

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
7 vector<int>::iterator it = lower_bound(v.
    begin(), v.end(), val);
8 //用binary search找大於或等於val的最小值的位
   置
9 vector<int>::iterator it = upper_bound(v.
    begin(), v.end(), val);
10 //用binary search找大於val的最小值的位置
11
12 /*queue*/
13 queue<datatype> q;
14 front(); /*取出最前面的值(沒有移除掉喔!!)*/
15 back();  /*取出最後面的值(沒有移除掉!!)*/
16 pop();   /*移除最前面的值*/
17 push();  /*新增值到最後面*/
18 empty(); /*回傳bool,檢查是不是空的queue*/
19 size();  /*queue 的大小*/
20
21 /*stack*/
22 stack<datatype> s;
23 top();   /*取出最上面的值(沒有移除掉喔!!)*/
24 pop();   /*移除最上面的值*/
25 push();  /*新增值到最上面*/
26 empty(); /*回傳bool,檢查是不是空的stack*/
27 size();  /*stack 的大小*/
28
29 /*unordered_set*/
30 unordered_set<datatype> s;
31 unordered_set<datatype> s(arr, arr + n);
32 /*initial with array*/
33 insert(); /*插入值*/
34 erase();  /*刪除值*/
35 empty(); /*bool 檢查是不是空*/
36 count(); /*判斷元素存在回傳1 無則回傳0*/

```

1.2 Codeblock setting

```

1 Settings -> Editor -> Keyboard shortcuts ->
2   Plugins -> Source code formatter (AStyle
   )
3 Settings -> Source Formatter -> Padding
4 Delete empty lines within a function or
   method
5 Insert space padding around operators
6 Insert space padding around parentheses on
   outside
7 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 }

```

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[
            i]);
9         for (int k = 1; num > 0; k *= 2)
10        {
11            if (k > num)
12                k = num;
13            num -= k;
14            for (int j = w; j >= weight[i] *
                k; --j)

```

```

15         c[j] = max(c[j], c[j -
            weight[i] * k] + cost[i]
            * k);
16     }
17 }
18 cout << "Max Prince" << c[w];
19 }

```

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

```

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]
                ] + cost[i]);
10    cout << "最高的價值為" << c[w];
11 }

```

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
13            if (arr1[i] == arr2[j])
14                if (current + 1 > table[j])
15                    table[j] = current + 1;
16
17            if (arr1[i] > arr2[j])
18                if (table[j] > current)
19                    current = table[j];
20        }
21    }
22    int result = 0;
23    for (int i = 0; i < m; i++)
24        if (table[i] > result)
25            result = table[i];
26    return result;
27 }

```

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<
        int>(M + 1, 0));
5     for (int i = 1; i <= N; ++i)
6     {
7         for (int j = 1; j <= M; ++j)
8         {
9             if (Ans[i - 1] == num[j - 1])
10                LCS[i][j] = LCS[i - 1][j -
                    1] + 1;
11            else
12                LCS[i][j] = max(LCS[i - 1][j
                    ], LCS[i][j - 1]);
13        }
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],
            LCS[n][m - 1]))
22        {
23            k.push_back(Ans[n - 1]);
24            n--;
25            m--;
26        }
27        else if (LCS[n][m] == LCS[n - 1][m])

```

```

28     n--;
29     else if (LCS[n][m] == LCS[n][m - 1])
30         m--;
31 }
32 reverse(k.begin(), k.end());
33 for (auto i : k)
34     cout << i << " ";
35 cout << endl;
36 return LCS[N][M];
37 }

```

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     {
29         for (int j = 0; j < i; j++)
30             if (LISTbl[j] < LISTbl[i])
31                 LISlen[i] = max(LISlen[i],
32                                 LISlen[j] + 1);
33     }
34     int maxlen = *max_element(LISlen.begin(),
35                               LISlen.end());
36     int num, pos;
37     vector<int> buf;
38     getMaxElementAndPos(LISTbl, LISlen,
39                         numeric_limits<int>::max(), maxlen,
40                         LISTbl.size() - 1, num, pos);
41     buf.push_back(num);
42     for (int len = maxlen - 1; len >= 1; len--)
43     {
44         int tnum = num;
45         int tpos = pos;
46         getMaxElementAndPos(LISTbl, LISlen,
47                             tnum, len, tpos - 1, num, pos);
48         buf.push_back(num);
49     }
50     reverse(buf.begin(), buf.end());

```

```

41 for (int k = 0; k < buf.size(); k++) //
42     列印
43 {
44     if (k == buf.size() - 1)
45         cout << buf[k] << endl;
46     else
47         cout << buf[k] << ",";
48 }
49 return maxlen;

```

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             length
11             x++;
12         x--;
13         if (2 * x + 1 > maxlen)
14         {
15             maxlen = 2 * x + 1;
16             l = i - x;
17             r = i + x;
18         }
19         x = 0;
20         while ((s[i - x] == s[i + 1 + x]) &&
21             (i - x >= 0) && (i + 1 + x < n)) //even length
22             x++;
23         if (2 * x > maxlen)
24         {
25             maxlen = 2 * x;
26             l = i - x + 1;
27             r = i + x;
28         }
29     }
30     cout << maxlen << '\n'; // 最後長度
31     cout << l + 1 << ' ' << r + 1 << '\n';
32     //頭到尾

```

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);

```

```

7 }
8 return global_max;
9 }

```

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 change(vector<int> price, int limit)
28 {
29     vector<int> c(limit + 1, 0);
30     c[0] = true;
31     for (int i = 0; i < price.size(); ++i)
32         for (int j = price[i]; j <= limit;
33             ++j)
34             c[j] = min(c[j], c[j - price[i]]
35                 + 1);
36     cout << c[limit] << '\n';
37 }
38 //湊得某個價位的錢幣用量，有哪幾種可能性
39 void change(vector<int> price, int limit)
40 {
41     vector<int> c(limit + 1, 0);
42     c[0] = true;
43     for (int i = 0; i < price.size(); ++i)
44         for (int j = price[i]; j <= limit;
45             ++j)
46             c[j] |= c[j - price[i]] << 1; //
47             錢幣數量加一，每一種可能性都
48             加一。
49     for (int i = 1; i <= 63; ++i)
50         if (c[i] & (1 << i))
51             cout << "用" << i << "個錢幣可湊
52             得價位" << i << '\n';

```

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[i.v] && edges[i].rest > 0)
22                cut(edges[i].v);
23        }
24    }
25    // min cut end
26    int add_node() {
27        return n++;
28    }
29    void add_edge(int u, int v, long long
30        cap) {
31        edges.push_back({u, v, cap, cap});
32        edges.push_back({v, u, 0, 0});
33        m = edges.size();
34        G[u].push_back(m - 2);
35        G[v].push_back(m - 1);
36    }
37    bool bfs() {
38        fill(d, d + n, -1);
39        queue<int> que;
40        que.push(s); d[s] = 0;
41        while (!que.empty()) {
42            int u = que.front(); que.pop();
43            for (int ei : G[u]) {
44                Edge &e = edges[ei];
45                if (d[e.v] < 0 && e.rest > 0) {
46                    d[e.v] = d[u] + 1;
47                    que.push(e.v);
48                }
49            }
50        }
51        return d[t] >= 0;
52    }
53    long long dfs(int u, long long a) {
54        if (u == t || a == 0) return a;
55        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)) )
            flow += mf;
75     }
76     return flow;
77 }
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<
8          int>(n+1, 0)); //residual network
9      while(true){
10         vector<int> bottleneck(n+1, 0);
11         bottleneck[s] = inf;
12         queue<int> q;
13         q.push(s);
14         vector<int> pre(n+1, 0);
15         while(!q.empty() && bottleneck[t] == 0){
16             int cur = q.front();
17             q.pop();
18             for(int i = 1; i <= n ; i++){
19                 if(bottleneck[i] == 0 && capacity[
20                     cur][i] > residual[cur][i]){
21                     q.push(i);
22                     pre[i] = cur;
23                     bottleneck[i] = min(bottleneck[cur]
24                         , capacity[cur][i] - residual
25                         [cur][i]);
26                 }
27             }
28         }
29         if(bottleneck[t] == 0) break;
30         for(int cur = t; cur != s; cur = pre[cur]
31             ){
32             residual[pre[cur]][cur] +=
33                 bottleneck[t];
34         }
35     }
36     return ans;
37 }

```

```

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)
38             break;
39         vector<vector<int>> capacity(n+1, vector<
40             int>(n+1, 0));
41         int s, t, c;
42         cin >> s >> t >> c;
43         int a, b, bandwidth;
44         for(int i = 0 ; i < c ; ++i){
45             cin >> a >> b >> bandwidth;
46             capacity[a][b] += bandwidth;
47             capacity[b][a] += bandwidth;
48         }
49         cout << "Network " << testcase++ << endl;
50         ;
51         cout << "The bandwidth is " <<
52             getMaxFlow(capacity, s, t, n) << "."
53             << endl;
54         cout << endl;
55     }
56     return 0;
57 }

```

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     }
11     bool dfs(int u){
12         for (int v:G[u]){
13             if (vis[v]) continue;
14             vis[v]=true;
15             if (match[v]==-1 || dfs(match[v]
16                 )){
17                 match[v] = u;
18                 match[u] = v;
19                 return true;
20             }
21         }
22         return false;
23     }
24     int solve(){
25         int res = 0;
26         memset(match, -1, sizeof(match));
27         for (int i=0; i<n; i++){
28             if (match[i]==-1){
29                 dfs(i);
30             }
31         }
32         return res;
33     }
34 }

```

```

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<
4      bool> pass){
5      for (int i = 0; i < res[0].size(); i++){
6          if(res[node][i] && !pass[i]){
7              pass[i] = true;
8              if(y[i] == -1 || dfs(res, y[i], x,
9                  y, pass)){
10                 x[node] = i;
11                 y[i] = node;
12                 return true;
13             }
14         }
15     }
16     return false;
17 }
18 int main(){
19     int n, m, l;
20     while(cin>>n>>m>>l){
21         vector<vector<bool>> res(n, vector<
22             bool>(m, false));
23         for (int i = 0; i < l; i++){
24             int a, b;
25             cin >> a >> b;
26             res[a][b] = true;
27         }
28         int ans = 0;
29         vector<int> x(n, -1);
30         vector<int> y(m, -1);
31         for (int i = 0; i < n; i++){
32             vector<bool> pass(m, false);
33             if(dfs(res, i, x, y, pass))
34                 ans += 1;
35         }
36         cout << ans << endl;
37     }
38     return 0;
39 }
40 /*
41 4 3 5 //n matching m, l links
42 0 0
43 0 2
44 1 0
45 2 1
46 3 1
47 answer is 3
48 */

```

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]
50                         ] + 1;
51                     que[++tail] = v[i];
52                 }
53             }
54             ++head;
55         }
56         return vi[t] == tim;
57     }
58     ll dfs(int x, ll a)
59     {
60         if (x == t || a == 0)
61             return a;
62         ll flw = 0, f;

```

```

60 int &i = cur[x];
61 for (i = first[x]; i != -1; i = next
62 [i])
63 {
64     if (dis[x] + 1 == dis[v[i]] && (
65         f = dfs(v[i], min(a, cap[i]
66             - flow[i])) > 0)
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);

```

```

15 for (int i = m + 1; i <= r && fabs(p[m].
16     x - p[i].x) < d; ++i)
17     vec.push_back(i);
18 sort(vec.begin(), vec.end(), [&](int a,
19     int b)
20 { return p[a].y < p[b].y; });
21 for (int i = 0; i < vec.size(); ++i)
22     for (int j = i + 1; j < vec.size()
23         && fabs(p[vec[j]].y - p[vec[i]].
24             y) < d; ++j)
25         d = min(d, dist(p[vec[i]], p[vec
26             [j]]));
27 return d;

```

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> &
8         y) : p1(x), p2(y) {}
9     void pton()
10    { //轉成一般式
11        a = p1.y - p2.y;
12        b = p2.x - p1.x;
13        c = -a * p1.x - b * p1.y;
14    }
15    T ori(const point<T> &p) const
16    { //點和有向直線的關係 · >0左邊、=0在線上
17        <0右邊
18        return (p2 - p1).cross(p - p1);
19    }
20    T btw(const point<T> &p) const
21    { //點投影落在線段上<=0
22        return (p1 - p).dot(p2 - p);
23    }
24    bool point_on_segment(const point<T> &p)
25        const
26    { //點是否在線段上
27        return ori(p) == 0 && btw(p) <= 0;
28    }
29    T dis2(const point<T> &p, bool
30        is_segment = 0) const
31    { //點跟直線/線段的距離平方
32        point<T> v = p2 - p1, v1 = p - p1;
33        if (is_segment)
34        {
35            point<T> v2 = p - p2;
36            if (v.dot(v1) <= 0)
37                return v1.abs2();
38            if (v.dot(v2) >= 0)
39                return v2.abs2();
40        }
41        T tmp = v.cross(v1);
42        return tmp * tmp / v.abs2();
43    }
44    T seg_dis2(const line<T> &l) const
45    { //兩線段距離平方

```

```

46 return min({dis2(l.p1, 1), dis2(l.p2
47     , 1), l.dis2(p1, 1), l.dis2(p2,
48     1)});
49 }
50 point<T> projection(const point<T> &p)
51 const
52 { //點對直線的投影
53     point<T> n = (p2 - p1).normal();
54     return p - n * (p - p1).dot(n) / n.
55         abs2();
56 }
57 point<T> mirror(const point<T> &p) const
58 {
59     //點對直線的鏡射 · 要先呼叫pton轉成一般式
60     point<T> R;
61     T d = a * a + b * b;
62     R.x = (b * b * p.x - a * a * p.x - 2
63         * a * b * p.y - 2 * a * c) / d;
64     R.y = (a * a * p.y - b * b * p.y - 2
65         * a * b * p.x - 2 * b * c) / d;
66     return R;
67 }
68 bool equal(const line &l) const
69 { //直線相等
70     return ori(l.p1) == 0 && ori(l.p2)
71         == 0;
72 }
73 bool parallel(const line &l) const
74 {
75     return (p1 - p2).cross(l.p1 - l.p2)
76         == 0;
77 }
78 bool cross_seg(const line &l) const
79 {
80     return (p2 - p1).cross(l.p1 - p1) *
81         (p2 - p1).cross(l.p2 - p1) <= 0;
82     //直線是否交線段
83 }
84 int line_intersect(const line &l) const
85 { //直線相交情況 · -1無限多點、1交於一
86     點、0不相交
87     return parallel(l) ? (ori(l.p1) == 0
88         ? -1 : 0) : 1;
89 }
90 int seg_intersect(const line &l) const
91 {
92     T c1 = ori(l.p1), c2 = ori(l.p2);
93     T c3 = l.ori(p1), c4 = l.ori(p2);
94     if (c1 == 0 && c2 == 0)
95     { //共線
96         bool b1 = btw(l.p1) >= 0, b2 =
97             btw(l.p2) >= 0;
98         T a3 = l.btw(p1), a4 = l.btw(p2);
99         ;
100         if (b1 && b2 && a3 == 0 && a4 >=
101             0)
102             return 2;
103         if (b1 && b2 && a3 >= 0 && a4 ==
104             0)
105             return 3;
106         if (b1 && b2 && a3 >= 0 && a4 >=
107             0)
108             return 0;
109     }

```

```

88 return -1; //無限交點
89 }
90 else if (c1 * c2 <= 0 && c3 * c4 <=
91     0)
92     return 1;
93 return 0; //不相交
94 }
95 point<T> line_intersection(const line &l1
96     ) const
97 { /*直線交點*/
98     point<T> a = p2 - p1, b = l.p2 - l.
99         p1, s = l.p1 - p1;
100     //if(a.cross(b)==0)return INF;
101     return p1 + a * (s.cross(b) / a.
102         cross(b));
103 }
104 point<T> seg_intersection(const line &l1)
105     const
106 { //線段交點
107     int res = seg_intersect(l1);
108     if (res <= 0)
109         assert(0);
110     if (res == 2)
111         return p1;
112     if (res == 3)
113         return p2;
114     return line_intersection(l1);
115 }

```

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

```

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 has(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;

```

```

28     for (int i = 0, j = p.size() - 1; i
29         < p.size(); j = i++)
30         if (line<T>(p[i], p[j]).
31             point_on_segment(t))
32             return -1;
33         else if ((p[i].y > t.y) != (p[j]
34             .y > t.y) &&
35             t.x < (p[j].x - p[i].x) /
36                 * (t.y - p[i].y) /
37                 (p[j].y - p[i].y)
38                 + p[i].x)
39             c = !c;
40     return c;
41 }
42 char point_in_convex(const point<T> &x)
43 const
44 {
45     int l = 1, r = (int)p.size() - 2;
46     while (l <= r)
47     { //點是否在凸多邊形內，是的話回傳1
48         //在邊上回傳-1、否則回傳0
49         int mid = (l + r) / 2;
50         T a1 = (p[mid] - p[0]).cross(x -
51             p[0]);
52         T a2 = (p[mid + 1] - p[0]).cross
53             (x - p[0]);
54         if (a1 >= 0 && a2 <= 0)
55         {
56             T res = (p[mid + 1] - p[mid]
57                 ).cross(x - p[mid]);
58             return res > 0 ? 1 : (res >=
59                 0 ? -1 : 0);
60         }
61         else if (a1 < 0)
62             r = mid - 1;
63         else
64             l = mid + 1;
65     }
66     return 0;
67 }
68 vector<T> getA() const
69 { //凸包邊對x軸的夾角
70     vector<T> res; //一定是遞增的
71     for (size_t i = 0; i < p.size(); ++i
72         )
73         res.push_back((p[(i + 1) % p.
74             size()] - p[i]).getA());
75     return res;
76 }
77 bool line_intersect(const vector<T> &A,
78     const line<T> &l) const
79 { //O(logN)
80     int f1 = upper_bound(A.begin(), A.
81         end(), (l.p1 - l.p2).getA()) - A
82         .begin();
83     int f2 = upper_bound(A.begin(), A.
84         end(), (l.p2 - l.p1).getA()) - A
85         .begin();
86     return l.cross_seg(line<T>(p[f1], p[
87         f2]));
88 }
89 polygon cut(const line<T> &l) const
90 { //凸包對直線切割，得到直線l左側的凸包
91     polygon ans;

```

```

92     for (int n = p.size(), i = n - 1, j
93         = 0; j < n; i = j++)
94     {
95         if (l.ori(p[i]) >= 0)
96         {
97             ans.p.push_back(p[i]);
98             if (l.ori(p[j]) < 0)
99                 ans.p.push_back(l.
100                     line_intersection(
101                         line<T>(p[i], p[j]),
102                         l));
103         }
104         else if (l.ori(p[j]) > 0)
105             ans.p.push_back(l.
106                 line_intersection(line<T>
107                     (p[i], p[j]), l));
108     }
109     return ans;
110 }
111 static bool graham_cmp(const point<T> &a
112     , const point<T> &b)
113 { //凸包排序函數 // 起始點不同
114     // return (a.x < b.x) || (a.x == b.x
115         && a.y < b.y); //最左下角開始
116     return (a.y < b.y) || (a.y == b.y &&
117         a.x < b.x); //Y最小開始
118 }
119 void graham(vector<point<T>> &s)
120 { //凸包 Convexhull 2D
121     sort(s.begin(), s.end(), graham_cmp)
122     ;
123     p.resize(s.size() + 1);
124     int m = 0;
125     // cross >= 0 順時針，cross <= 0 逆
126     // 時針旋轉
127     for (size_t i = 0; i < s.size(); ++i
128         )
129     {
130         while (m >= 2 && (p[m - 1] - p[m]
131             - 2]).cross(s[i] - p[m -
132                 2]) <= 0)
133             --m;
134         p[m++] = s[i];
135     }
136     for (int i = s.size() - 2, t = m +
137         1; i >= 0; --i)
138     {
139         while (m >= t && (p[m - 1] - p[m]
140             - 2]).cross(s[i] - p[m -
141                 2]) <= 0)
142             --m;
143         p[m++] = s[i];
144     }
145     if (s.size() > 1) // 重複頭一次需扣
146         掉
147         p.resize(m);
148 }
149 T diam()
150 { //直徑
151     int n = p.size(), t = 1;
152     T ans = 0;
153     p.push_back(p[0]);
154     for (int i = 0; i < n; i++)

```

```

155     {
156         point<T> now = p[i + 1] - p[i];
157         while (now.cross(p[t + 1] - p[i]
158             ]) > now.cross(p[t] - p[i]))
159             t = (t + 1) % n;
160         ans = max(ans, (p[i] - p[t]).
161             abs2());
162     }
163     return p.pop_back(), ans;
164 }
165 T min_cover_rectangle()
166 { //最小覆蓋矩形
167     int n = p.size(), t = 1, r = 1, l;
168     if (n < 3)
169         return 0; //也可以做最小周長矩形
170     T ans = 1e99;
171     p.push_back(p[0]);
172     for (int i = 0; i < n; i++)
173     {
174         point<T> now = p[i + 1] - p[i];
175         while (now.cross(p[t + 1] - p[i]
176             ]) > now.cross(p[t] - p[i]))
177             t = (t + 1) % n;
178         while (now.dot(p[r + 1] - p[i])
179             > now.dot(p[r] - p[i]))
180             r = (r + 1) % n;
181         if (!i)
182             l = r;
183         while (now.dot(p[l + 1] - p[i])
184             <= now.dot(p[l] - p[i]))
185             l = (l + 1) % n;
186         T d = now.abs2();
187         T tmp = now.cross(p[t] - p[i]) *
188             (now.dot(p[r] - p[i]) - now
189                 .dot(p[l] - p[i])) / d;
190         ans = min(ans, tmp);
191     }
192     return p.pop_back(), ans;
193 }
194 T dis2(polygon &p1)
195 { //凸包最近距離平方
196     vector<point<T>> &P = p, &Q = p1.p;
197     int n = P.size(), m = Q.size(), l =
198         0, r = 0;
199     for (int i = 0; i < n; ++i)
200         if (P[i].y < P[l].y)
201             l = i;
202     for (int i = 0; i < m; ++i)
203         if (Q[i].y < Q[r].y)
204             r = i;
205     P.push_back(P[0]), Q.push_back(Q[0])
206     ;
207     T ans = 1e99;
208     for (int i = 0; i < n; ++i)
209     {
210         while ((P[l] - P[l + 1]).cross(Q
211             [r + 1] - Q[r]) < 0)
212             r = (r + 1) % m;
213         ans = min(ans, line<T>(P[l], P[l
214             + 1]).seg_dis2(line<T>(Q[r]
215                 , Q[r + 1])));
216         l = (l + 1) % n;
217     }
218     return P.pop_back(), Q.pop_back(),
219         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,
176                       const line<T> &B)
177 {
178     point<T> a = A.p2 - A.p1, b = B.p2 -
179     B.p1;
180     return sign(a) < sign(b) || (sign(a)
181     == sign(b) && a.cross(b) > 0);
182 }
183 int halfplane_intersection(vector<line<T>
184 >> &s)
185 { //半平面交
186     sort(s.begin(), s.end(), angle_cmp);
187     //線段左側為該線段半平面
188     int L, R, n = s.size();
189     vector<point<T>> px(n);
190     vector<line<T>> q(n);
191     q[L = R = 0] = s[0];
192     for (int i = 1; i < n; ++i)
193     {
194         while (L < R && s[i].ori(px[R -
195         1]) <= 0)
196             --R;
197         while (L < R && s[i].ori(px[L])
198             <= 0)
199             ++L;
200         q[++R] = s[i];
201         if (q[R].parallel(q[R - 1]))
202         {
203             --R;
204             if (q[R].ori(s[i].p1) > 0)
205                 q[R] = s[i];
206         }
207         if (L < R)
208             px[R - 1] = q[R - 1].
209             line_intersection(q[R]);
210     }
211     while (L < R && q[L].ori(px[R - 1])
212         <= 0)
213         --R;
214     p.clear();
215     if (R - L <= 1)
216         return 0;
217     px[R] = q[R].line_intersection(q[L]);
218     for (int i = L; i <= R; ++i)
219         p.push_back(px[i]);
220     return R - L + 1;
221 }
222 };

```

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<
7     T> &b, const point<T> &c) : a(a), b(
8     b), c(c) {}
9     T area() const
10    {
11        T t = (b - a).cross(c - a) / 2;
12        return t > 0 ? t : -t;
13    }
14    point<T> barycenter() const
15    { //重心
16        return (a + b + c) / 3;
17    }
18    point<T> circumcenter() const
19    { //外心
20        static line<T> u, v;
21        u.p1 = (a + b) / 2;
22        u.p2 = point<T>(u.p1.x - a.y + b.y,
23        u.p1.y + a.x - b.x);
24        v.p1 = (a + c) / 2;
25        v.p2 = point<T>(v.p1.x - a.y + c.y,
26        v.p1.y + a.x - c.x);
27        return u.line_intersection(v);
28    }
29    point<T> incenter() const
30    { //內心
31        T A = sqrt((b - c).abs2()), B = sqrt(
32        ((a - c).abs2()), C = sqrt((a -
33        b).abs2());
34        return point<T>(A * a.x + B * b.x +
35        C * c.x, A * a.y + B * b.y + C *
36        c.y) / (A + B + C);
37    }
38    point<T> perpcenter() const
39    { //垂心
40        return barycenter() * 3 -
41        circumcenter() * 2;
42    }
43 };

```

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> dist;
6 vector<int> ancestor;
7 void BellmanFord(int start, int node){
8     dist[start] = 0;
9     for(int it = 0; it < node-1; it++){
10         for(int i = 0; i < node; i++){
11             for(int j = 0; j < node; j++){
12                 if(edges[i][j] != -1){
13                     if(dist[i] + edges[i][j]
14                     < dist[j]){
15                         dist[j] = dist[i] +
16                         edges[i][j];
17                         ancestor[j] = i;
18                     }
19                 }
20             }
21         }
22     }
23 }

```

```

24     }
25     }
26     }
27     }
28     }
29     }
30     }
31     }
32     }
33     }
34     }
35     }
36     }
37     }
38     }
39     }
40     }
41     }
42     }
43     }
44     }
45     }
46     }
47     }
48     }
49     }
50     }
51     }
52     }
53     }
54     }
55     }
56     }
57     }
58     }
59     }
60     }
61     }
62     }
63     }
64     }
65     }
66     }
67     }
68     }
69     }
70     }
71     }
72     }
73     }
74     }
75     }
76     }
77     }
78     }
79     }
80     }
81     }
82     }
83     }
84     }
85     }
86     }
87     }
88     }
89     }
90     }
91     }
92     }
93     }
94     }
95     }
96     }
97     }
98     }
99     }
100    }

```

5.2 BFS-queue

```

1 /*BFS - queue version*/
2 void BFS(vector<int> &result, vector<pair<
3 int, int>> edges, int node, int start)
4 {
5     vector<int> pass(node, 0);
6     queue<int> q;
7     queue<int> p;
8     q.push(start);
9     int count = 1;
10    vector<pair<int, int>> newedges;
11    while (!q.empty())
12    {
13        pass[q.front()] = 1;
14        for (int i = 0; i < edges.size(); i
15        ++){
16            if (edges[i].first == q.front()
17            && pass[edges[i].second] == 0)
18            {
19                p.push(edges[i].second);
20                result[edges[i].second] =
21                count;
22            }
23        }
24    }
25 }

```

```

26     }
27     }
28     }
29     }
30     }
31     }
32     }
33     }
34     }
35     }
36     }
37     }
38     }
39     }
40     }
41     }
42     }
43     }
44     }
45     }
46     }
47     }
48     }
49     }
50     }
51     }
52     }
53     }
54     }
55     }
56     }
57     }
58     }
59     }
60     }
61     }
62     }
63     }
64     }
65     }
66     }
67     }
68     }
69     }
70     }
71     }
72     }
73     }
74     }
75     }
76     }
77     }
78     }
79     }
80     }
81     }
82     }
83     }
84     }
85     }
86     }
87     }
88     }
89     }
90     }
91     }
92     }
93     }
94     }
95     }
96     }
97     }
98     }
99     }
100    }

```

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 }

```

```

13     else if((*iter).first.second ==
14         start && (*iter).second == 0 &&
15         pass[(*iter).first.first] == 0){
16         route.push_back((*iter).first.
17             first);
18         DFS((*iter).first.first);
19     }
20 }
21 int main(){
22     int node;
23     cin>>node;
24     pass.resize(node,0);
25     int a,b;
26     while(cin>>a>>b){
27         if(a == -1 && b == -1)
28             break;
29         edges.insert(pair<pair<int,int>,int>
30             >(pair<int,int>(a,b),0));
31     }
32     int start;
33     cin>>start;
34     route.push_back(start);
35     DFS(start);
36     return 0;
37 }

```

5.4 Dijkstra

```

1 /*SPA - Dijkstra*/
2 #define inf INT_MAX
3 vector<vector<int>> weight;
4 vector<int> ancestor;
5 vector<int> dist;
6 void dijkstra(int start){
7     priority_queue<pair<int,int>, vector<
8         pair<int,int>, >, greater<pair<int,
9             int>>> pq;
10     pq.push(make_pair(0,start));
11     while(!pq.empty()){
12         int cur = pq.top().second;
13         pq.pop();
14         for(int i = 0; i < weight[cur].size()
15             ; i++){
16             if(dist[i] > dist[cur] + weight[
17                 cur][i] && weight[cur][i] !=
18                 -1){
19                 dist[i] = dist[cur] + weight
20                     [cur][i];
21                 ancestor[i] = cur;
22                 pq.push(make_pair(dist[i],i)
23                     );
24             }
25         }
26     }
27 }
28 int main(){
29     int node;
30     cin>>node;
31     int a,b,d;
32     weight.resize(node,vector<int>(node,-1))
33     ;
34     while(cin>>a>>b>>d){

```

```

27     /*input: source destination weight*/
28     if(a == -1 && b == -1 && d == -1)
29         break;
30     weight[a][b] = d;
31 }
32 ancestor.resize(node,-1);
33 dist.resize(node,inf);
34 int start;
35 cin>>start;
36 dist[start] = 0;
37 dijkstra(start);
38 return 0;
39 }

```

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++){
7         for (int i = 0; i < n; i++){
8             for (int j = 0; j < n; j++){
9                 if(distance[i][k] + distance
10                     [k][j] < distance[i][j])
11                     distance[i][j] =
12                         distance[i][k] +
13                         distance[k][j];
14                 ancestor[i][j] =
15                     ancestor[k][j];
16             }
17         }
18     }
19 }
20 int main(){
21     int n;
22     cin >> n;

```

```

18 int a, b, d;
19 vector<vector<int>> distance(n, vector<
20     int>(n,99999));
21 vector<vector<int>> ancestor(n, vector<
22     int>(n,-1));
23 while(cin>>a>>b>>d){
24     if(a == -1 && b == -1 && d == -1)
25         break;
26     distance[a][b] = d;
27     ancestor[a][b] = a;
28 }
29 for (int i = 0; i < n; i++)
30     distance[i][i] = 0;
31 floyd_warshall(distance, ancestor, n);
32 /*Negative cycle detection*/
33 for (int i = 0; i < n; i++){
34     if(distance[i][i] < 0){
35         cout << "Negative cycle!" <<
36             endl;
37         break;
38     }
39 }
40 return 0;
41 }

```

5.7 Hamilton_cycle

```

1 /*find hamilton cycle*/
2 void hamilton(vector<vector<int>> gp, int k,
3     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++){
19         if(gp[cur][i] == 1 && !pass[i]){
20             pass[i] = true;
21             solution[i] = cur;
22             hamilton(gp, k + 1, i, solution,
23                 pass,flag);
24             pass[i] = false;
25         }
26     }
27 }
28 int main(){
29     int n;
30     while(cin>>n){
31         int a,b;
32         bool end = false;
33         vector<vector<int>> gp(n+1,vector<
34             int>(n+1,0));
35         while(cin>>a>>b){
36             if(a == 0 && b == 0)

```

```

32         break;
33         gp[a][b] = 1;
34         gp[b][a] = 1;
35     }
36     vector<int> solution(n + 1, -1);
37     vector<bool> pass(n + 1, false);
38     solution[1] = 0;
39     pass[1] = true;
40     bool flag = false;
41     hamilton(gp, 1,1 ,solution,pass,flag
42         );
43     if(!flag)
44         cout << "N" << endl;
45 }
46 return 0;
47 /*
48 4
49 1 2
50 2 3
51 2 4
52 3 4
53 3 1
54 0 0
55 output: 1 3 4 2 1
56 */

```

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 int find(int x,vector<int>& union_set){
12     if(x != union_set[x])
13         union_set[x] = find(union_set[x],
14             union_set);
15     return union_set[x];
16 }
17 void merge(int a,int b,vector<int>&
18     union_set){
19     int pa = find(a, union_set);
20     int pb = find(b, union_set);
21     if(pa != pb)
22         union_set[pa] = pb;
23 }
24 void kruskal(priority_queue<edges> pq,int n)
25 {
26     vector<int> union_set(n, 0);
27     for (int i = 0; i < n; i++)
28         union_set[i] = i;
29     int edge = 0;
30     int cost = 0; //evaluate cost of mst
31     while(!pq.empty() && edge < n - 1){
32         edges cur = pq.top();
33         int from = find(cur.from, union_set)
34             ;

```

```

30     int to = find(cur.to, union_set);
31     if(from != to){
32         merge(from, to, union_set);
33         edge += 1;
34         cost += cur.weight;
35     }
36     pq.pop();
37 }
38 if(edge < n-1)
39     cout << "No mst" << endl;
40 else
41     cout << cost << endl;
42 }
43 int main(){
44     int n;
45     cin >> n;
46     int a, b, d;
47     priority_queue<edges> pq;
48     while(cin>>a>>b>>d){
49         if(a == -1 && b == -1 && d == -1)
50             break;
51         edges tmp;
52         tmp.from = a;
53         tmp.to = b;
54         tmp.weight = d;
55         pq.push(tmp);
56     }
57     kruskal(pq, n);
58     return 0;
59 }

```

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 };
12 void Prim(vector<vector<int>> gp, int n, int
13 start){
14     vector<bool> pass(n, false);
15     int edge = 0;
16     int cost = 0; //evaluate cost of mst
17     priority_queue<edges> pq;
18     for (int i = 0; i < n; i++){
19         if(gp[start][i] != inf){
20             edges tmp;
21             tmp.from = start;
22             tmp.to = i;
23             tmp.weight = gp[start][i];
24             pq.push(tmp);
25         }
26     }
27     pass[start] = true;
28     while(!pq.empty() && edge < n-1){
29         edges cur = pq.top();
30         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,
54 inf));
55     while(cin>>a>>b>>d){
56         if(a == -1 && b == -1 && d == -1)
57             break;
58         if(gp[a][b] > d)
59             gp[a][b] = d;
60     }
61     Prim(gp, n, 0);
62     return 0;

```

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],
5 union_set); //compress path
6      return union_set[x];
7 }
8 void merge(int x, int y, vector<int> &
9 union_set, vector<int> &rank)
10 {
11     int rx, ry;
12     rx = find(x, union_set);
13     ry = find(y, union_set);
14     if (rx == ry)
15         return;
16     /*merge by rank -> always merge small
17 tree to big tree*/
18     if (rank[rx] > rank[ry])
19         union_set[ry] = rx;
20     else
21     {
22         union_set[rx] = ry;
23         if (rank[rx] == rank[ry])
24             ++rank[ry];

```

```

22     }
23 }
24 int main()
25 {
26     int node;
27     cin >> node; //Input Node number
28     vector<int> union_set(node, 0);
29     vector<int> rank(node, 0);
30     for (int i = 0; i < node; i++)
31         union_set[i] = i;
32     int edge;
33     cin >> edge; //Input Edge number
34     for (int i = 0; i < edge; i++)
35     {
36         int a, b;
37         cin >> a >> b;
38         merge(a, b, union_set, rank);
39     }
40     /*build party*/
41     vector<vector<int>> party(node, vector<
42 int>(0));
43     for (int i = 0; i < node; i++)
44         party[find(i, union_set)].push_back(
45 i);

```

6 Mathematics

6.1 Combination

```

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

```

6.2 Extended Euclidean

```

1  // ax + by = gcd(a,b)
2  pair<long long, long long> extgcd(long long
3 a, long long b)
4 {
5     if (b == 0)
6         return {1, 0};
7     long long k = a / b;
8     pair<long long, long long> p = extgcd(b,
9 a - k * b);
10    //cout << p.first << " " << p.second <<
11 endl;
12    //cout << "商數(k)= " << k << endl <<
13 endl;
14    return {p.second, p.first - k * p.second
15 };

```

```

11 }
12 }
13 int main()
14 {
15     int a, b;
16     cin >> a >> b;
17     pair<long long, long long> xy = extgcd(a
18 , b); //(x0,y0)
19     cout << xy.first << " " << xy.second <<
20 endl;
21     cout << xy.first << " * " << a << " + "
22 << xy.second << " * " << b << endl;
23     return 0;
24 }
25 // ax + by = gcd(a,b) * r
26 /*find |x|+|y| -> min*/
27 int main()
28 {
29     long long r, p, q; /*px+qy = r*/
30     int cases;
31     cin >> cases;
32     while (cases--)
33     {
34         cin >> r >> p >> q;
35         pair<long long, long long> xy =
36 extgcd(q, p); //(x0,y0)
37         long long ans = 0, tmp = 0;
38         double k, k1;
39         long long s, s1;
40         k = 1 - (double)(r * xy.first) / p;
41         s = round(k);
42         ans = llabs(r * xy.first + s * p) +
43 llabs(r * xy.second - s * q);
44         k1 = -(double)(r * xy.first) / p;
45         s1 = round(k1);
46         /*cout << k << endl << k1 << endl;
47 cout << s << endl << s1 << endl;
48 */
49         tmp = llabs(r * xy.first + s1 * p) +
50 llabs(r * xy.second - s1 * q);
51         ans = min(ans, tmp);
52     }
53     cout << ans << endl;
54     return 0;
55 }

```

6.3 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      {
8          if (num[i] == '0' && num[i] == '9')
9          {
10             temp += (num[i] - 48) * base;
11             base = base * 16;
12         }
13         else if (num[i] == 'A' && num[i] == 'F'
14 ')

```



```

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

```

6.4 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 }

```

6.5 Mod

```

1 int pow_mod(int a, int n, int m) // a ^ n
2     mod m;
3 { // a, n, m < 10 ^ 9
4     if (n == 0)
5         return 1;
6     int x = pow_mid(a, n / 2, m);
7     long long ans = (long long)x * x % m;
8     if (n % 2 == 1)
9         ans = ans * a % m;
10    return (int)ans;
11 }
12 //****基本運算****/
13 加法: (a + b) % p = (a % p + b % p) % p;
14 減法: (a - b) % p = (a % p - b % p + p) % p;
15 乘法: (a * b) % p = (a % p * b % p) % p;
16 次方: (a ^ b) % p = ((a % p) ^ b) % p;
17 加法結合律: ((a + b) % p + c) % p = (a + (b
18 + c)) % p;
19 乘法結合律: ((a * b) % p * c) % p = (a * (b
20 * c)) % p;
21 加法交換律: (a + b) % p = (b + a) % p;
22 乘法交換律: (a * b) % p = (b * a) % p;
23 結合律: ((a + b) % p * c) = ((a * c) % p + (
24 b * c) % p) % p;
25 //****同餘****/
26 如果 a ≡ b(mod m) · 我們會說 a,b 在模 m 下同
27 餘。
28 整除性: a ≡ b(mod m) ⇔ c | m = a - b, c | Z
29 ⇔ a ≡ b (mod m) ⇔ m | a-b

```

```

24 遞移性: 若 a ≡ b (mod c), b ≡ d(mod c) 則 a
25     ≡ d (mod c)
26 a ≡ b (mod m) ⇔ { a ± c ≡ b ± d (mod m) }
27 c ≡ d (mod m) ⇔ { a * c ≡ b * d (mod m) }
28 放大縮小模數: k ∈ Z+, a ≡ b (mod m) ⇔ k * a
29     ≡ k * b (mod k*m)
30 //****費瑪定理****/
31 假如 a 是一個整數 · p 是一個質數 · 且 a, p 互
32     質
33 a ^ (p - 1) ≡ 1 mod P 如果 gcd(a,p) = 1 且
34     p 為質數
35 得 a * a ^ (p - 2) = 1 (a ^ -1 = a ^ (p - 2)
36     )

```

6.6 Permutation

```

1 // 全排列要先 sort !!!
2 // num -> vector or string
3 next_permutation(num.begin(), num.end());
4 prev_permutation(num.begin(), num.end());

```

6.7 PI

```

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

```

6.8 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();
13             ++j)
14         {
15             if (i * p[j] > maxn)
16                 break;
17             is_notp[i * p[j]] = 1;
18             if (i % p[j] == 0)
19                 break;
20         }
21     }
22 }

```

6.9 primeBOOL

```

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;
39     for (int i = 0; i < t; ++i)
40     {
41         a = fmul(a, a, n);
42         if (a == 1)
43             return false;
44         if (a == n - 1)
45             return true;
46     }
47     return false;
48 }
49 bool is_prime(long long n)
50 {
51     if (n < 2)
52         return false;
53     if (n % 2 == 0)
54         return n == 2;
55     long long u = n - 1;
56     int t = 0;
57     for (; u & 1; u >>= 1, ++t)
58         ;
59     for (long long i : chk)
60     {
61         if (!check(i, u, n, t))
62             return false;
63     }
64 }

```

```

59     }
60     return true;
61 }
62 // if (is_prime(int num)) // true == prime
63 // 反之亦然

```

6.10 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));

```

6.11 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 }

```

6.12 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 +
14                1, ri);
15    }
16 }

```

```

13     if (s[i] == '-' && c == 0)
14         return DFS(le, i - 1) - DFS(i + 1, ri);
15 }
16 for (int i = ri; i >= le; i--)
17 {
18     if (s[i] == '(')
19         c++;
20     if (s[i] == '(')
21         c--;
22     if (s[i] == '*' && c == 0)
23         return DFS(le, i - 1) * DFS(i + 1, ri);
24     if (s[i] == '/' && c == 0)
25         return DFS(le, i - 1) / DFS(i + 1, ri);
26     if (s[i] == '%' && c == 0)
27         return DFS(le, i - 1) % DFS(i + 1, ri);
28 }
29 if ((s[le] == '(' && (s[ri] == ')'))
30     return DFS(le + 1, ri - 1); // 去除括號
31 if (s[le] == ' ' && s[ri] == ' ')
32     return DFS(le + 1, ri - 1); // 去除左右兩邊空格
33 if (s[le] == '(')
34     return DFS(le + 1, ri); // 去除左邊空格
35 if (s[ri] == ')')
36     return DFS(le, ri - 1); // 去除右邊空格
37 long long int num = 0;
38 for (int i = le; i <= ri; i++)
39     num = num * 10 + s[i] - '0';
40 return num;
41 }

```

6.13 數字乘法組合

```

1 void dfs(int j, int old, int num, vector<int>
  > com, vector<vector<int>> &ans)
2 {
3     for (int i = j; i <= sqrt(num); i++)
4     {
5         if (old == num)
6             com.clear();
7         if (num % i == 0)
8         {
9             vector<int> a;
10            a = com;
11            a.push_back(i);
12            finds(i, old, num / i, a, ans);
13            a.push_back(num / i);
14            ans.push_back(a);
15        }
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] << endl;
27 }

```

6.14 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
  &out, vector<vector<int>> &ans)
2 {
3     if (n == 0)
4     {
5         for (int i : out)
6             if (i > m)
7                 return;
8         ans.push_back(out);
9     }
10    for (int j = i; j <= n; j++)
11    {
12        out.push_back(j);
13        recur(j, n - j, m, out, ans);
14        out.pop_back();
15    }
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] << endl;
26 }

```

6.15 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 }

```

6.16 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] >
7         array[root])
8         largest = left;
9     else
10        largest = root;
11    if (right <= length && array[right] >
12        array[largest])
13        largest = right;
14    if (largest != root)
15    {
16        swap(array[largest], array[root]);
17        MaxHeapify(array, largest, length);
18    }
19 }
20 void HeapSort(vector<int> &array)
21 {
22     array.insert(array.begin(), 0);
23     for (int i = (int)array.size() / 2; i >= 1; i--)
24         MaxHeapify(array, i, (int)array.size() - 1);
25     int size = (int)array.size() - 1;
26     for (int i = (int)array.size() - 1; i >= 2; i--)
27     {
28         swap(array[1], array[i]);
29         size--;
30         MaxHeapify(array, 1, size);
31     }
32     array.erase(array.begin());

```

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
9     for (int i = front; i <= end; i++)
10    {
11
12        if (LeftSub[idxLeft] <= RightSub[
          idxRight])
13        {
14            arr[i] = LeftSub[idxLeft];
15            idxLeft++;
16        }
17        else
18        {
19            arr[i] = RightSub[idxRight];
20            idxRight++;
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 python

```

1 import sys
2 line = sys.stdin.readline() // 讀一行
3 D, R, N = map(int, line[:-1].split()) // 分
  三個 int 變數

```

7.5 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 }

```

7.6 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
34     return result;
35 }

```

7.7 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,
              colNum, n)][i] = false;
31        }
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<
  int>(n * n + 1, 0));
38 vector<vector<bool>> isRow(n * n + 1, vector<
  bool>(n * n + 1, false));
39 vector<vector<bool>> isColumn(n * n + 1,
  vector<bool>(n * n + 1, false));
40 vector<vector<bool>> isSquare(n * n + 1,
  vector<bool>(n * n + 1, false));
41
42 for (int i = 0; i < n * n; ++i)
43 {
44     for (int j = 0; j < n * n; ++j)
45     {
46         int number;
47         cin >> number;
48         board[i][j] = number;

```

```

49         if (number == 0)
50             continue;
51         isRow[i][number] = true;
52         isColumn[j][number] = true;
53         isSquare[getSquareIndex(i, j, n)][
          number] = true;
54     }
55 }
56 if (backtracking(board, isRow, isColumn,
  isSquare, 0, n))
57     /*有解答*/
58 else
59     /*解答*/

```

8 String

8.1 KMP

```

1 // 用在一個 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
              shift " << i - m + 1 << endl;
32             ;
33             q = 0;
34         }
35     }
36 }
37 // string s = "abcdabcdebcd";
38 // string p = "bcd";
39 // KMPMatcher(s, p);
40 // 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 - 1][j], dp[i][j - 1]),
18                     dp[i - 1][j - 1]);
19         }
20     }
21     return dp[m][n];
22 }

```

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);
26 }

```

8.4 Split

```

1 vector<string> mysplit(const string &str,
2     const string &delim)

```

```

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

```

9 data structure

9.1 Bigint

```

1 //台大
2 struct Bigint
3 {
4     static const int LEN = 60; //
5     static const int BIGMOD = 10000; //10為
6     int s; // 正常位數
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        v1 = nl; //v.resize(nl);
74        fill(v, v + v1, 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        out << str;
107    }
108    return out;
109 }
110 int cp3(const Bigint &b) const
111 {
112     if (s != b.s)
113         return s - b.s;
114     if (s == -1)
115         return -(*this).cp3(-b);
116     if (len() != b.len())
117         return len() - b.len(); //int
118     for (int i = len() - 1; i >= 0; i--)
119         if (v[i] != b.v[i])
120             return v[i] - b.v[i];
121     return 0;
122 }
123 bool operator<(const Bigint &b) const
124 {
125     return cp3(b) < 0;
126 }
127 bool operator<=(const Bigint &b) const
128 {
129     return cp3(b) <= 0;
130 }
131 bool operator==(const Bigint &b) const
132 {
133     return cp3(b) == 0;
134 }
135 bool operator!=(const Bigint &b) const
136 {
137     return cp3(b) != 0;
138 }
139 bool operator>(const Bigint &b) const
140 {
141     return cp3(b) > 0;
142 }
143 bool operator>=(const Bigint &b) const
144 {
145     return cp3(b) >= 0;
146 }
147 Bigint operator-() const
148 {
149     Bigint r = (*this);
150     r.s = -r.s;
151     return r;
152 }
153 Bigint operator+(const Bigint &b) const
154 {
155     if (s == -1)
156         return -(*this) + (-b);
157     if (b.s == -1)
158         return (*this) - (-b);
159     Bigint r;
160     int nl = max(len(), b.len());
161     r.resize(nl + 1);
162     for (int i = 0; i < nl; i++)
163     {
164         if (i < len())
165             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] /
            BIGMOD;
168         r.v[i] %= BIGMOD;
169     }
170 }
171 r.n();
172 return r;
173 }
174 BigInt operator-(const BigInt &b) const
175 {
176     if (s == -1)
177         return -(*this) - (-b);
178     if (b.s == -1)
179         return (*this) + (-b);
180     if ((*this) < b)
181         return -(b - (*this));
182     BigInt r;
183     r.resize(len());
184     for (int i = 0; i < len(); i++)
185     {
186         r.v[i] += v[i];
187         if (i < b.len())
188             r.v[i] -= b.v[i];
189         if (r.v[i] < 0)
190         {
191             r.v[i] += BIGMOD;
192             r.v[i + 1]--;
193         }
194     }
195     r.n();
196     return r;
197 }
198 BigInt operator*(const BigInt &b)
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             r.v[i + j] += v[i] * b.v[j];
208             if (r.v[i + j] >= BIGMOD)
209             {
210                 r.v[i + j + 1] += r.v[i
                    + j] / BIGMOD;
211                 r.v[i + j] %= BIGMOD;
212             }
213         }
214     }
215     r.n();
216     return r;
217 }
218 BigInt operator/(const BigInt &b)
219 {
220     BigInt r;
221     r.resize(max(1, len() - b.len() + 1)
        );
222     int oriS = s;
223     BigInt b2 = b; // b2 = abs(b)
224     s = b2.s = r.s = 1;

```

```

225     for (int i = r.len() - 1; i >= 0; i
        --)
226     {
227         int d = 0, u = BIGMOD - 1;
228         while (d < u)
229         {
230             int m = (d + u + 1) >> 1;
231             r.v[i] = m;
232             if ((r * b2) > (*this))
233                 u = m - 1;
234             else
235                 d = m;
236         }
237         r.v[i] = d;
238     }
239     s = oriS;
240     r.s = s * b.s;
241     r.n();
242     return r;
243 }
244 BigInt operator%(const BigInt &b)
245 {
246     return (*this) - (*this) / b * b;
247 }
248 };

```

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,
        c) {}
10    Matrix(mt a) { m = a, r = a.size(), c =
        a[0].size(); }
11    rt &operator[](int i) { return m[i]; }
12    matrix operator+(const matrix &a)
13    {
14        matrix rev(r, c);
15        for (int i = 0; i < r; ++i)
16            for (int j = 0; j < c; ++j)
17                rev[i][j] = m[i][j] + a.m[i
                    ][j];
18        return rev;
19    }
20    matrix operator-(const matrix &a)
21    {
22        matrix rev(r, c);
23        for (int i = 0; i < r; ++i)
24            for (int j = 0; j < c; ++j)
25                rev[i][j] = m[i][j] - a.m[i
                    ][j];
26        return rev;
27    }
28    matrix operator*(const matrix &a)
29    {
30        matrix rev(r, a.c);
31        matrix tmp(a.c, a.r);

```

```

32     for (int i = 0; i < a.r; ++i)
33         for (int j = 0; j < a.c; ++j)
34             tmp[j][i] = a.m[i][j];
35     for (int i = 0; i < r; ++i)
36         for (int j = 0; j < a.c; ++j)
37             for (int k = 0; k < c; ++k)
38                 rev.m[i][j] += m[i][k] *
                    tmp[j][k];
39     return rev;
40 }
41 bool inverse() //逆矩陣判斷
42 {
43     Matrix t(r, r + c);
44     for (int y = 0; y < r; y++)
45     {
46         t.m[y][c + y] = 1;
47         for (int x = 0; x < c; ++x)
48             t.m[y][x] = m[y][x];
49     }
50     if (!t.gas())
51         return false;
52     for (int y = 0; y < r; y++)
53         for (int x = 0; x < c; ++x)
54             m[y][x] = t.m[y][c + x] / t.
                m[y][y];
55     return true;
56 }
57 T gas() //行列式
58 {
59     vector<T> lazy(r, 1);
60     bool sign = false;
61     for (int i = 0; i < r; ++i)
62     {
63         if (m[i][i] == 0)
64         {
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]
                    [i] - m[i][k] * mx;
81         }
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 // biginteger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        s.clear();
19        int x, len = (str.length() - 1) /
            WIDTH + 1;
20        for(int i = 0; i < len; i++){
21            int end = str.length() - i*WIDTH
                ;
22            int start = max(0, end-WIDTH);
23            sscanf(str.substr(start, end-
                start).c_str(), "%d", &x);
24            s.push_back(x);
25        }
26        return *this;
27    }
28    BigInteger operator + (const BigInteger&
        b) const{
29        BigInteger c;
30        c.s.clear();
31        for(int i = 0, g = 0; i < s.size() && i
            < b.s.size()){
32            int x = g;
33            if(i < s.size()) x+=s[i];
34            if(i < b.s.size()) x+=b.s[i];
35            c.s.push_back(x % BASE);
36            g = x / BASE;
37        }
38        return c;
39    }
40    };
41 };
42 };
43 };
44 ostream& operator << (ostream &out, const
    BigInteger& x){
45     out << x.s.back();
46     for(int i = x.s.size()-2; i >= 0; i--){
47         char buf[20];
48         sprintf(buf, "%08d", x.s[i]);
49         for(int j = 0; j < strlen(buf); j++){
50             out << buf[j];
51         }
52     }
53     return out;
54 }
55 }

```



```

56 | istream& operator >> (istream &in,
    |     BigInteger& x){
57 |     string s;
58 |     if(!(in >> s))
59 |         return in;
60 |     x = s;
61 |     return in;
62 | }
63 |
64 | struct Trie{
65 |     int c[5000005][10];
66 |     int val[5000005];
67 |     int sz;
68 |     int getIndex(char c){
69 |         return c - '0';
70 |     }
71 |     void init(){
72 |         memset(c[0], 0, sizeof(c[0]));
73 |         memset(val, -1, sizeof(val));
74 |         sz = 1;
75 |     }
76 |     void insert(BigInteger x, int v){
77 |         int u = 0;
78 |         int max_len_count = 0;
79 |         int firstNum = x.s.back();
80 |         char firstBuf[20];
81 |         sprintf(firstBuf, "%d", firstNum);
82 |         for(int j = 0; j < strlen(firstBuf);
    |             j++){
83 |             int index = getIndex(firstBuf[j
    |             ]);
84 |             if(!c[u][index]){
85 |                 memset(c[sz], 0, sizeof(c[
    |                 sz]));
86 |                 val[sz] = v;
87 |                 c[u][index] = sz++;
88 |             }
89 |             u = c[u][index];
90 |             max_len_count++;
91 |         }
92 |         for(int i = x.s.size()-2; i >= 0; i
    |         --){
93 |             char buf[20];
94 |             sprintf(buf, "%08d", x.s[i]);
95 |             for(int j = 0; j < strlen(buf)
    |             && max_len_count < 50; j++){
96 |                 int index = getIndex(buf[j])
    |                 ;
97 |                 if(!c[u][index]){
98 |                     memset(c[sz], 0, sizeof
    |                     (c[sz]));
99 |                     val[sz] = v;
100 |                     c[u][index] = sz++;
101 |                 }
102 |                 u = c[u][index];
103 |                 max_len_count++;
104 |             }
105 |             if(max_len_count >= 50){
106 |                 break;
107 |             }
108 |         }
109 |     }
110 |     int find(const char* s){
111 |         int u = 0;
112 |         int n = strlen(s);
113 |         for(int i = 0; i < n; ++i)

```

```

114 |         {
115 |             int index = getIndex(s[i]);
116 |             if(!c[u][index]){
117 |                 return -1;
118 |             }
119 |             u = c[u][index];
120 |         }
121 |         return val[u];
122 |     }
123 | }

```

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 =
    |     1) : n(_n), d(_d)
6 |     {
7 |         ll t = __gcd(n, d);
8 |         n /= t, d /= t;
9 |         if (d < 0)
10 |             n = -n, d = -d;
11 |     }
12 |     fraction operator-() const
13 |     {
14 |         return fraction(-n, d);
15 |     }
16 |     fraction operator+(const fraction &b)
    |     const
17 |     {
18 |         return fraction(n * b.d + b.n * d, d * b
    |         .d);
19 |     }
20 |     fraction operator-(const fraction &b)
    |     const
21 |     {
22 |         return fraction(n * b.d - b.n * d, d * b
    |         .d);
23 |     }
24 |     fraction operator*(const fraction &b)
    |     const
25 |     {
26 |         return fraction(n * b.n, d * b.d);
27 |     }
28 |     fraction operator/(const fraction &b)
    |     const
29 |     {
30 |         return fraction(n * b.d, d * b.n);
31 |     }
32 |     void print()
33 |     {
34 |         cout << n;
35 |         if (d != 1)
36 |             cout << "/" << d;
37 |     }
38 | };

```

TO DO WRITING NOT THINKING

Contents

1 Basic	1	2.7 LIS	2	5.4 Dijkstra	7	6.15 羅馬數字	10
1.1 A function	1	2.8 LPS	2	5.5 Euler circuit	7	6.16 質因數分解	10
1.2 Codeblock setting	1	2.9 Max_subarray	2	5.6 Floyd-warshall	7		
1.3 data range	1	2.10 Money problem	2	5.7 Hamilton_cycle	7	7 Other	10
1.4 IO_fast	1			5.8 Kruskal	7	7.1 binary search 三類變化	10
2 DP	1	3 Flow & matching	2	5.9 Prim	8	7.2 heap sort	10
2.1 3 維 DP 思路	1	3.1 Dinic	2	5.10 Union_find	8	7.3 Merge sort	11
2.2 Knapsack Bounded	1	3.2 Edmonds_karp	3			7.4 python	11
2.3 Knapsack sample	1	3.3 hungarian	3	6 Mathematics	8	7.5 Quick	11
2.4 Knapsack Unbounded	1	3.4 Maximum_matching	3	6.1 Combination	8	7.6 Weighted Job Scheduling	11
2.5 LCIS	1	3.5 MFlow Model	3	6.2 Extended Euclidean	8	7.7 數獨解法	11
2.6 LCS	1			6.3 Hex to Dec	8		
		4 Geometry	4	6.4 log	9	8 String	11
		4.1 Closest Pair	4	6.5 Mod	9	8.1 KMP	11
		4.2 Line	4	6.6 Permutation	9	8.2 Min Edit Distance	12
		4.3 Point	4	6.7 PI	9	8.3 Sliding window	12
		4.4 Polygon	5	6.8 Prime table	9	8.4 Split	12
		4.5 Triangle	6	6.9 primeBOOL	9		
		5 Graph	6	6.10 Round(小數)	9	9 data structure	12
		5.1 Bellman-Ford	6	6.11 二分逼近法	9	9.1 Bigint	12
		5.2 BFS-queue	6	6.12 四則運算	9	9.2 matirx	13
		5.3 DFS-rec	6	6.13 數字乘法組合	10	9.3 Trie	13
				6.14 數字加法組合	10	9.4 分數	14