[MAXN];
11    void init() {
12        edges.clear();
13        for (int i = 0; i < n; i++) G[i].clear();
14        n = 0;
15    }
16    // min cut start
17    bool side[MAXN];
18    void cut(int u) {
19        side[u] = 1;
20        for (int i : G[u]) {
21            if (!side[edges[i].v] &&
22                edges[i].rest > 0)
23                cut(edges[i].v);
24        }
25    }
26    // min cut end
27    int add_node() {
28        return n++;
29    }
30    void add_edge(int u, int v, long long
31        cap) {
32        edges.push_back( {u, v, cap, cap} );
33        edges.push_back( {v, u, 0, 0LL} );
34        m = edges.size();
35        G[u].push_back(m-2);
36        G[v].push_back(m-1);
37    }
38 }

```

```

36 bool bfs(){
37     fill(d,d+n,-1);
38     queue<int> que;
39     que.push(s); d[s]=0;
40     while (!que.empty()){
41         int u = que.front(); que.pop();
42         for (int ei : G[u]){
43             Edge &e = edges[ei];
44             if (d[e.v] < 0 && e.rest > 0){
45                 d[e.v] = d[u] + 1;
46                 que.push(e.v);
47             }
48         }
49     }
50     return d[t] >= 0;
51 }
52 long long dfs(int u, long long a){
53     if (u == t || a == 0) return a;
54     long long flow = 0, f;
55     for (int &i=cur[u]; i < (int)G[u].size(); i++) {
56         Edge &e = edges[ G[u][i] ];
57         if ( d[u] + 1 != d[e.v] ) continue;
58         f = dfs(e.v, min(a, e.rest) );
59         if ( f > 0 ) {
60             e.rest -= f;
61             edges[ G[u][i]^1 ].rest += f;
62             flow += f;
63             a -= f;
64             if ( a == 0 ) break;
65         }
66     }
67     return flow;
68 }
69 long long maxflow(int _s, int _t){
70     s = _s, t = _t;
71     long long flow = 0, mf;
72     while ( bfs() ){
73         fill(cur,cur+n,0);
74         while ( (mf = dfs(s, INF)) )
75             flow += mf;
76     }
77     return flow;
78 } dinic;

```

3.2 Edmonds_karp

```

1 /*Flow - Edmonds-karp*/
2 /*Based on UVa820*/
3 #define inf 1000000
4 int getMaxFlow(vector<vector<int>> &capacity,
5     int s, int t, int n){
6     int ans = 0;
7     vector<vector<int>> residual(n+1, vector<int>(n+1, 0)); //residual network
8     while(true){
9         vector<int> bottleneck(n+1, 0);
10        bottleneck[s] = inf;
11        queue<int> q;
12        q.push(s);

```

```

12 vector<int> pre(n+1, 0);
13 while(!q.empty() && bottleneck[t] == 0){
14     int cur = q.front();
15     q.pop();
16     for(int i = 1; i <= n ; i++){
17         if(bottleneck[i] == 0 && capacity[
18             cur][i] > residual[cur][i]){
19             q.push(i);
20             pre[i] = cur;
21             bottleneck[i] = min(bottleneck[cur], capacity[cur][i] - residual[cur][i]);
22         }
23     }
24     if(bottleneck[t] == 0) break;
25     for(int cur = t; cur != s; cur = pre[cur]){
26         residual[pre[cur]][cur] += bottleneck[t];
27         residual[cur][pre[cur]] -= bottleneck[t];
28     }
29     ans += bottleneck[t];
30 }
31 return ans;
32 }
33 int main(){
34     int testcase = 1;
35     int n;
36     while(cin>>n){
37         if(n == 0) break;
38         vector<vector<int>> capacity(n+1, vector<int>(n+1, 0));
39         int s, t, c;
40         cin >> s >> t >> c;
41         int a, b, bandwidth;
42         for(int i = 0 ; i < c ; ++i){
43             cin >> a >> b >> bandwidth;
44             capacity[a][b] += bandwidth;
45             capacity[b][a] += bandwidth;
46         }
47         cout << "Network " << testcase++ << endl;
48         ;
49         cout << "The bandwidth is " << getMaxFlow(capacity, s, t, n) << "." << endl;
50         cout << endl;
51     }
52     return 0;
53 }

```

3.3 hungarian

```

1 /*bipartite - hungarian*/
2 struct Graph{
3     static const int MAXN = 5003;
4     vector<int> G[MAXN];
5     int n, match[MAXN], vis[MAXN];
6     void init(int _n){
7         n = _n;

```

```

8         for (int i=0; i<n; i++) G[i].clear();
9     }
10    bool dfs(int u){
11        for (int v:G[u]){
12            if (vis[v]) continue;
13            vis[v]=true;
14            if (match[v]==-1 || dfs(match[v])){
15                match[v] = u;
16                match[u] = v;
17                return true;
18            }
19        }
20        return false;
21    }
22    int solve(){
23        int res = 0;
24        memset(match,-1,sizeof(match));
25        for (int i=0; i<n; i++){
26            if (match[i]==-1){
27                memset(vis,0,sizeof(vis));
28                if ( dfs(i) ) res++;
29            }
30        }
31        return res;
32    }
33 } graph;

```

3.4 Maximum_matching

```

1 /*bipartite - maximum matching*/
2 bool dfs(vector<vector<bool>> res,int node,
3     vector<int>& x, vector<int>& y, vector<bool> pass){
4     for (int i = 0; i < res[0].size(); i++){
5         if(res[node][i] && !pass[i]){
6             pass[i] = true;
7             if(y[i] == -1 || dfs(res,y[i],x,
8                 y,pass)){
9                 x[node] = i;
10                y[i] = node;
11                return true;
12            }
13        }
14    }
15    return false;
16 }
17 int main(){
18     int n,m,l;
19     while(cin>>n>>m>>l){
20         vector<vector<bool>> res(n, vector<bool>(m, false));
21         for (int i = 0; i < l; i++){
22             int a, b;
23             cin >> a >> b;
24             res[a][b] = true;
25         }
26         int ans = 0;
27         vector<int> x(n, -1);
28         vector<int> y(m, -1);
29         for (int i = 0; i < n; i++){
30             vector<bool> pass(m, false);

```

```

29         if(dfs(res,i,x,y,pass))
30             ans += 1;
31     }
32     cout << ans << endl;
33     return 0;
34 }
35 /*
36 input:
37 4 3 5 //n matching m, l links
38 0 0
39 0 2
40 1 0
41 2 1
42 3 1
43 answer is 3
44 */

```

3.5 MFlow Model

```

1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 1000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];
13
14    void Clear()
15    {
16        tot = 0;
17        tim = 0;
18        for (int i = 1; i <= n; ++i)
19            first[i] = -1;
20    }
21    void Add(int from, int to, ll cp, ll flw)
22    {
23        u[tot] = from;
24        v[tot] = to;
25        cap[tot] = cp;
26        flow[tot] = flw;
27        next[tot] = first[u[tot]];
28        first[u[tot]] = tot;
29        ++tot;
30    }
31    bool bfs()
32    {
33        ++tim;
34        dis[s] = 0;
35        vi[s] = tim;
36
37        int head, tail;
38        head = tail = 1;
39        que[head] = s;
40        while (head <= tail)
41        {

```

```

42     for (int i = first[que[head]]; i
43         != -1; i = next[i])
44     {
45         if (vi[v[i]] != tim && cap[i]
46             > flow[i])
47         {
48             vi[v[i]] = tim;
49             dis[v[i]] = dis[que[head]] + 1;
50             que[++tail] = v[i];
51         }
52     }
53     ++head;
54     return vi[t] == tim;
55 }
56 ll dfs(int x, ll a)
57 {
58     if (x == t || a == 0)
59         return a;
60     ll flw = 0, f;
61     int &i = cur[x];
62     for (i = first[x]; i != -1; i = next[i])
63     {
64         if (dis[x] + 1 == dis[v[i]] && (
65             f = dfs(v[i], min(a, cap[i] - flow[i])) > 0)
66         )
67         {
68             flow[i] += f;
69             flow[i ^ 1] -= f;
70             a -= f;
71             flw += f;
72             if (a == 0)
73                 break;
74         }
75     }
76     return flw;
77 }
78 ll MaxFlow(int s, int t)
79 {
80     this->s = s;
81     this->t = t;
82     ll flw = 0;
83     while (bfs())
84     {
85         for (int i = 1; i <= n; ++i)
86             cur[i] = 0;
87         flw += dfs(s, oo);
88     }
89     return flw;
90 }
91 // MF Net;
92 // Net.n = n;
93 // Net.Clear();
94 // a 到 b (注意從1開始!!!!)
95 // Net.Add(a, b, w, 0);
96 // Net.MaxFlow(s, d)
97 // s 到 d 的 MF

```

4 Geometry

4.1 Closest Pair

```

1 //最近點對 (距離) //台大
2 vector<pair<double, double>> p;
3 double closest_pair(int l, int r)
4 {
5     // p 要對 x 軸做 sort
6     if (l == r)
7         return 1e9;
8     if (r - l == 1)
9         return dist(p[l], p[r]); // 兩點距離
10    int m = (l + r) >> 1;
11    double d = min(closest_pair(l, m),
12                    closest_pair(m + 1, r));
13    vector<int> vec;
14    for (int i = m; i >= l && fabs(p[m].x -
15        p[i].x) < d; --i)
16        vec.push_back(i);
17    for (int i = m + 1; i <= r && fabs(p[m].
18        x - p[i].x) < d; ++i)
19        vec.push_back(i);
20    sort(vec.begin(), vec.end(), [&](int a,
21        int b)
22        { return p[a].y < p[b].y; });
23    for (int i = 0; i < vec.size(); ++i)
24        for (int j = i + 1; j < vec.size()
25            && fabs(p[vec[j]].y - p[vec[i]].
26                y) < d; ++j)
27            d = min(d, dist(p[vec[i]], p[vec[j]]));
28    return d;
29 }

```

4.2 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &x, const point<T> &y) : p1(x), p2(y) {}
8     void pton()
9     { //轉成一般式
10         a = p1.y - p2.y;
11         b = p2.x - p1.x;
12         c = -a * p1.x - b * p1.y;
13     }
14     T ori(const point<T> &p) const
15     { //點和有向直線的關係 · >0左邊、=0在線上、<0右邊
16         return (p2 - p1).cross(p - p1);
17     }
18     T btw(const point<T> &p) const
19     { //點投影落在線段上<=0
20         return (p1 - p).dot(p2 - p);

```

```

21 }
22 bool point_on_segment(const point<T> &p)
23     const
24 { //點是否在線段上
25     return ori(p) == 0 && btw(p) <= 0;
26 }
27 T dis2(const point<T> &p, bool
28     is_segment = 0) const
29 { //點跟直線/線段的距離平方
30     point<T> v = p2 - p1, v1 = p - p1;
31     if (is_segment)
32     {
33         point<T> v2 = p - p2;
34         if (v.dot(v1) <= 0)
35             return v1.abs2();
36         if (v.dot(v2) >= 0)
37             return v2.abs2();
38     }
39     T tmp = v.cross(v1);
40     return tmp * tmp / v.abs2();
41 }
42 T seg_dis2(const line<T> &l) const
43 { //兩線段距離平方
44     return min({dis2(l.p1, 1), dis2(l.p2, 1), l.dis2(p1, 1), l.dis2(p2, 1)});
45 }
46 point<T> projection(const point<T> &p)
47     const
48 { //點對直線的投影
49     point<T> n = (p2 - p1).normal();
50     return p - n * (p - p1).dot(n) / n.abs2();
51 }
52 point<T> mirror(const point<T> &p) const
53 {
54     //點對直線的鏡射 · 要先呼叫pton轉成一般式
55     point<T> R;
56     T d = a * a + b * b;
57     R.x = (b * b * p.x - a * a * p.x - 2 * a * b * p.y - 2 * a * c) / d;
58     R.y = (a * a * p.y - b * b * p.y - 2 * a * b * p.x - 2 * b * c) / d;
59     return R;
60 }
61 bool equal(const line &l) const
62 { //直線相等
63     return ori(l.p1) == 0 && ori(l.p2) == 0;
64 }
65 bool parallel(const line &l) const
66 {
67     return (p1 - p2).cross(l.p1 - l.p2) == 0;
68 }
69 bool cross_seg(const line &l) const
70 {
71     return (p2 - p1).cross(l.p1 - p1) * (p2 - p1).cross(l.p2 - p1) <= 0;
72     //直線是否交線段
73 }
74 int line_intersect(const line &l) const
75 {
76     T x, y;
77     point() {}

```

```

78 { //直線相交情況 · -1無限多點、1交於一點、0不相交
79     return parallel(l) ? (ori(l.p1) == 0 ? -1 : 0) : 1;
80 }
81 int seg_intersect(const line &l) const
82 {
83     T c1 = ori(l.p1), c2 = ori(l.p2);
84     T c3 = l.ori(p1), c4 = l.ori(p2);
85     if (c1 == 0 && c2 == 0)
86     { //共線
87         bool b1 = btw(l.p1) >= 0, b2 = btw(l.p2) >= 0;
88         T a3 = l.btw(p1), a4 = l.btw(p2);
89         if (b1 && b2 && a3 == 0 && a4 >= 0)
90             return 2;
91         if (b1 && b2 && a3 >= 0 && a4 == 0)
92             return 3;
93         if (b1 && b2 && a3 >= 0 && a4 >= 0)
94             return 0;
95         return -1; //無限交點
96     }
97     else if (c1 * c2 <= 0 && c3 * c4 <= 0)
98         return 1;
99     return 0; //不相交
100 }
101 point<T> line_intersection(const line &l) const
102 { /*直線交點*/
103     point<T> a = p2 - p1, b = l.p2 - l.p1, s = l.p1 - p1;
104     //if(a.cross(b)==0)return INF;
105     return p1 + a * (s.cross(b) / a.cross(b));
106 }
107 point<T> seg_intersection(const line &l) const
108 { //線段交點
109     int res = seg_intersect(l);
110     if (res <= 0)
111         assert(0);
112     if (res == 2)
113         return p1;
114     if (res == 3)
115         return p2;
116     return line_intersection(l);
117 }
118 }

```

4.3 Point

```

1 template <typename T>
2 struct point
3 {
4     T x, y;
5     point() {}

```

```

6 point(const T &x, const T &y) : x(x), y(y) {}
7 point operator+(const point &b) const
8 {
9     return point(x + b.x, y + b.y);
10 }
11 point operator-(const point &b) const
12 {
13     return point(x - b.x, y - b.y);
14 }
15 point operator*(const T &b) const
16 {
17     return point(x * b, y * b);
18 }
19 point operator/(const T &b) const
20 {
21     return point(x / b, y / b);
22 }
23 bool operator==(const point &b) const
24 {
25     return x == b.x && y == b.y;
26 }
27 T dot(const point &b) const
28 {
29     return x * b.x + y * b.y;
30 }
31 T cross(const point &b) const
32 {
33     return x * b.y - y * b.x;
34 }
35 point normal() const
36 {
37     // 求法向量
38     return point(-y, x);
39 }
40 T abs2() const
41 {
42     // 向量長度的平方
43     return dot(*this);
44 }
45 T rad(const point &b) const
46 {
47     // 兩向量的弧度
48     return fabs(atan2(fabs(cross(b)), dot(b)));
49 }
50 T getA() const
51 {
52     // 對x軸的弧度
53     T A = atan2(y, x); // 超過180度會變負
54     if (A <= -PI / 2)
55         A += PI * 2;
56     return A;
57 }
58 };

```

4.4 Polygon

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}
5     vector<point<T>> p; // 逆時針順序
6     T area() const
7     { // 面積

```

```

8         T ans = 0;
9         for (int i = p.size() - 1, j = 0; j
10             < (int)p.size(); i = j++)
11             ans += p[i].cross(p[j]);
12         return ans / 2;
13     }
14     point<T> center_of_mass() const
15     { // 重心
16         T cx = 0, cy = 0, w = 0;
17         for (int i = p.size() - 1, j = 0; j
18             < (int)p.size(); i = j++)
19         {
20             T a = p[i].cross(p[j]);
21             cx += (p[i].x + p[j].x) * a;
22             cy += (p[i].y + p[j].y) * a;
23             w += a;
24         }
25         return point<T>(cx / 3 / w, cy / 3 /
26             w);
27     }
28     char ahas(const point<T> &t) const
29     { // 點是否在簡單多邊形內，是的話回傳1、
30         // 在邊上回傳-1、否則回傳0
31         bool c = 0;
32         for (int i = 0, j = p.size() - 1; i
33             < p.size(); j = i++)
34             if (line<T>(p[i], p[j]).
35                 point_on_segment(t))
36                 return -1;
37             else if ((p[i].y > t.y) != (p[j]
38                 .y > t.y) &&
39                 t.x < (p[j].x - p[i].x)
40                     * (t.y - p[i].y) /
41                     (p[j].y - p[i].y)
42                     + p[i].x)
43                 c = !c;
44         return c;
45     }
46     char point_in_convex(const point<T> &x)
47     const
48     {
49         int l = 1, r = (int)p.size() - 2;
50         while (l <= r)
51         { // 點是否在凸多邊形內，是的話回傳1
52             // 在邊上回傳-1、否則回傳0
53             int mid = (l + r) / 2;
54             T a1 = (p[mid] - p[0]).cross(x -
55                 p[0]);
56             T a2 = (p[mid + 1] - p[0]).cross
57                 (x - p[0]);
58             if (a1 >= 0 && a2 <= 0)
59             {
60                 T res = (p[mid + 1] - p[mid]
61                     ).cross(x - p[mid]);
62                 return res > 0 ? 1 : (res >=
63                     0 ? -1 : 0);
64             }
65             else if (a1 < 0)
66                 r = mid - 1;
67             else
68                 l = mid + 1;
69         }
70         return 0;
71     }
72     vector<T> getA() const

```

```

99     { // 凸包邊對x軸的夾角
100         vector<T> res; // 一定是遞增的
101         for (size_t i = 0; i < p.size(); ++i
102             )
103             res.push_back((p[(i + 1) % p.
104                 size()] - p[i]).getA());
105         return res;
106     }
107     bool line_intersect(const vector<T> &A,
108         const line<T> &l) const
109     { // O(logN)
110         int f1 = upper_bound(A.begin(), A.
111             end(), (l.p1 - l.p2).getA()) - A
112             .begin();
113         int f2 = upper_bound(A.begin(), A.
114             end(), (l.p2 - l.p1).getA()) - A
115             .begin();
116         return l.cross_seg(line<T>(p[f1], p[
117             f2]));
118     }
119     polygon cut(const line<T> &l) const
120     { // 凸包對直線切割，得到直線l左側的凸包
121         polygon ans;
122         for (int n = p.size(), i = n - 1, j
123             = 0; j < n; i = j++)
124         {
125             if (l.ori(p[i]) >= 0)
126             {
127                 ans.p.push_back(p[i]);
128                 if (l.ori(p[j]) < 0)
129                     ans.p.push_back(l.
130                         line_intersection(
131                             line<T>(p[i], p[j])),
132                             );
133             }
134             else if (l.ori(p[j]) > 0)
135                 ans.p.push_back(l.
136                     line_intersection(line<T>
137                         >(p[i], p[j])));
138         }
139         return ans;
140     }
141     static bool graham_cmp(const point<T> &a
142         , const point<T> &b)
143     { // 凸包排序函數 // 起始點不同
144         // return (a.x < b.x) || (a.x == b.x
145             && a.y < b.y); // 最左下角開始
146         return (a.y < b.y) || (a.y == b.y &&
147             a.x < b.x); // Y最小開始
148     }
149     void graham(vector<point<T>> &s)
150     { // 凸包 Convexhull 2D
151         sort(s.begin(), s.end(), graham_cmp)
152         ;
153         p.resize(s.size() + 1);
154         int m = 0;
155         // cross >= 0 順時針，cross <= 0 逆
156         // 時針旋轉
157         for (size_t i = 0; i < s.size(); ++i
158             )
159         {
160             while (m >= 2 && (p[m - 1] - p[m
161                 - 2]).cross(s[i] - p[m -
162                 2]) <= 0)

```

```

99             --m;
100             p[m++] = s[i];
101         }
102         for (int i = s.size() - 2, t = m +
103             1; i >= 0; --i)
104         {
105             while (m >= t && (p[m - 1] - p[m
106                 - 2]).cross(s[i] - p[m -
107                 2]) <= 0)
108                 --m;
109             p[m++] = s[i];
110         }
111         if (s.size() > 1) // 重複頭一次需扣
112             --m;
113         p.resize(m);
114     }
115     T diam()
116     { // 直徑
117         int n = p.size(), t = 1;
118         T ans = 0;
119         p.push_back(p[0]);
120         for (int i = 0; i < n; i++)
121         {
122             point<T> now = p[i + 1] - p[i];
123             while (now.cross(p[t + 1] - p[i]
124                 ]) > now.cross(p[t] - p[i]))
125                 t = (t + 1) % n;
126             ans = max(ans, (p[i] - p[t]).
127                 abs2());
128         }
129         return p.pop_back(), ans;
130     }
131     T min_cover_rectangle()
132     { // 最小覆蓋矩形
133         int n = p.size(), t = 1, r = 1, l;
134         if (n < 3)
135             return 0; // 也可以做最小周長矩形
136         T ans = 1e99;
137         p.push_back(p[0]);
138         for (int i = 0; i < n; i++)
139         {
140             point<T> now = p[i + 1] - p[i];
141             while (now.cross(p[t + 1] - p[i]
142                 ]) > now.cross(p[t] - p[i]))
143                 t = (t + 1) % n;
144             while (now.dot(p[r + 1] - p[i])
145                 > now.dot(p[r] - p[i]))
146                 r = (r + 1) % n;
147             if (!l)
148                 l = r;
149             while (now.dot(p[l + 1] - p[i])
150                 <= now.dot(p[l] - p[i]))
151                 l = (l + 1) % n;
152             T d = now.abs2();
153             T tmp = now.cross(p[t] - p[i]) *
154                 (now.dot(p[r] - p[i]) - now
155                     .dot(p[l] - p[i])) / d;
156             ans = min(ans, tmp);
157         }
158         return p.pop_back(), ans;
159     }
160     T dis2(polygon &pl)
161     { // 凸包最近距離平方
162         vector<point<T>> &P = p, &Q = pl.p;

```



```

153 int n = P.size(), m = Q.size(), l = 203
    0, r = 0;
154 for (int i = 0; i < n; ++i) 204
155     if (P[i].y < P[l].y) 205
156         l = i; 206
157 for (int i = 0; i < m; ++i) 207
158     if (Q[i].y < Q[r].y) 208
159         r = i;
160 P.push_back(P[0]), Q.push_back(Q[0]) 209
    ;
161 T ans = 1e99; 210
162 for (int i = 0; i < n; ++i) 211
163 { 212
164     while ((P[l] - P[l + 1]).cross(Q 213
        [r + 1] - Q[r]) < 0)
165         r = (r + 1) % m;
166     ans = min(ans, line<T>(P[l], P[l
        + 1]).seg_dis2(line<T>(Q[r
        ], Q[r + 1])));
167     l = (l + 1) % n;
168 }
169 return P.pop_back(), Q.pop_back(),
    ans;
170 }
171 static char sign(const point<T> &t)
172 {
173     return (t.y == 0 ? t.x : t.y) < 0;
174 }
175 static bool angle_cmp(const line<T> &A,
    const line<T> &B)
176 {
177     point<T> a = A.p2 - A.p1, b = B.p2 -
        B.p1;
178     return sign(a) < sign(b) || (sign(a)
        == sign(b) && a.cross(b) > 0);
179 }
180 int halfplane_intersection(vector<line<T
    >> &s)
181 { //半平面交
182     sort(s.begin(), s.end(), angle_cmp);
183     //線段左側為該線段半平面
184     int L, R, n = s.size();
185     vector<point<T>> px(n);
186     vector<line<T>> q(n);
187     q[L = R = 0] = s[0];
188     for (int i = 1; i < n; ++i)
189     {
190         while (L < R && s[i].ori(px[R -
            1]) <= 0)
191             --R;
192         while (L < R && s[i].ori(px[L])
            <= 0)
193             ++L;
194         q[++R] = s[i];
195         if (q[R].parallel(q[R - 1]))
196         {
197             --R;
198             if (q[R].ori(s[i].p1) > 0)
199                 q[R] = s[i];
200         }
201         if (L < R)
202             px[R - 1] = q[R - 1].
                line_intersection(q[R]);
    }
}

```

4.5 Triangle

```

1 template <typename T>
2 struct triangle
3 {
4     point<T> a, b, c;
5     triangle() {}
6     triangle(const point<T> &a, const point<
    T> &b, const point<T> &c) : a(a), b(b),
    c(c) {}
7     T area() const
8     {
9         T t = (b - a).cross(c - a) / 2;
10        return t > 0 ? t : -t;
11    }
12    point<T> barycenter() const
13    { //重心
14        return (a + b + c) / 3;
15    }
16    point<T> circumcenter() const
17    { //外心
18        static line<T> u, v;
19        u.p1 = (a + b) / 2;
20        u.p2 = point<T>(u.p1.x - a.y + b.y,
            u.p1.y + a.x - b.x);
21        v.p1 = (a + c) / 2;
22        v.p2 = point<T>(v.p1.x - a.y + c.y,
            v.p1.y + a.x - c.x);
23        return u.line_intersection(v);
24    }
25    point<T> incenter() const
26    { //內心
27        T A = sqrt((b - c).abs2()), B = sqrt
            ((a - c).abs2()), C = sqrt((a -
            b).abs2());
28        return point<T>(A * a.x + B * b.x +
            C * c.x, A * a.y + B * b.y + C *
            c.y) / (A + B + C);
29    }
30    point<T> perpcenter() const
31    { //垂心
32        return barycenter() * 3 -
            circumcenter() * 2;
33    }
34 }

```

5 Graph

5.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/
2 #define inf 99999 //define by you maximum
3 edges weight
4 vector<vector<int>> edges;
5 vector<int> ancestor;
6 void BellmanFord(int start, int node){
7     dist[start] = 0;
8     for(int i = 0; i < node-1; i++){
9         for(int i = 0; i < node; i++){
10             for(int j = 0; j < node; j++){
11                 if(edges[i][j] != -1){
12                     if(dist[i] + edges[i][j]
                        < dist[j]){
13                         dist[j] = dist[i] +
                            edges[i][j];
14                         ancestor[j] = i;
15                     }
16                 }
17             }
18         }
19     }
20     for(int i = 0; i < node; i++) //
        negative cycle detection
21     for(int j = 0; j < node; j++)
22         if(dist[i] + edges[i][j] < dist[
            j])
23         {
24             cout<<"Negative cycle!"<<
                endl;
25             return;
26         }
27     }
28 }
29 int main(){
30     int node;
31     cin>>node;
32     edges.resize(node, vector<int>(node, inf))
33     ;
34     dist.resize(node, inf);
35     ancestor.resize(node, -1);
36     int a, b, d;
37     while(cin>>a>>b>>d){
38         /*input: source destination weight*/
39         if(a == -1 && b == -1 && d == -1)
40             break;
41         edges[a][b] = d;
42     }
43     int start;
44     cin>>start;
45     BellmanFord(start, node);
46     return 0;
}

```

5.2 BFS-queue

```

1 /*BFS - queue version*/
2 void BFS(vector<int> &result, vector<pair<
    int, int>> edges, int node, int start)
3 {
4     vector<int> pass(node, 0);
5     queue<int> q;
6     queue<int> p;
7     q.push(start);
8     int count = 1;
9     vector<pair<int, int>> newedges;
10    while (!q.empty())
11    {
12        pass[q.front()] = 1;
13        for (int i = 0; i < edges.size(); i
            ++){
14            if (edges[i].first == q.front()
                && pass[edges[i].second] ==
                0)
15            {
16                p.push(edges[i].second);
17                result[edges[i].second] =
                    count;
18            }
19            else if (edges[i].second == q.
                front() && pass[edges[i].
                first] == 0)
20            {
21                p.push(edges[i].first);
22                result[edges[i].first] =
                    count;
23            }
24            else
25                newedges.push_back(edges[i])
                ;
26        }
27        edges = newedges;
28        newedges.clear();
29        q.pop();
30        if (q.empty() == true)
31        {
32            q = p;
33            queue<int> tmp;
34            p = tmp;
35            count++;
36        }
37    }
38 }
39 int main()
40 {
41     int node;
42     cin >> node;
43     vector<pair<int, int>> edges;
44     int a, b;
45     while (cin >> a >> b)
46     {
47         /*a = b = -1 means input edges ended
            */
48         if (a == -1 && b == -1)
49             break;
50         edges.push_back(pair<int, int>(a, b)
            );
51     }
52     vector<int> result(node, -1);
53     BFS(result, edges, node, 0);
54 }
55

```

```

56     return 0;
57 }

```

5.3 DFS-rec

```

1  /*DFS - Recursive version*/
2  map<pair<int,int>,int> edges;
3  vector<int> pass;
4  vector<int> route;
5  void DFS(int start){
6      pass[start] = 1;
7      map<pair<int,int>,int>::iterator iter;
8      for(iter = edges.begin(); iter != edges.
9          end(); iter++){
10         if((*iter).first.first == start &&
11             (*iter).second == 0 && pass[(*)
12             iter).first.second] == 0){
13             route.push_back((*iter).first.
14                 second);
15             DFS((*iter).first.second);
16         }
17     }
18     else if((*iter).first.second ==
19         start && (*iter).second == 0 &&
20         pass[(*)iter).first.first] == 0){
21         route.push_back((*iter).first.
22             first);
23         DFS((*iter).first.first);
24     }
25 }
26
27 int main(){
28     int node;
29     cin>>node;
30     pass.resize(node,0);
31     int a,b;
32     while(cin>>a>>b){
33         if(a == -1 && b == -1)
34             break;
35         edges.insert(pair<pair<int,int>,int>
36             >(pair<int,int>(a,b),0));
37     }
38     int start;
39     cin>>start;
40     route.push_back(start);
41     DFS(start);
42     return 0;
43 }

```

5.4 Dijkstra

```

1  /*SPA - Dijkstra*/
2  const int MAXN = 1e5 + 3;
3  const int inf = INT_MAX;
4  typedef pair<int, int> pii;
5  vector<vector<pii>> weight;
6  vector<int> isDone(MAXN, false), dist,
7  ancestor;
8  void dijkstra(int s)
9  {

```

```

9      priority_queue<pii, vector<pii>, greater
10         <pii>> pq;
11      pq.push(pii(0, s));
12      ancestor[s] = -1;
13      while (!pq.empty()){
14          {
15              int u = pq.top().second;
16              pq.pop();
17              isDone[u] = true;
18              for (auto &pr : weight[u])
19              {
20                  int v = pr.first, w = pr.second;
21                  if (!isDone[v] && dist[u] + w <
22                      dist[v])
23                  {
24                      dist[v] = dist[u] + w;
25                      pq.push(pii(dist[v], v));
26                      ancestor[v] = u;
27                  }
28              }
29          }
30      }
31      // weight[a - 1].push_back(pii(b - 1, w));
32      // weight[b - 1].push_back(pii(a - 1, w));
33      // dist.resize(n, inf);
34      // ancestor.resize(n, -1);
35      // dist[0] = 0;
36      // dijkstra(0);
37 }

```

5.5 Euler circuit

```

1  /*Euler circuit*/
2  /*From NTU kiseki*/
3  /*G is graph, vis is visited, la is path*/
4  bool vis[ N ]; size_t la[ K ];
5  void dfs( int u, vector< int >& vec ) {
6      while ( la[ u ] < G[ u ].size() ) {
7          if( vis[ G[ u ][ la[ u ] ].second ]
8              ) {
9              ++ la[ u ];
10             continue;
11         }
12         int v = G[ u ][ la[ u ] ].first;
13         vis[ G[ u ][ la[ u ] ].second ] = true;
14         ++ la[ u ]; dfs( v, vec );
15         vec.push_back( v );
16     }
17 }

```

5.6 Floyd-warshall

```

1  /*SPA - Floyd-Warshall*/
2  #define inf 99999
3  void floyd_warshall(vector<vector<int>>&
4      distance, vector<vector<int>>& ancestor,
5      int n){
6      for (int k = 0; k < n; k++){

```

```

5      for (int i = 0; i < n; i++){
6          for (int j = 0; j < n; j++){
7              if(distance[i][k] + distance
8                  [k][j] < distance[i][j])
9                  {
10                     distance[i][j] =
11                         distance[i][k] +
12                         distance[k][j];
13                     ancestor[i][j] =
14                         ancestor[k][j];
15                 }
16             }
17         }
18     }
19 }
20
21 int main(){
22     int n;
23     cin >> n;
24     int a, b, d;
25     vector<vector<int>> distance(n, vector<
26         int>(n,99999));
27     vector<vector<int>> ancestor(n, vector<
28         int>(n,-1));
29     while(cin>>a>>b>>d){
30         if(a == -1 && b == -1 && d == -1)
31             break;
32         distance[a][b] = d;
33         ancestor[a][b] = a;
34     }
35     for (int i = 0; i < n; i++)
36         distance[i][i] = 0;
37     floyd_warshall(distance, ancestor, n);
38     /*Negative cycle detection*/
39     for (int i = 0; i < n; i++){
40         if(distance[i][i] < 0){
41             cout << "Negative cycle!" <<
42                 endl;
43             break;
44         }
45     }
46     return 0;
47 }

```

5.7 Hamilton_cycle

```

1  /*find hamilton cycle*/
2  void hamilton(vector<vector<int>> gp, int k,
3      int cur, vector<int>& solution,vector<
4      bool> pass,bool& flag){
5      if(k == gp.size()-1){
6          if(gp[cur][1] == 1){
7              cout << 1 << " ";
8              while(cur != 1){
9                  cout << cur << " ";
10                 cur = solution[cur];
11             }
12             cout << cur << endl;
13             flag = true;
14             return;
15         }
16     }
17     for (int i = 0; i < gp[cur].size() && !
18         flag; i++){

```

```

16         if(gp[cur][i] == 1 && !pass[i]){
17             pass[i] = true;
18             solution[i] = cur;
19             hamilton(gp, k + 1, i, solution,
20                 pass,flag);
21             pass[i] = false;
22         }
23     }
24 }
25
26 int main(){
27     int n;
28     while(cin>>n){
29         int a,b;
30         bool end = false;
31         vector<vector<int>> gp(n+1,vector<
32             int>(n+1,0));
33         while(cin>>a>>b){
34             if(a == 0 && b == 0)
35                 break;
36             gp[a][b] = 1;
37             gp[b][a] = 1;
38         }
39         vector<int> solution(n + 1, -1);
40         vector<bool> pass(n + 1, false);
41         solution[1] = 0;
42         pass[1] = true;
43         bool flag = false;
44         hamilton(gp, 1,1 ,solution,pass,flag
45             );
46         if(!flag)
47             cout << "N" << endl;
48     }
49     return 0;
50 }
51
52 /*
53 4
54 1 2
55 2 3
56 2 4
57 3 4
58 3 1
59 0 0
60 output: 1 3 4 2 1
61 */

```

5.8 Kruskal

```

1  /*mst - Kruskal*/
2  struct edges{
3      int from;
4      int to;
5      int weight;
6      friend bool operator < (edges a, edges b
7          ){
8          return a.weight > b.weight;
9      }
10 };
11
12 int find(int x,vector<int>& union_set){
13     if(x != union_set[x])
14         union_set[x] = find(union_set[x],
15             union_set);
16     return union_set[x];
17 }

```

```

15 void merge(int a,int b,vector<int>&
    union_set){
16     int pa = find(a, union_set);
17     int pb = find(b, union_set);
18     if(pa != pb)
19         union_set[pa] = pb;
20 }
21 void kruskal(priority_queue<edges> pq,int n)
22 {
23     vector<int> union_set(n, 0);
24     for (int i = 0; i < n; i++)
25         union_set[i] = i;
26     int edge = 0;
27     int cost = 0; //evaluate cost of mst
28     while(!pq.empty() && edge < n - 1){
29         edges cur = pq.top();
30         int from = find(cur.from, union_set);
31         int to = find(cur.to, union_set);
32         if(from != to){
33             merge(from, to, union_set);
34             edge += 1;
35             cost += cur.weight;
36         }
37         pq.pop();
38     }
39     if(edge < n-1)
40         cout << "No mst" << endl;
41     else
42         cout << cost << endl;
43 }
44 int main(){
45     int n;
46     cin >> n;
47     int a, b, d;
48     priority_queue<edges> pq;
49     while(cin>>a>>b>>d){
50         if(a == -1 && b == -1 && d == -1)
51             break;
52         edges tmp;
53         tmp.from = a;
54         tmp.to = b;
55         tmp.weight = d;
56         pq.push(tmp);
57     }
58     kruskal(pq, n);
59     return 0;

```

5.9 Prim

```

1 /*mst - Prim*/
2 #define inf 99999
3 struct edges{
4     int from;
5     int to;
6     int weight;
7     friend bool operator < (edges a, edges b)
8     ){
9         return a.weight > b.weight;
10 }
11 };

```

```

11 void Prim(vector<vector<int>> gp,int n,int
    start){
12     vector<bool> pass(n,false);
13     int edge = 0;
14     int cost = 0; //evaluate cost of mst
15     priority_queue<edges> pq;
16     for (int i = 0; i < n; i++){
17         if(gp[start][i] != inf){
18             edges tmp;
19             tmp.from = start;
20             tmp.to = i;
21             tmp.weight = gp[start][i];
22             pq.push(tmp);
23         }
24     }
25     pass[start] = true;
26     while(!pq.empty() && edge < n-1){
27         edges cur = pq.top();
28         pq.pop();
29         if(!pass[cur.to]){
30             for (int i = 0; i < n; i++){
31                 if(gp[cur.to][i] != inf){
32                     edges tmp;
33                     tmp.from = cur.to;
34                     tmp.to = i;
35                     tmp.weight = gp[cur.to][i];
36                     pq.push(tmp);
37                 }
38             }
39             pass[cur.to] = true;
40             edge += 1;
41             cost += cur.weight;
42         }
43     }
44     if(edge < n-1)
45         cout << "No mst" << endl;
46     else
47         cout << cost << endl;
48 }
49 int main(){
50     int n;
51     cin >> n;
52     int a, b, d;
53     vector<vector<int>> gp(n,vector<int>(n,
        inf));
54     while(cin>>a>>b>>d){
55         if(a == -1 && b == -1 && d == -1)
56             break;
57         if(gp[a][b] > d)
58             gp[a][b] = d;
59     }
60     Prim(gp,n,0);
61     return 0;
62 }

```

5.10 Union_find

```

1 int find(int x, vector<int> &union_set)
2 {
3     if (union_set[x] != x)
4         union_set[x] = find(union_set[x],
        union_set); //compress path

```

```

5     return union_set[x];
6 }
7 void merge(int x, int y, vector<int> &
    union_set, vector<int> &rank)
8 {
9     int rx, ry;
10    rx = find(x, union_set);
11    ry = find(y, union_set);
12    if (rx == ry)
13        return;
14    /*merge by rank -> always merge small
        tree to big tree*/
15    if (rank[rx] > rank[ry])
16        union_set[ry] = rx;
17    else
18    {
19        union_set[rx] = ry;
20        ++rank[ry];
21    }
22 }
23 int main()
24 {
25     int node;
26     cin >> node; //Input Node number
27     vector<int> union_set(node, 0);
28     vector<int> rank(node, 0);
29     for (int i = 0; i < node; i++)
30         union_set[i] = i;
31     int edge;
32     cin >> edge; //Input Edge number
33     for (int i = 0; i < edge; i++)
34     {
35         int a, b;
36         cin >> a >> b;
37         merge(a, b, union_set, rank);
38     }
39     /*build party*/
40     vector<vector<int>> party(node, vector<
        int>(0));
41     for (int i = 0; i < node; i++)
42         party[find(i, union_set)].push_back(
            i);
43 }

```

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

6.2 Combination

```

1 /*input type string or vector*/
2 for (int i = 0; i < (1 << input.size()); ++i
3 )
4 {
5     string testCase = "";
6     for (int j = 0; j < input.size(); ++j)
7         if (i & (1 << j))
8             testCase += input[j];

```

6.3 Extended Euclidean

```

1 // ax + by = gcd(a,b)
2 pair<long long, long long> extgcd(long long
    a, long long b)
3 {
4     if (b == 0)
5         return {1, 0};
6     long long k = a / b;
7     pair<long long, long long> p = extgcd(b,
        a - k * b);
8     //cout << p.first << " " << p.second <<
        endl;
9     //cout << "商數(k)= " << k << endl <<
        endl;
10    return {p.second, p.first - k * p.second
        };
11 }
12 int main()
13 {
14     int a, b;
15     cin >> a >> b;
16     pair<long long, long long> xy = extgcd(a,
        b); //(x0,y0)
17     cout << xy.first << " " << xy.second <<
        endl;
18     cout << xy.first << " * " << a << " + "
        << xy.second << " * " << b << endl;
19     return 0;
20 }
21 // ax + by = gcd(a,b) * r
22 /*find |x|+|y| -> min*/
23 int main()
24 {
25     long long r, p, q; /*px+qy = r*/
26     int cases;
27     cin >> cases;
28     while (cases--)
29     {
30         cin >> r >> p >> q;
31         pair<long long, long long> xy =
            extgcd(q, p); //(x0,y0)
32         long long ans = 0, tmp = 0;
33         double k, k1;
34         long long s, s1;
35         k = 1 - (double)(r * xy.first) / p;
36         s = round(k);
37         ans = llabs(r * xy.first + s * p) +
            llabs(r * xy.second - s * q);

```



```

39 k1 = -(double)(r * xy.first) / p;
40 s1 = round(k1);
41 /*cout << k << endl << k1 << endl;
42 cout << s << endl << s1 << endl;
43 */
44 tmp = llabs(r * xy.first + s1 * p) +
45 llabs(r * xy.second - s1 * q);
46 ans = min(ans, tmp);
47 cout << ans << endl;
48 }
49 return 0;
}

```

```

13 {
14     temp += (num[i] - 55) * base;
15     base = base * 16;
16 }
17 }
18 return temp;
19 }
20 void DecToHex(int p) //10 to 16
21 {
22     char *l = new (char);
23     sprintf(l, "%X", p);
24     //int l_intResult = stoi(l);
25     cout << l << "\n";
26     //return l_intResult;
27 }

```

6.4 Fermat

Little 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 Log

```

1 double mylog(double a, double base)
2 {
3     //a 的對數底數 b = 自然對數 (a) / 自然對
4     //數 (b)。
5     return log(a) / log(base);
}

```

6.7 Mod

6.5 Hex to Dec

```

1 int HextoDec(string num) //16 to 10
2 {
3     int base = 1;
4     int temp = 0;
5     for (int i = num.length() - 1; i >= 0; i--)
6     {
7         if (num[i] >= '0' && num[i] <= '9')
8         {
9             temp += (num[i] - 48) * base;
10            base = base * 16;
11        }
12        else if (num[i] >= 'A' && num[i] <= 'F')

```

```

1 int pow_mod(int a, int n, int m) // a ^ n
2 {
3     mod m;
4     { // a, n, m < 10 ^ 9
5         if (n == 0)
6             return 1;
7         int x = pow_mid(a, n / 2, m);
8         long long ans = (long long)x * x % m;
9         if (n % 2 == 1)
10            ans = ans * a % m;
11        return (int)ans;
12    }
}

```

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： $((a + b) \bmod p * c) = ((a * c) \bmod p + (b * c) \bmod p) \bmod p$

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 $\bmod c$ 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 $\gcd(a, n) = 1$ ， $a^{\Phi(n)} \equiv 1 \pmod{n}$ 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

```

1 #define PI acos(-1)
2 #define PI_M_PI
3 const double PI = atan2(0.0, -1.0);

```

6.10 Prime table

```

1 const int maxn = sqrt(INT_MAX);

```

```

2 vector<int> p;
3 bitset<maxn> is_notp;
4 void PrimeTable()
5 {
6     is_notp.reset();
7     is_notp[0] = is_notp[1] = 1;
8     for (int i = 2; i <= maxn; ++i)
9     {
10         if (!is_notp[i])
11             p.push_back(i);
12         for (int j = 0; j < (int)p.size(); ++j)
13         {
14             if (i * p[j] > maxn)
15                 break;
16             is_notp[i * p[j]] = 1;
17             if (i % p[j] == 0)
18                 break;
19         }
20     }
21 }

```

6.11 Prime 判斷

```

1 // n < 4759123141     chk = [2, 7, 61]
2 // n < 1122004669633  chk = [2, 13, 23,
3 //                    1662803]
4 // n < 2^64           chk = [2, 325, 9375,
5 //                    28178, 450775, 9780504, 1795265022]
6 vector<long long> chk = {};
7 long long fmul(long long a, long long n,
8 long long mod)
9 {
10     long long ret = 0;
11     for (; n; n >>= 1)
12     {
13         if (n & 1)
14             (ret += a) %= mod;
15         (a += a) %= mod;
16     }
17     return ret;
18 }
19 long long fpow(long long a, long long n,
20 long long mod)
21 {
22     long long ret = 1LL;
23     for (; n; n >>= 1)
24     {
25         if (n & 1)
26             ret = fmul(ret, a, mod);
27         a = fmul(a, a, mod);
28     }
29     return ret;
30 }
31 bool check(long long a, long long u, long
32 long n, int t)
33 {
34     a = fpow(a, u, n);
35     if (a == 0)
36         return true;
37     if (a == 1 || a == n - 1)
38         return true;

```

```

35 for (int i = 0; i < t; ++i)
36 {
37     a = fmul(a, a, n);
38     if (a == 1)
39         return false;
40     if (a == n - 1)
41         return true;
42 }
43 return false;
44 }
45 bool is_prime(long long n)
46 {
47     if (n < 2)
48         return false;
49     if (n % 2 == 0)
50         return n == 2;
51     long long u = n - 1;
52     int t = 0;
53     for (; u & 1; u >>= 1, ++t)
54         ;
55     for (long long i : chk)
56     {
57         if (!check(i, u, n, t))
58             return false;
59     }
60     return true;
61 }
62 // if (is_prime(int num)) // true == prime
63 // 反之亦然

```

6.12 Round(小數)

```

1 double myround(double number, unsigned int
2   bits)
3 {
4     LL integerPart = number;
5     number -= integerPart;
6     for (unsigned int i = 0; i < bits; ++i)
7         number *= 10;
8     number = (LL)(number + 0.5);
9     for (unsigned int i = 0; i < bits; ++i)
10        number /= 10;
11     return integerPart + number;
12 }
13 //printf("%.1f\n", round(3.4515239, 1));

```

6.13 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else

```

```

11        R = M;
12    }
13    printf("%.3lf\n", R);
14 }

```

6.14 公式

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

$$\sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6} \quad \sum_{k=1}^n k^3 = \left[\frac{n(n+1)}{2} \right]^2$$

6.15 四則運算

```

1 string s = ""; //開頭是負號要補0
2 long long int DFS(int le, int ri) // (0,
3   string final index)
4 {
5     int c = 0;
6     for (int i = ri; i >= le; i--)
7     {
8         if (s[i] == '(')
9             c++;
10        if (s[i] == ')')
11            c--;
12        if (s[i] == '+' && c == 0)
13            return DFS(le, i - 1) + DFS(i + 1, ri);
14        if (s[i] == '-' && c == 0)
15            return DFS(le, i - 1) - DFS(i + 1, ri);
16    }
17    for (int i = ri; i >= le; i--)
18    {
19        if (s[i] == '(')
20            c++;
21        if (s[i] == ')')
22            c--;
23        if (s[i] == '*' && c == 0)
24            return DFS(le, i - 1) * DFS(i + 1, ri);
25        if (s[i] == '/' && c == 0)
26            return DFS(le, i - 1) / DFS(i + 1, ri);
27        if (s[i] == '%' && c == 0)
28            return DFS(le, i - 1) % DFS(i + 1, ri);
29    }
30    if ((s[le] == '(' && (s[ri] == ')'))
31        return DFS(le + 1, ri - 1); //去除刮
32        號
33    if (s[le] == ' ' && s[ri] == ' ')
34        return DFS(le + 1, ri - 1); //去除左
35        右兩邊空格
36    if (s[le] == ' ')
37        return DFS(le + 1, ri); //去除左邊空
38        格
39    if (s[ri] == ' ')

```

```

36 return DFS(le, ri - 1); //去除右邊空
37 格
38 long long int num = 0;
39 for (int i = le; i <= ri; i++)
40     num = num * 10 + s[i] - '0';
41 return num;

```

6.16 數字乘法組合

```

1 void dfs(int j, int old, int num, vector<int>
2   > com, vector<vector<int>> &ans)
3 {
4     for (int i = j; i <= sqrt(num); i++)
5     {
6         if (old == num)
7             com.clear();
8         if (num % i == 0)
9         {
10            vector<int> a;
11            a = com;
12            a.push_back(i);
13            finds(i, old, num / i, a, ans);
14            a.push_back(num / i);
15            ans.push_back(a);
16        }
17    }
18    vector<vector<int>> ans;
19    vector<int> zero;
20    dfs(2, num, num, zero, ans);
21    /*num 為 input 數字*/
22    for (int i = 0; i < ans.size(); i++)
23    {
24        for (int j = 0; j < ans[i].size() - 1; j++)
25            cout << ans[i][j] << " ";
26        cout << ans[i][ans[i].size() - 1] <<
27        endl;

```

6.17 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
2   &out, vector<vector<int>> &ans)
3 {
4     if (n == 0)
5     {
6         for (int i : out)
7             if (i > m)
8                 return;
9         ans.push_back(out);
10    }
11    for (int j = i; j <= n; j++)
12    {
13        out.push_back(j);
14        recur(j, n - j, m, out, ans);
15        out.pop_back();

```

```

16 }
17 vector<vector<int>> ans;
18 vector<int> zero;
19 recur(1, num, num, zero, ans);
20 // num 為 input 數字
21 for (int i = 0; i < ans.size(); i++)
22 {
23     for (int j = 0; j < ans[i].size() - 1; j++)
24         cout << ans[i][j] << " ";
25     cout << ans[i][ans[i].size() - 1] <<
26     endl;

```

6.18 羅馬數字

```

1 int romanToInt(string s)
2 {
3     unordered_map<char, int> T;
4     T['I'] = 1;
5     T['V'] = 5;
6     T['X'] = 10;
7     T['L'] = 50;
8     T['C'] = 100;
9     T['D'] = 500;
10    T['M'] = 1000;
11
12    int sum = T[s.back()];
13    for (int i = s.length() - 2; i >= 0; --i)
14    {
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19    }
20    return sum;
21 }

```

6.19 質因數分解

```

1 void primeFactorization(int n) // 配合質數表
2 {
3     for (int i = 0; i < (int)p.size(); ++i)
4     {
5         if (p[i] * p[i] > n)
6             break;
7         if (n % p[i])
8             continue;
9         cout << p[i] << ' ';
10        while (n % p[i] == 0)
11            n /= p[i];
12    }
13    if (n != 1)
14        cout << n << ' ';
15    cout << '\n';
16 }

```

7 Other

7.1 binary search 三類變化

```

1 // 查找和目標值完全相等的數
2 int find(vector<int> &nums, int target)
3 {
4     int left = 0, right = nums.size();
5     while (left < right)
6     {
7         int mid = left + (right - left) / 2;
8         if (nums[mid] == target)
9             return mid;
10        else if (nums[mid] < target)
11            left = mid + 1;
12        else
13            right = mid;
14    }
15    return -1;
16 }
17 // 找第一個不小於目標值的數 == 找最後一個小
    於目標值的數
18 /*(lower_bound)*/
19 int find(vector<int> &nums, int target)
20 {
21     int left = 0, right = nums.size();
22     while (left < right)
23     {
24         int mid = left + (right - left) / 2;
25         if (nums[mid] < target)
26             left = mid + 1;
27        else
28            right = mid;
29    }
30    return right;
31 }
32 // 找第一個大於目標值的數 == 找最後一個不大
    於目標值的數
33 /*(upper_bound)*/
34 int find(vector<int> &nums, int target)
35 {
36     int left = 0, right = nums.size();
37     while (left < right)
38     {
39         int mid = left + (right - left) / 2;
40         if (nums[mid] <= target)
41             left = mid + 1;
42        else
43            right = mid;
44    }
45    return right;
46 }

```

7.2 heap sort

```

1 void MaxHeapify(vector<int> &array, int root
    , int length)
2 {
3     int left = 2 * root,
4         right = 2 * root + 1,

```

```

5         largest;
6     if (left <= length && array[left] >
        array[root])
7         largest = left;
8     else
9         largest = root;
10    if (right <= length && array[right] >
        array[largest])
11        largest = right;
12    if (largest != root)
13    {
14        swap(array[largest], array[root]);
15        MaxHeapify(array, largest, length);
16    }
17 }
18 void HeapSort(vector<int> &array)
19 {
20     array.insert(array.begin(), 0);
21     for (int i = (int)array.size() / 2; i >=
        1; i--)
22        MaxHeapify(array, i, (int)array.size
        () - 1);
23    int size = (int)array.size() - 1;
24    for (int i = (int)array.size() - 1; i >=
        2; i--)
25    {
26        swap(array[1], array[i]);
27        size--;
28        MaxHeapify(array, 1, size);
29    }
30    array.erase(array.begin());
31 }

```

7.3 Merge sort

```

1 void Merge(vector<int> &arr, int front, int
    mid, int end)
2 {
3     vector<int> LeftSub(arr.begin() + front,
        arr.begin() + mid + 1);
4     vector<int> RightSub(arr.begin() + mid +
        1, arr.begin() + end + 1);
5     LeftSub.insert(LeftSub.end(), INT_MAX);
6     RightSub.insert(RightSub.end(), INT_MAX)
        ;
7     int idxLeft = 0, idxRight = 0;
8     for (int i = front; i <= end; i++)
9     {
10        if (LeftSub[idxLeft] <= RightSub[
            idxRight])
11        {
12            arr[i] = LeftSub[idxLeft];
13            idxLeft++;
14        }
15        else
16        {
17            arr[i] = RightSub[idxRight];
18            idxRight++;
19        }
20    }
21 }
22 }
23 }

```

```

24 void MergeSort(vector<int> &arr, int front,
    int end)
25 {
26     // front = 0, end = arr.size() - 1
27     if (front < end)
28     {
29         int mid = (front + end) / 2;
30         MergeSort(arr, front, mid);
31         MergeSort(arr, mid + 1, end);
32         Merge(arr, front, mid, end);
33     }
34 }

```

7.4 Quick

```

1 int Partition(vector<int> &arr, int front,
    int end)
2 {
3     int pivot = arr[end];
4     int i = front - 1;
5     for (int j = front; j < end; j++)
6     {
7         if (arr[j] < pivot)
8         {
9             i++;
10            swap(arr[i], arr[j]);
11        }
12    }
13    i++;
14    swap(arr[i], arr[end]);
15    return i;
16 }
17 void QuickSort(vector<int> &arr, int front,
    int end)
18 {
19     // front = 0, end = arr.size() - 1
20     if (front < end)
21     {
22         int pivot = Partition(arr, front,
            end);
23         QuickSort(arr, front, pivot - 1);
24         QuickSort(arr, pivot + 1, end);
25    }
26 }

```

7.5 Weighted Job Scheduling

```

1 struct Job
2 {
3     int start, finish, profit;
4 };
5 bool jobComparataor(Job s1, Job s2)
6 {
7     return (s1.finish < s2.finish);
8 }
9 int latestNonConflict(Job arr[], int i)
10 {
11     for (int j = i - 1; j >= 0; j--)
12     {
13         if (arr[j].finish <= arr[i].start)

```

```

14         return j;
15    }
16    return -1;
17 }
18 int findMaxProfit(Job arr[], int n)
19 {
20     sort(arr, arr + n, jobComparataor);
21     int *table = new int[n];
22     table[0] = arr[0].profit;
23     for (int i = 1; i < n; i++)
24     {
25         int inclProf = arr[i].profit;
26         int l = latestNonConflict(arr, i);
27         if (l != -1)
28             inclProf += table[l];
29         table[i] = max(inclProf, table[i -
            1]);
30    }
31    int result = table[n - 1];
32    delete[] table;
33    return result;
34 }
35 }

```

7.6 數獨解法

```

1 int getSquareIndex(int row, int column, int
    n)
2 {
3     return row / n * n + column / n;
4 }
5
6 bool backtracking(vector<vector<int>> &board
    , vector<vector<bool>> &rows, vector<
    vector<bool>> &cols,
7         vector<vector<bool>> &boxes
        , int index, int n)
8 {
9     int n2 = n * n;
10    int rowNum = index / n2, colNum = index
        % n2;
11    if (index >= n2 * n2)
12        return true;
13
14    if (board[rowNum][colNum] != 0)
15        return backtracking(board, rows,
        cols, boxes, index + 1, n);
16
17    for (int i = 1; i <= n2; i++)
18    {
19        if (!rows[rowNum][i] && !cols[colNum
            ][i] && !boxes[getSquareIndex(
                rowNum, colNum, n)][i])
20        {
21            rows[rowNum][i] = true;
22            cols[colNum][i] = true;
23            boxes[getSquareIndex(rowNum,
                colNum, n)][i] = true;
24            board[rowNum][colNum] = i;
25            if (backtracking(board, rows,
                cols, boxes, index + 1, n))
26                return true;
27            board[rowNum][colNum] = 0;

```

```

28     rows[rowNum][i] = false;
29     cols[colNum][i] = false;
30     boxes[getSquareIndex(rowNum,
31         colNum, n)][i] = false;
32 }
33 return false;
34 }
35 /*用法 main*/
36 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
37 vector<vector<int>> board(n * n + 1, vector<
38     int>(n * n + 1, 0));
39 vector<vector<bool>> isRow(n * n + 1, vector<
40     bool>(n * n + 1, false));
41 vector<vector<bool>> isColumn(n * n + 1,
42     vector<bool>(n * n + 1, false));
43 vector<vector<bool>> isSquare(n * n + 1,
44     vector<bool>(n * n + 1, false));
45 for (int i = 0; i < n * n; ++i)
46 {
47     for (int j = 0; j < n * n; ++j)
48     {
49         int number;
50         cin >> number;
51         board[i][j] = number;
52         if (number == 0)
53             continue;
54         isRow[i][number] = true;
55         isColumn[j][number] = true;
56         isSquare[getSquareIndex(i, j, n)][
57             number] = true;
58     }
59 }
60 if (backtracking(board, isRow, isColumn,
61     isSquare, 0, n))
62     /*有解答*/
63 else
64     /*解答*/

```

8 String

8.1 KMP

```

1 // 用在在一個 S 內查找一個詞 W 的出現位置
2 void ComputePrefix(string s, int next[])
3 {
4     int n = s.length();
5     int q, k;
6     next[0] = 0;
7     for (k = 0, q = 1; q < n; q++)
8     {
9         while (k > 0 && s[k] != s[q])
10             k = next[k];
11         if (s[k] == s[q])
12             k++;
13         next[q] = k;
14     }
15 }
16 void KMPMatcher(string text, string pattern)

```

```

17 {
18     int n = text.length();
19     int m = pattern.length();
20     int next[pattern.length()];
21     ComputePrefix(pattern, next);
22
23     for (int i = 0, q = 0; i < n; i++)
24     {
25         while (q > 0 && pattern[q] != text[i])
26             q = next[q];
27         if (pattern[q] == text[i])
28             q++;
29         if (q == m)
30         {
31             cout << "Pattern occurs with
32                 shift " << i - m + 1 << endl;
33             q = 0;
34         }
35     }
36     // string s = "abcdabcdebcd";
37     // string p = "bcd";
38     // KMPMatcher(s, p);
39     // cout << endl;

```

8.2 Min Edit Distance

```

1 int EditDistance(string a, string b)
2 {
3     vector<vector<int>> dp(a.size() + 1,
4         vector<int>(b.size() + 1, 0));
5     int m = a.length(), n = b.length();
6     for (int i = 0; i < m + 1; i++)
7     {
8         for (int j = 0; j < n + 1; j++)
9         {
10             if (i == 0)
11                 dp[i][j] = j;
12             else if (j == 0)
13                 dp[i][j] = i;
14             else if (a[i - 1] == b[j - 1])
15                 dp[i][j] = dp[i - 1][j - 1];
16             else
17                 dp[i][j] = 1 + min(min(dp[i]
18                     - 1][j], dp[i][j - 1]),
19                     dp[i - 1][j - 1]);
20         }
21     }
22     return dp[m][n];

```

8.3 Sliding window

```

1 string minWindow(string s, string t)
2 {
3     unordered_map<char, int> letterCnt;
4     for (int i = 0; i < t.length(); i++)
5         letterCnt[t[i]]++;

```

```

6     int minLength = INT_MAX, minStart = -1;
7     int left = 0, matchCnt = 0;
8     for (int i = 0; i < s.length(); i++)
9     {
10         if (--letterCnt[s[i]] >= 0)
11             matchCnt++;
12         while (matchCnt == t.length())
13         {
14             if (i - left + 1 < minLength)
15             {
16                 minLength = i - left + 1;
17                 minStart = left;
18             }
19             if (++letterCnt[s[left]] > 0)
20                 matchCnt--;
21             left++;
22         }
23     }
24     return minLength == INT_MAX ? "" : s.
25         substr(minStart, minLength);

```

8.4 Split

```

1 vector<string> mysplit(const string &str,
2     const string &delim)
3 {
4     vector<string> res;
5     if (" " == str)
6         return res;
7
8     char *strs = new char[str.length() + 1];
9     char *d = new char[delim.length() + 1];
10    strcpy(strs, str.c_str());
11    strcpy(d, delim.c_str());
12    char *p = strtok(strs, d);
13    while (p)
14    {
15        string s = p;
16        res.push_back(s);
17        p = strtok(NULL, d);
18    }
19    return res;

```

9 data structure

9.1 Bigint

```

1 //台大 //非必要請用python
2 struct Bigint
3 {
4     static const int LEN = 60; //
5     static const int BIGMOD = 10000; //10為
6     //正常位數
7     int s;

```

```

7     int v1, v[LEN];
8     // vector<int> v;
9     Bigint() : s(1) { v1 = 0; }
10    Bigint(long long a)
11    {
12        s = 1;
13        v1 = 0;
14        if (a < 0)
15        {
16            s = -1;
17            a = -a;
18        }
19        while (a)
20        {
21            push_back(a % BIGMOD);
22            a /= BIGMOD;
23        }
24    }
25    Bigint(string str)
26    {
27        s = 1;
28        v1 = 0;
29        int stPos = 0, num = 0;
30        if (!str.empty() && str[0] == '-')
31        {
32            stPos = 1;
33            s = -1;
34        }
35        for (int i = str.length() - 1, q =
36            1; i >= stPos; i--)
37        {
38            num += (str[i] - '0') * q;
39            if ((q *= 10) >= BIGMOD)
40            {
41                push_back(num);
42                num = 0;
43                q = 1;
44            }
45            if (num)
46                push_back(num);
47            n();
48        }
49        int len() const
50        {
51            return v1; //return SZ(v);
52        }
53        bool empty() const { return len() == 0; }
54        void push_back(int x)
55        {
56            v[v1++] = x; //v.PB(x);
57        }
58        void pop_back()
59        {
60            v1--; //v.pop_back();
61        }
62        int back() const
63        {
64            return v[v1 - 1]; //return v.back();
65        }
66        void n()
67        {
68            while (!empty() && !back())
69                pop_back();
70        }

```

```

71 void resize(int nl)
72 {
73     vl = nl; //v.resize(nl);
74     fill(v, v + vl, 0); //fill(ALL(v),
75         0);
76 }
77 void print() const
78 {
79     if (empty())
80     {
81         putchar('0');
82         return;
83     }
84     if (s == -1)
85         putchar('-');
86     printf("%d", back());
87     for (int i = len() - 2; i >= 0; i--)
88         printf("%.4d", v[i]);
89 }
90 friend std::ostream &operator<<(std::
91     ostream &out, const Bigint &a)
92 {
93     if (a.empty())
94     {
95         out << "0";
96         return out;
97     }
98     if (a.s == -1)
99         out << "-";
100     out << a.back();
101     for (int i = a.len() - 2; i >= 0; i
102         --)
103     {
104         char str[10];
105         snprintf(str, 5, "%.4d", a.v[i])
106             ;
107         out << str;
108     }
109     return out;
110 }
111 int cp3(const Bigint &b) const
112 {
113     if (s != b.s)
114         return s - b.s;
115     if (s == -1)
116         return -(*this).cp3(-b);
117     if (len() != b.len())
118         return len() - b.len(); //int
119     for (int i = len() - 1; i >= 0; i--)
120         if (v[i] != b.v[i])
121             return v[i] - b.v[i];
122     return 0;
123 }
124 bool operator<(const Bigint &b) const
125 {
126     return cp3(b) < 0;
127 }
128 bool operator<=(const Bigint &b) const
129 {
130     return cp3(b) <= 0;
131 }
132 bool operator==(const Bigint &b) const

```

```

133 {
134     return cp3(b) != 0;
135 }
136 bool operator>(const Bigint &b) const
137 {
138     return cp3(b) > 0;
139 }
140 bool operator>=(const Bigint &b) const
141 {
142     return cp3(b) >= 0;
143 }
144 Bigint operator-() const
145 {
146     Bigint r = (*this);
147     r.s = -r.s;
148     return r;
149 }
150 Bigint operator+(const Bigint &b) const
151 {
152     if (s == -1)
153         return -(*this) + (-b);
154     if (b.s == -1)
155         return (*this) - (-b);
156     Bigint r;
157     int nl = max(len(), b.len());
158     r.resize(nl + 1);
159     for (int i = 0; i < nl; i++)
160     {
161         if (i < len())
162             r.v[i] += v[i];
163         if (i < b.len())
164             r.v[i] += b.v[i];
165         if (r.v[i] >= BIGMOD)
166         {
167             r.v[i + 1] += r.v[i] /
168                 BIGMOD;
169             r.v[i] %= BIGMOD;
170         }
171     }
172     r.n();
173     return r;
174 }
175 Bigint operator-(const Bigint &b) const
176 {
177     if (s == -1)
178         return -(*this) - (-b);
179     if (b.s == -1)
180         return (*this) + (-b);
181     if ((*this) < b)
182         return -(*this);
183     Bigint r;
184     r.resize(len());
185     for (int i = 0; i < len(); i++)
186     {
187         r.v[i] += v[i];
188         if (i < b.len())
189             r.v[i] -= b.v[i];
190         if (r.v[i] < 0)
191         {
192             r.v[i] += BIGMOD;
193             r.v[i + 1]--;
194         }
195     }
196     r.n();
197     return r;
198 }

```

```

199 {
200     Bigint r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j
206             ++
207         {
208             r.v[i + j] += v[i] * b.v[j];
209             if (r.v[i + j] >= BIGMOD)
210             {
211                 r.v[i + j + 1] += r.v[i
212                     + j] / BIGMOD;
213                 r.v[i + j] %= BIGMOD;
214             }
215         }
216     }
217     r.n();
218     return r;
219 }
220 Bigint operator/(const Bigint &b)
221 {
222     Bigint r;
223     r.resize(max(1, len() - b.len() + 1)
224         );
225     int oriS = s;
226     Bigint b2 = b; // b2 = abs(b)
227     s = b2.s = r.s = 1;
228     for (int i = r.len() - 1; i >= 0; i
229         --)
230     {
231         int d = 0, u = BIGMOD - 1;
232         while (d < u)
233         {
234             int m = (d + u + 1) >> 1;
235             r.v[i] = m;
236             if ((r * b2) > (*this))
237                 u = m - 1;
238             else
239                 d = m;
240         }
241         r.v[i] = d;
242     }
243     s = oriS;
244     r.s = s * b.s;
245     r.n();
246     return r;
247 }
248 Bigint operator%(const Bigint &b)
249 {
250     return (*this) - (*this) / b * b;
251 }

```

9.2 Matirx

```

1 template <typename T>
2 struct Matrix
3 {
4     using rt = std::vector<T>;
5     using mt = std::vector<rt>;

```

```

6 using matrix = Matrix<T>;
7 int r, c; // [r][c]
8 mt m;
9 Matrix(int r, int c) : r(r), c(c), m(r,
10     rt(c)) {}
11 Matrix(mt a) { m = a, r = a.size(), c =
12     a[0].size(); }
13 rt &operator[] (int i) { return m[i]; }
14 matrix operator+(const matrix &a)
15 {
16     matrix rev(r, c);
17     for (int i = 0; i < r; ++i)
18         for (int j = 0; j < c; ++j)
19             rev[i][j] = m[i][j] + a.m[i
20                 ][j];
21     return rev;
22 }
23 matrix operator-(const matrix &a)
24 {
25     matrix rev(r, c);
26     for (int i = 0; i < r; ++i)
27         for (int j = 0; j < c; ++j)
28             rev[i][j] = m[i][j] - a.m[i
29                 ][j];
30     return rev;
31 }
32 matrix operator*(const matrix &a)
33 {
34     matrix rev(r, a.c);
35     matrix tmp(a.c, a.r);
36     for (int i = 0; i < a.r; ++i)
37         for (int j = 0; j < a.c; ++j)
38             tmp[j][i] = a.m[i][j];
39     for (int i = 0; i < r; ++i)
40         for (int j = 0; j < a.c; ++j)
41             for (int k = 0; k < c; ++k)
42                 rev.m[i][j] += m[i][k] *
43                     tmp[j][k];
44     return rev;
45 }
46 bool inverse() //逆矩陣判斷
47 {
48     Matrix t(r, r + c);
49     for (int y = 0; y < r; y++)
50     {
51         t.m[y][c + y] = 1;
52         for (int x = 0; x < c; ++x)
53             t.m[y][x] = m[y][x];
54     }
55     if (!t.gas())
56         return false;
57     for (int y = 0; y < r; y++)
58         for (int x = 0; x < c; ++x)
59             m[y][x] = t.m[y][c + x] / t.
60                 m[y][y];
61     return true;
62 }
63 T gas() //行列式
64 {
65     vector<T> lazy(r, 1);
66     bool sign = false;
67     for (int i = 0; i < r; ++i)
68     {
69         if (m[i][i] == 0)
70         {

```



```

65     int j = i + 1;
66     while (j < r && !m[j][i])
67         j++;
68     if (j == r)
69         continue;
70     m[i].swap(m[j]);
71     sign = !sign;
72 }
73 for (int j = 0; j < r; ++j)
74 {
75     if (i == j)
76         continue;
77     lazy[j] = lazy[j] * m[i][i];
78     T mx = m[j][i];
79     for (int k = 0; k < c; ++k)
80         m[j][k] = m[j][k] * m[i]
81             [i] - m[i][k] * mx;
82 }
83 T det = sign ? -1 : 1;
84 for (int i = 0; i < r; ++i)
85 {
86     det = det * m[i][i];
87     det = det / lazy[i];
88     for (auto &j : m[i])
89         j /= lazy[i];
90 }
91 return det;
92 }
93 };

```

9.3 Trie

```

1 // biginter字典數
2 struct BigInteger{
3     static const int BASE = 100000000;
4     static const int WIDTH = 8;
5     vector<int> s;
6     BigInteger(long long num = 0){
7         *this = num;
8     }
9     BigInteger operator = (long long num){
10         s.clear();
11         do{
12             s.push_back(num % BASE);
13             num /= BASE;
14         }while(num > 0);
15         return *this;
16     }
17     BigInteger operator = (const string& str
18     ){
19         s.clear();
20         int x, len = (str.length() - 1) /
21             WIDTH + 1;
22         for(int i = 0; i < len; i++){
23             int end = str.length() - i*WIDTH
24             ;
25             int start = max(0, end-WIDTH);
26             sscanf(str.substr(start, end-
27                 start).c_str(), "%d", &x);
28             s.push_back(x);
29         }
30         return *this;
31     }
32     BigInteger operator + (const BigInteger&
33     b) const{
34         BigInteger c;
35         c.s.clear();
36         for(int i = 0, g = 0; i++){
37             if(g == 0 && i >= s.size() && i
38                 >= b.s.size()) break;
39             int x = g;
40             if(i < s.size()) x+=s[i];
41             if(i < b.s.size()) x+=b.s[i];
42             c.s.push_back(x % BASE);
43             g = x / BASE;
44         }
45         return c;
46     }
47     ostream& operator << (ostream &out, const
48     BigInteger& x){
49         out << x.s.back();
50         for(int i = x.s.size()-2; i >= 0; i--){
51             char buf[20];
52             sprintf(buf, "%08d", x.s[i]);
53             for(int j = 0; j < strlen(buf); j++){
54                 out << buf[j];
55             }
56         }
57         return out;
58     }
59     istream& operator >> (istream &in,
60     BigInteger& x){
61         string s;
62         if(!(in >> s))
63             return in;
64         x = s;
65         return in;
66     }
67 }
68 struct Trie{
69     int c[5000005][10];
70     int val[5000005];
71     int sz;
72     int getIndex(char c){
73         return c - '0';
74     }
75     void init(){
76         memset(c[0], 0, sizeof(c[0]));
77         memset(val, -1, sizeof(val));
78         sz = 1;
79     }
80     void insert(BigInteger x, int v){
81         int u = 0;
82         int max_len_count = 0;
83         int firstNum = x.s.back();
84         char firstBuf[20];
85         sprintf(firstBuf, "%d", firstNum);
86         for(int j = 0; j < strlen(firstBuf);
87             j++){
88             int index = getIndex(firstBuf[j]
89             );
90             if(!c[u][index]){
91                 memset(c[sz], 0, sizeof(c[
92                     sz]));

```

9.4 分數

```

1 typedef long long ll;
2 struct fraction
3 {
4     ll n, d;
5     fraction(const ll &n = 0, const ll &d =
6     1) : n(_n), d(_d)
7     {
8         ll t = __gcd(n, d);
9         n /= t, d /= t;
10        if (d < 0)
11            n = -n, d = -d;
12    }
13    fraction operator-() const
14    {
15        return fraction(-n, d);
16    }
17    fraction operator+(const fraction &b)
18    const

```

TO DO WRITING NOT THINKING

Contents

1 Basic	1	2.7 LIS	2	5.5 Euler circuit	7	6.16 數字乘法組合	10
1.1 A function	1	2.8 LPS	2	5.6 Floyd-warshall	7	6.17 數字加法組合	10
1.2 Codeblock setting	1	2.9 Max_subarray	2	5.7 Hamilton_cycle	7	6.18 羅馬數字	10
1.3 data range	1	2.10 Money problem	2	5.8 Kruskal	7	6.19 質因數分解	10
1.4 IO_fast	1			5.9 Prim	8		
1.5 Python	1	3 Flow & matching	2	5.10 Union_find	8	7 Other	11
2 DP	1	3.1 Dinic	2			7.1 binary search 三類變化	11
2.1 3 維 DP 思路	1	3.2 Edmonds_karp	3	6 Mathematics	8	7.2 heap sort	11
2.2 Knapsack Bounded	1	3.3 hungarian	3	6.1 Catalan	8	7.3 Merge sort	11
2.3 Knapsack sample	1	3.4 Maximum_matching	3	6.2 Combination	8	7.4 Quick	11
2.4 Knapsack Unbounded	1	3.5 MFlow Model	3	6.3 Extended Euclidean	8	7.5 Weighted Job Scheduling	11
2.5 LCIS	1			6.4 Fermat	9	7.6 數獨解法	11
2.6 LCS	1	4 Geometry	4	6.5 Hex to Dec	9		
		4.1 Closest Pair	4	6.6 Log	9	8 String	12
		4.2 Line	4	6.7 Mod	9	8.1 KMP	12
		4.3 Point	4	6.8 Mod 性質	9	8.2 Min Edit Distance	12
		4.4 Polygon	5	6.9 PI	9	8.3 Sliding window	12
		4.5 Triangle	6	6.10 Prime table	9	8.4 Split	12
		5 Graph	6	6.11 Prime 判斷	9		
		5.1 Bellman-Ford	6	6.12 Round(小數)	10	9 data structure	12
		5.2 BFS-queue	6	6.13 二分逼近法	10	9.1 Bigint	12
		5.3 DFS-rec	7	6.14 公式	10	9.2 Matirx	13
		5.4 Dijkstra	7	6.15 四則運算	10	9.3 Trie	14
						9.4 分數	14