

1 Basic

1.1 Code Template

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef unsigned long long ull;
5 typedef pair<int, int> pii;
6 #define pb push_back
7 #define endl '\n'
8 #define bug(x) cout << "value of x is " << x
9 << endl;
10 #define bugarr(x) \
11     for (auto i : x) \
12         cout << i << ' '; \
13         cout << endl;
14 #define x first
15 #define y second
16 int main()
17 {
18     ios::sync_with_stdio(0);
19     cin.tie(0);
20     return 0;
21 }
```

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 IO_fast

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

1.4 Python

```
1 //輸入
2 import sys
3 line = sys.stdin.readline() // 會讀到換行
4 input().strip()
```

```
5 |
6 | array = [0] * (N) //N個0
7 | range(0, N) // 0 ~ N-1
8 | D, R, N = map(int, line[: -1].split()) // 分
   | 三個 int 變數
9 |
10 | pow(a, b, c) // a ^ b % c
11 |
12 | print(*objects, sep = ' ', end = '\n')
13 | // objects -- 可以一次輸出多個對象
14 | // sep -- 分開多個objects
15 | // end -- 默認值是\n
```

1.5 Range data

```
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)
```

1.6 Some 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 | /** 全排列要先 sort !!! **/
8 | next_permutation(num.begin(), num.end());
9 | prev_permutation(num.begin(), num.end());
10 | //用binary search找大於或等於val的最小值的位
    | 置
11 | vector<int>::iterator it = lower_bound(v.
    | begin(), v.end(), val);
12 | //用binary search找大於val的最小值的位置
13 | vector<int>::iterator it = upper_bound(v.
    | begin(), v.end(), val);
14 | /**queue*/
15 |
16 | queue<datatype> q;
17 | front(); /*取出最前面的值(沒有移除掉)*/
18 | back(); /*取出最後面的值(沒有移除掉)*/
19 | pop(); /*移掉最前面的值*/
20 | push(); /*新增值到最後面*/
21 | empty(); /*回傳bool, 檢查是不是空的queue*/
22 | size(); /*queue 的大小*/
23 |
24 | /**stack*/
25 | stack<datatype> s;
26 | top(); /*取出最上面的值(沒有移除掉)*/
27 | pop(); /*移掉最上面的值*/
```

```
28 | push(); /*新增值到最上面*/
29 | empty(); /*bool 檢查是不是空*/
30 | size(); /*stack 的大小*/
31 |
32 | /*unordered_set*/
33 | unordered_set<datatype> s;
34 | unordered_set<datatype> s(arr, arr + n);
35 | /*initial with array*/
36 | insert(); /*插入值*/
37 | erase(); /*刪除值*/
38 | empty(); /*bool 檢查是不是空*/
39 | count(); /*判斷元素存在回傳1 無則回傳0*/
```

1.7 Time

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

1.8 Vim setting

```
1 | /*at home directory*/
2 | /* vi ~/.vimrc */
3 | syntax enable
4 | set smartindent
5 | set tabstop=4
6 | set shiftwidth=4
7 | set expandtab
8 | set relativenumber
```

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(),
   |     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
15 |                [i - 1][j];
16 |            else dp[i][j] = max(dp[i - 1][j],
17 |                dp[i - 1][j - weight[i]]
18 |                + value[i]);
19 |        }
20 |    }
21 |    cout << dp[weight.size() - 1][bagWeight]
22 |        << endl;
23 | }
```

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];
}

```

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<
        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 vector<int> ans;
2 void printLIS(vector<int> &arr, vector<int>
    &pos, int index)
3 {
4     if (pos[index] != -1)
5         printLIS(arr, pos, pos[index]);
6     // printf("%d", arr[index]);
7     ans.push_back(arr[index]);
8 }
9 void LIS(vector<int> &arr)
10 {
11     vector<int> dp(arr.size(), 1);
12     vector<int> pos(arr.size(), -1);
13     int res = INT_MIN, index = 0;
14     for (int i = 0; i < arr.size(); ++i)
15     {
16         for (int j = i + 1; j < arr.size();
            ++j)
17         {
18             if (arr[j] > arr[i])
19             {
20                 if (dp[i] + 1 > dp[j])
21                     dp[j] = dp[i] + 1;
22                 pos[j] = i;
23             }
24         }
25     }
26     if (dp[i] > res)
27     {
28         res = dp[i];
29         index = i;
30     }
31 }
32 cout << res << endl; // length
33 printLIS(arr, pos, index);
34 for (int i = 0; i < ans.size(); i++)
35 {
36     cout << ans[i];
37 }

```

```

38     if (i != ans.size() - 1)
39         cout << ' ';
40 }
41 cout << '\n';
42 }

```

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
            - x >= 0) && (i + x < n)) //odd
9             x++;
10        x--;
11        if (2 * x + 1 > maxlen)
12        {
13            maxlen = 2 * x + 1;
14            l = i - x;
15            r = i + x;
16        }
17        x = 0;
18        while ((s[i - x] == s[i + 1 + x]) &&
            (i - x >= 0) && (i + 1 + x < n)) //even length
19            x++;
20        if (2 * x > maxlen)
21        {
22            maxlen = 2 * x;
23            l = i - x + 1;
24            r = i + x;
25        }
26    }
27    cout << maxlen << '\n'; // 最後長度
28    cout << l + 1 << ' ' << r + 1 << '\n';
29    //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 =
        nums[0];
4     for (int i = 1; i < nums.size(); i++) {
5         local_max = max(nums[i], nums[i] +
            local_max);
6         global_max = max(local_max,
            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;
            ++j) // 由低價位逐步到高價位
9             c[j] = c[j] | c[j - price[i]];
10        // 湊、湊、湊
11        if (c[limit]) cout << "YES\n";
12        else cout << "NO\n";
13 }
14 // 湊得某個價位的湊法總共幾種
15 void change(vector<int> price, int limit)
16 {
17     vector<int> c(limit + 1, 0);
18     c[0] = true;
19     for (int i = 0; i < price.size(); ++i)
20         for (int j = price[i]; j <= limit;
            ++j)
21             c[j] += c[j - price[i]];
22    cout << c[limit] << '\n';
23 }
24 // 湊得某個價位的最少錢幣用量
25 void change(vector<int> price, int limit)
26 {
27     vector<int> c(limit + 1, 0);
28     c[0] = true;
29     for (int i = 0; i < price.size(); ++i)
30         for (int j = price[i]; j <= limit;
            ++j)
31             c[j] = min(c[j], c[j - price[i]]
                + 1);
32    cout << c[limit] << '\n';
33 }
34 //湊得某個價位的錢幣用量，有哪幾種可能性
35 void change(vector<int> price, int limit)
36 {
37     vector<int> c(limit + 1, 0);
38     c[0] = true;
39     for (int i = 0; i < price.size(); ++i)
40         for (int j = price[i]; j <= limit;
            ++j)
41             c[j] |= c[j - price[i]] << 1; //
            錢幣數量加一，每一種可能性都
            加一。
42    for (int i = 1; i <= 63; ++i)
43        if (c[i] & (1 << i))
44            cout << "用" << i << "個錢幣可湊
            得價位" << m;

```

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

```

```

27         residual[cur][pre[cur]] -=
28         bottleneck[t];
29     }
30     ans += bottleneck[t];
31 }
32 return ans;
33 }
34 int main(){
35     int testcase = 1;
36     int n;
37     while(cin>>n){
38         if(n == 0)
39             break;
40         vector<vector<int>> capacity(n+1, vector
41             <int>(n+1, 0));
42         int s, t, c;
43         cin >> s >> t >> c;
44         int a, b, bandwidth;
45         for(int i = 0 ; i < c ; ++i){
46             cin >> a >> b >> bandwidth;
47             capacity[a][b] += bandwidth;
48             capacity[b][a] += bandwidth;
49         }
50         cout << "Network " << testcase++ << endl
51             << endl;
52         cout << "The bandwidth is " <<
53             getMaxFlow(capacity, s, t, n) << "."
54             << 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){

```

```

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 input:
42 4 3 5 //n matching m, l links
43 0 0
44 0 2
45 1 0
46 2 1
47 3 1
48 answer is 3
49 */

```

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;
63        for (int i = first[x]; i != -1; i = next[i])
64        {
65            if (dis[x] + 1 == dis[v[i]] && (
66                f = dfs(v[i], min(a, cap[i]
67                    - flow[i])) > 0)
68            )
69            {
70                flow[i] += f;
71                flow[i ^ 1] -= f;
72                a -= f;
73                flw += f;
74                if (a == 0)
75                    break;
76            }
77        }
78        return flw;
79    }
80    ll MaxFlow(int s, int t)
81    {
82        this->s = s;
83        this->t = t;
84        ll flw = 0;
85        while (bfs())
86        {
87            for (int i = 1; i <= n; ++i)
88                cur[i] = 0;
89            flw += dfs(s, oo);
90        }
91        return flw;
92    }
93    // MF Net;
94    // Net.n = n;
95    // Net.Clear();
96    // a 到 b (注意從1開始!!!!)
97    // Net.Add(a, b, w, 0);
98    // Net.MaxFlow(s, d)
99    // 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;
28 }

```

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

```

```

1     return min({dis2(l.p1, 1), dis2(l.p2
2         , 1), l.dis2(p1, 1), l.dis2(p2,
3             1)});
4     }
5     point<T> projection(const point<T> &p)
6         const
7     { //點對直線的投影
8         point<T> n = (p2 - p1).normal();
9         return p - n * (p - p1).dot(n) / n.
10             abs2();
11    }
12    point<T> mirror(const point<T> &p) const
13    {
14        //點對直線的鏡射 · 要先呼叫pton轉成一般式
15        point<T> R;
16        T d = a * a + b * b;
17        R.x = (b * b * p.x - a * a * p.x - 2
18            * a * b * p.y - 2 * a * c) / d;
19        R.y = (a * a * p.y - b * b * p.y - 2
20            * a * b * p.x - 2 * b * c) / d;
21        return R;
22    }
23    bool equal(const line &l) const
24    { //直線相等
25        return ori(l.p1) == 0 && ori(l.p2)
26            == 0;
27    }
28    bool parallel(const line &l) const
29    {
30        return (p1 - p2).cross(l.p1 - l.p2)
31            == 0;
32    }
33    bool cross_seg(const line &l) const
34    {
35        return (p2 - p1).cross(l.p1 - p1) *
36            (p2 - p1).cross(l.p2 - p1) <= 0;
37        //直線是否交線段
38    }
39    int line_intersect(const line &l) const
40    { //直線相交情況 · -1無限多點、1交於一
41        點、0不相交
42        return parallel(l) ? (ori(l.p1) == 0
43            ? -1 : 0) : 1;
44    }
45    int seg_intersect(const line &l) const
46    {
47        T c1 = ori(l.p1), c2 = ori(l.p2);
48        T c3 = l.ori(p1), c4 = l.ori(p2);
49        if (c1 == 0 && c2 == 0)
50        { //共線
51            bool b1 = btw(l.p1) >= 0, b2 =
52                btw(l.p2) >= 0;
53            T a3 = l.btw(p1), a4 = l.btw(p2);
54            ;
55            if (b1 && b2 && a3 == 0 && a4 >=
56                0)
57                return 2;
58            if (b1 && b2 && a3 >= 0 && a4 ==
59                0)
60                return 3;
61            if (b1 && b2 && a3 >= 0 && a4 >=
62                0)
63                return 0;
64        }
65    }

```

```

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 &l
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 &l)
105     const
106 { //線段交點
107     int res = seg_intersect(l);
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(l);
115 }
116 };

```

```

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 {
50     //對x軸的弧度
51     //超過180度會變負
52     T A = atan2(y, x);
53     if (A <= -PI / 2)
54         A += PI * 2;
55     return A;
56 }
57 };

```

4.4 Polygon

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;
31     }
32     T cross(const point &b) const
33     {
34         return x * b.y - y * b.x;
35     }
36     point normal() const
37     { //求法向量
38         return point(-y, x);
39     }
40     T abs2() const
41     { //向量長度的平方
42         return dot(*this);
43     }
44     T rad(const point &b) const
45     { //兩向量的弧度
46         return fabs(atan2(fabs(cross(b)),
47             dot(b)));
48     }
49     T getA() const
50     {
51         //對x軸的弧度
52         //超過180度會變負
53         T A = atan2(y, x);
54         if (A <= -PI / 2)
55             A += PI * 2;
56         return A;
57     }
58 };

```

```

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;

```

```

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             )
40             return c;
41         c = !c;
42     }
43     return c;
44 }
45 char point_in_convex(const point<T> &x)
46     const
47 {
48     int l = 1, r = (int)p.size() - 2;
49     while (l <= r)
50     { //點是否在凸多邊形內，是的話回傳1
51         //在邊上回傳-1、否則回傳0
52         int mid = (l + r) / 2;
53         T a1 = (p[mid] - p[0]).cross(x -
54             p[0]);
55         T a2 = (p[mid + 1] - p[0]).cross
56             (x - p[0]);
57         if (a1 >= 0 && a2 <= 0)
58         {
59             T res = (p[mid + 1] - p[mid]
60                 ).cross(x - p[mid]);
61             return res > 0 ? 1 : (res >=
62                 0 ? -1 : 0);
63         }
64         else if (a1 < 0)
65             r = mid - 1;
66         else
67             l = mid + 1;
68     }
69     return 0;
70 }
71 vector<T> getA() const
72 { //凸包邊對x軸的夾角
73     vector<T> res; //一定是遞增的
74     for (size_t i = 0; i < p.size(); ++i
75         )
76         res.push_back((p[(i + 1) % p.
77             size()] - p[i]).getA());
78     return res;
79 }
80 bool line_intersect(const vector<T> &A,
81     const line<T> &l) const
82 { //O(logN)
83     int f1 = upper_bound(A.begin(), A.
84         end(), (l.p1 - l.p2).getA()) - A.
85         .begin();
86     int f2 = upper_bound(A.begin(), A.
87         end(), (l.p2 - l.p1).getA()) - A.
88         .begin();
89     return l.cross_seg(line<T>(p[f1], p[
90         f2]));
91 }
92 polygon cut(const line<T> &l) const
93 { //凸包對直線切割，得到直線l左側的凸包
94     polygon ans;
95     for (int n = p.size(), i = n - 1, j
96         = 0; j < n; i = j++)
97     {
98         if (l.ori(p[i]) >= 0)
99             ans.p.push_back(p[i]);
100         if (l.ori(p[j]) < 0)
101             ans.p.push_back(l.
102                 line_intersection(
103                     line<T>(p[i], p[j]));
104     }
105     else if (l.ori(p[j]) > 0)
106         ans.p.push_back(l.
107             line_intersection(line<T>
108                 >(p[i], p[j])));
109     }
110     return ans;
111 }
112 static bool graham_cmp(const point<T> &a
113     , const point<T> &b)
114 { //凸包排序函數 // 起點點不同
115     // return (a.x < b.x) || (a.x == b.x
116         && a.y < b.y); //最左下角開始
117     return (a.y < b.y) || (a.y == b.y &&
118         a.x < b.x); //Y最小開始
119 }
120 void graham(vector<point<T>> &s)
121 { //凸包 Convexhull 2D
122     sort(s.begin(), s.end(), graham_cmp)
123     ;
124     p.resize(s.size() + 1);
125     int m = 0;
126     // cross >= 0 順時針 * cross <= 0 逆
127     // 時針旋轉
128     for (size_t i = 0; i < s.size(); ++i
129         )
130     {
131         while (m >= 2 && (p[m - 1] - p[m
132             - 2]).cross(s[i] - p[m -
133                 2]) <= 0)
134             --m;
135         p[m++] = s[i];
136     }
137     for (int i = s.size() - 2, t = m +
138         1; i >= 0; --i)
139     {
140         while (m >= t && (p[m - 1] - p[m
141             - 2]).cross(s[i] - p[m -
142                 2]) <= 0)
143             --m;
144         p[m++] = s[i];
145     }
146     if (s.size() > 1) // 重複頭一次需扣
147         --m;
148     p.resize(m);
149 }
150 T diam()
151 { //直徑
152     int n = p.size(), t = 1;
153     T ans = 0;
154     p.push_back(p[0]);
155     for (int i = 0; i < n; i++)

```



```

118 {
119     point<T> now = p[i + 1] - p[i];
120     while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
121         t = (t + 1) % n;
122     ans = max(ans, (p[i] - p[t]).abs2());
123 }
124 return p.pop_back(), ans;
125 }
126 T min_cover_rectangle()
127 { //最小覆蓋矩形
128     int n = p.size(), t = 1, r = 1, l;
129     if (n < 3)
130         return 0; //也可以做最小周長矩形
131     T ans = 1e99;
132     p.push_back(p[0]);
133     for (int i = 0; i < n; i++)
134     {
135         point<T> now = p[i + 1] - p[i];
136         while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
137             t = (t + 1) % n;
138         while (now.dot(p[r + 1] - p[i]) > now.dot(p[r] - p[i]))
139             r = (r + 1) % n;
140         if (!i)
141             l = r;
142         while (now.dot(p[l + 1] - p[i]) <= now.dot(p[l] - p[i]))
143             l = (l + 1) % n;
144         T d = now.abs2();
145         T tmp = now.cross(p[t] - p[i]) * (now.dot(p[r] - p[i]) - now.dot(p[l] - p[i])) / d;
146         ans = min(ans, tmp);
147     }
148     return p.pop_back(), ans;
149 }
150 T dis2(polygon &p1)
151 { //凸包最近距離平方
152     vector<point<T>> &P = p, &Q = p1.p;
153     int n = P.size(), m = Q.size(), l = 0, r = 0;
154     for (int i = 0; i < n; ++i)
155         if (P[i].y < P[l].y)
156             l = i;
157     for (int i = 0; i < m; ++i)
158         if (Q[i].y < Q[r].y)
159             r = i;
160     P.push_back(P[0]), Q.push_back(Q[0]);
161     T ans = 1e99;
162     for (int i = 0; i < n; ++i)
163     {
164         while ((P[l] - P[l + 1]).cross(Q[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]);
203     }
204     while (L < R && q[L].ori(px[R - 1]) <= 0)
205         --R;
206     p.clear();
207     if (R - L <= 1)
208         return 0;
209     px[R] = q[R].line_intersection(q[L]);
210     for (int i = L; i <= R; ++i)
211         p.push_back(px[i]);
212     return R - L + 1;
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

4.5 Triangle

```

1 template <typename T>
2 struct triangle
3 {
4     point<T> a, b, c;
5     triangle() {}

```

```

15 triangle(const point<T> &a, const point<
16 T> &b, const point<T> &c) : a(a), b(b),
17 c(c) {}
18 T area() const
19 {
20     T t = (b - a).cross(c - a) / 2;
21     return t > 0 ? t : -t;
22 }
23 point<T> barycenter() const
24 { //重心
25     return (a + b + c) / 3;
26 }
27 point<T> circumcenter() const
28 { //外心
29     static line<T> u, v;
30     u.p1 = (a + b) / 2;
31     u.p2 = point<T>(u.p1.x - a.y + b.y,
32 u.p1.y + a.x - b.x);
33 v.p1 = (a + c) / 2;
34 v.p2 = point<T>(v.p1.x - a.y + c.y,
35 v.p1.y + a.x - c.x);
36 return u.line_intersection(v);
37 }
38 point<T> incenter() const
39 { //內心
40     T A = sqrt((b - c).abs2()), B = sqrt(
41 ((a - c).abs2()), C = sqrt((a - b).abs2());
42 return point<T>(A * a.x + B * b.x +
43 C * c.x, A * a.y + B * b.y + C *
44 c.y) / (A + B + C);
45 }
46 point<T> perpencenter() const
47 { //垂心
48     return barycenter() * 3 -
49 circumcenter() * 2;
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 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

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 i = 0; i < node-1; i++){
10         for(int j = 0; j < node; j++){
11             for(int k = 0; k < node; k++){
12                 if(edges[j][k] != -1){
13                     if(dist[j] + edges[j][k] < dist[k]){
14                         dist[k] = dist[j] +
15 edges[j][k];
16                         ancestor[k] = j;
17                     }
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 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254
```

```

20     else if (edges[i].second == q.
21         front() && pass[edges[i].
22         first] == 0)
23     {
24         p.push(edges[i].first);
25         result[edges[i].first] =
26             count;
27     }
28     else
29         newedges.push_back(edges[i])
30         ;
31     }
32     edges = newedges;
33     newedges.clear();
34     q.pop();
35     if (q.empty() == true)
36     {
37         q = p;
38         queue<int> tmp;
39         p = tmp;
40         count++;
41     }
42 }
43 int main()
44 {
45     int node;
46     cin >> node;
47     vector<pair<int, int>> edges;
48     int a, b;
49     while (cin >> a >> b)
50     {
51         /*a = b = -1 means input edges ended
52         */
53         if (a == -1 && b == -1)
54             break;
55         edges.push_back(pair<int, int>(a, b)
56         );
57     }
58     vector<int> result(node, -1);
59     BFS(result, edges, node, 0);
60     return 0;
61 }

```

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

```

```

32 // weight[a - 1].push_back(pii(b - 1, w));
33 // weight[b - 1].push_back(pii(a - 1, w));
34 // dist.resize(n, inf);
35 // ancestor.resize(n, -1);
36 // dist[0] = 0;
37 // dijkstra(0);

```

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                  {
12                      distance[i][j] =
13                          distance[i][k] +
14                          distance[k][j];
15                      ancestor[i][j] =
16                          ancestor[k][j];
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-1][b-1] = d;
33             ancestor[a-1][b-1] = a-1;
34         }
35         floyd_warshall(distance, ancestor, n);
36         for(int i = 0; i < n; i++){
37             for(int j = 0; j < n; j++){
38                 if(distance[i][j] < 99999)
39                     cout << "Negative cycle!" <<
40                     endl;
41             }
42         }
43         return 0;
44     }

```

```

23         break;
24         distance[a][b] = d;
25         ancestor[a][b] = a;
26     }
27     for (int i = 0; i < n; i++)
28         distance[i][i] = 0;
29     floyd_warshall(distance, ancestor, n);
30     /*Negative cycle detection*/
31     for (int i = 0; i < n; i++){
32         if(distance[i][i] < 0){
33             cout << "Negative cycle!" <<
34             endl;
35             break;
36         }
37     }
38     return 0;
39 }

```

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++){
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)
37                 break;
38             gp[a][b] = 1;
39             gp[b][a] = 1;
40         }
41         vector<int> solution(n + 1, -1);
42         vector<bool> pass(n + 1, false);
43         solution[1] = 0;
44     }

```

```

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             ;
35         int to = find(cur.to, union_set);
36         if(from != to){
37             merge(from, to, union_set);
38             edge += 1;
39             cost += cur.weight;
40         }
41         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();
31         if(!pass[cur.to]){
32             for (int i = 0; i < n; i++){
33                 if(gp[cur.to][i] != inf){
34                     edges tmp;
35                     tmp.from = cur.to;
36                     tmp.to = i;
37                     tmp.weight = gp[cur.to][i];
38                     pq.push(tmp);
39                 }
40             }
41             pass[cur.to] = true;
42             edge += 1;
43             cost += cur.weight;
44         }
45         pq.pop();

```

```

35         tmp.weight = gp[cur.to][
36             i];
37         pq.push(tmp);
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];
25     }
26 }
27 int main()
28 {
29     int node;
30     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 Catalan

Catalan number

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

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

6.2 Combination

```

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
3      a, long long b)
4  {
5      if (b == 0)

```



```

5     return {1, 0};
6     long long k = a / b;
7     pair<long long, long long> p = extgcd(b,
8     //cout << p.first << " " << p.second <<
9     //cout << "商數(k)= " << k << endl <<
10    //return {p.second, p.first - k * p.second};
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
26 // ax + by = gcd(a,b) * r
27 /*find |x|+|y| -> min*/
28 int main()
29 {
30     long long r, p, q; /*px+qy = r*/
31     int cases;
32     cin >> cases;
33     while (cases--)
34     {
35         cin >> r >> p >> q;
36         pair<long long, long long> xy =
37         extgcd(q, p); //(x0,y0)
38         long long ans = 0, tmp = 0;
39         double k, k1;
40         long long s, s1;
41         k = 1 - (double)(r * xy.first) / p;
42         s = round(k);
43         ans = llabs(r * xy.first + s * p) +
44         llabs(r * xy.second - s * q);
45         k1 = -(double)(r * xy.first) / p;
46         s1 = round(k1);
47         /*cout << k << endl << k1 << endl;
48         cout << s << endl << s1 << endl;
49         */
50         tmp = llabs(r * xy.first + s1 * p) +
51         llabs(r * xy.second - s1 * q);
52         ans = min(ans, tmp);
53         cout << ans << endl;
54     }
55     return 0;
56 }

```

6.4 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
 - $a \equiv b \pmod{p} \Leftrightarrow am \equiv bm \pmod{p}$

6.5 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     {
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] <=
14         'F')
15         {
16             temp += (num[i] - 55) * base;
17             base = base * 16;
18         }
19     }
20     return temp;
21 }
22 void DecToHex(int p) //10 to 16
23 {
24     char *l = new (char);
25     sprintf(l, "%X", p);
26     //int l_intResult = stoi(l);
27     cout << l << "\n";
28     //return l_intResult;
29 }

```

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 }

```

6.7 Mod

```

1 int pow_mod(int a, int n, int m) // a ^ n
2     mod m;
3 {
4     // a, n, m
5     < 10 ^ 9
6     if (n == 0)
7         return 1;
8     int x = pow_mid(a, n / 2, m);
9     long long ans = (long long)x * x % m;
10    if (n % 2 == 1)
11        ans = ans * a % m;
12    return (int)ans;
13 }
14 int inv(int a, int n, int p) // n = p-2
15 {
16     long long res = 1;
17     for (; n >= 1, (a *= a) %= p)
18         if (n & 1)
19             (res *= a) %= p;
20     return res;
21 }

```

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 $\gcd(a, n) = 1$ ，則： $a^{\phi(n)} \equiv 1 \pmod{n}$ 如果 n 是質數， $\phi(n) = n - 1$ ，也就是費馬小定理。

Wilson's theorem

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

6.9 PI

```

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 >= 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 >= 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 }
//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 因數表

```

1 vector<vector<int>> arr(1000000);
2 const int limit = 10e7;
3 for (int i = 1; i <= limit; i++)
4 {
5     for (int j = i; j <= limit; j += i)
6         arr[j].pb(i); // i 為因數
7 }

```

6.17 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;

```

6.18 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
2           &out, vector<vector<int>> &ans)
3 {

```

```

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         ++
25         cout << ans[i][j] << " ";
26     cout << ans[i][ans[i].size() - 1] <<
27     endl;
28 }

```

6.19 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 }

```

6.20 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
18 // 於目標值的數
19 /*(lower_bound)*/
20 int find(vector<int> &nums, int target)
21 {
22     int left = 0, right = nums.size();
23     while (left < right)
24     {
25         int mid = left + (right - left) / 2;
26         if (nums[mid] < target)
27             left = mid + 1;
28        else
29            right = mid;
30    }
31    return right;
32 }
33 // 找第一個大於目標值的數 == 找最後一個不大
34 // 於目標值的數
35 /*(upper_bound)*/
36 int find(vector<int> &nums, int target)
37 {
38     int left = 0, right = nums.size();
39     while (left < right)
40     {
41         int mid = left + (right - left) / 2;
42         if (nums[mid] <= target)
43             left = mid + 1;
44        else

```

```

43        right = mid;
44    }
45    return right;
46 }

```

7.2 heap sort

```

1 void MaxHeapify(vector<int> &array, int root
2     , int length)
3 {
4     int left = 2 * root,
5         right = 2 * root + 1,
6         largest;
7     if (left <= length && array[left] >
8         array[root])
9         largest = left;
10    else if (right <= length && array[right] >
11        array[largest])
12        largest = right;
13    if (largest != root)
14    {
15        swap(array[largest], array[root]);
16        MaxHeapify(array, largest, length);
17    }
18 }
19 void HeapSort(vector<int> &array)
20 {
21     array.insert(array.begin(), 0);
22     for (int i = (int)array.size() / 2; i >=
23         1; i--)
24         MaxHeapify(array, i, (int)array.size
25             () - 1);
26     int size = (int)array.size() - 1;
27     for (int i = (int)array.size() - 1; i >=
28         2; i--)
29     {
30         swap(array[1], array[i]);
31         size--;
32         MaxHeapify(array, 1, size);
33     }
34     array.erase(array.begin());
35 }

```

7.3 Merge sort

```

1 void Merge(vector<int> &arr, int front, int
2     mid, int end)
3 {
4     vector<int> LeftSub(arr.begin() + front,
5         arr.begin() + mid + 1);
6     vector<int> RightSub(arr.begin() + mid +
7         1, arr.begin() + end + 1);
8     LeftSub.insert(LeftSub.end(), INT_MAX);
9     RightSub.insert(RightSub.end(), INT_MAX);
10    ;
11    int idxLeft = 0, idxRight = 0;
12    for (int i = front; i <= end; i++)

```

```

10    {
11        if (LeftSub[idxLeft] <= RightSub[
12            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,
25     int end)
26 {
27     // front = 0, end = arr.size() - 1
28     if (front < end)
29     {
30         int mid = (front + end) / 2;
31         MergeSort(arr, front, mid);
32         MergeSort(arr, mid + 1, end);
33         Merge(arr, front, mid, end);
34     }
35 }

```

7.4 Quick

```

1 int Partition(vector<int> &arr, int front,
2     int end)
3 {
4     int pivot = arr[end];
5     int i = front - 1;
6     for (int j = front; j < end; j++)
7     {
8         if (arr[j] < pivot)
9         {
10            i++;
11            swap(arr[i], arr[j]);
12        }
13    }
14    i++;
15    swap(arr[i], arr[end]);
16    return i;
17 }
18 void QuickSort(vector<int> &arr, int front,
19     int end)
20 {
21     // front = 0, end = arr.size() - 1
22     if (front < end)
23     {
24         int pivot = Partition(arr, front,
25             end);
26         QuickSort(arr, front, pivot - 1);
27         QuickSort(arr, pivot + 1, end);
28     }
29 }

```

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

```

```

16     for (int i = 1; i <= n2; i++)
17     {
18         if (!rows[rowNum][i] && !cols[colNum
19             ][i] && !boxes[getSquareIndex(
20                 rowNum, colNum, n)][i])
21         {
22             rows[rowNum][i] = true;
23             cols[colNum][i] = true;
24             boxes[getSquareIndex(rowNum,
25                 colNum, n)][i] = true;
26             board[rowNum][colNum] = i;
27             if (backtracking(board, rows,
28                 cols, boxes, index + 1, n))
29                 return true;
30             board[rowNum][colNum] = 0;
31             rows[rowNum][i] = false;
32             cols[colNum][i] = false;
33             boxes[getSquareIndex(rowNum,
34                 colNum, n)][i] = false;
35         }
36     }
37     return false;
38 }
39 /*用法 main*/
40 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
41 vector<vector<int>> board(n * n + 1, vector<
42     int>(n * n + 1, 0));
43 vector<vector<bool>> isRow(n * n + 1, vector<
44     bool>(n * n + 1, false));
45 vector<vector<bool>> isColumn(n * n + 1,
46     vector<bool>(n * n + 1, false));
47 vector<vector<bool>> isSquare(n * n + 1,
48     vector<bool>(n * n + 1, false));
49 for (int i = 0; i < n * n; ++i)
50 {
51     for (int j = 0; j < n * n; ++j)
52     {
53         int number;
54         cin >> number;
55         board[i][j] = number;
56         if (number == 0)
57             continue;
58         isRow[i][number] = true;
59         isColumn[j][number] = true;
60         isSquare[getSquareIndex(i, j, n)][
61             number] = true;
62     }
63 }
64 if (backtracking(board, isRow, isColumn,
65     isSquare, 0, n))
66     /*有解答*/
67 else
68     /*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     for (int i = 0, q = 0; i < n; i++)
23     {
24         while (q > 0 && pattern[q] != text[i]
25             )
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 }
37 // string s = "abcdabcbcd";
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
18                     - 1][j], dp[i][j - 1]),

```

```

17         )
18         )
19         return dp[m][n];
20 }

```

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     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 //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;
8     int vl, v[LEN];
9     // vector<int> v;
10    Bigint() : s(1) { vl = 0; }
11    Bigint(long long a)
12    {
13        s = 1;
14        vl = 0;
15        if (a < 0)
16        {
17            s = -1;
18            a = -a;
19        }
20        while (a)
21        {
22            push_back(a % BIGMOD);
23            a /= BIGMOD;
24        }
25        Bigint(string str)
26        {
27            s = 1;
28            vl = 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            }
46            if (num)
47                push_back(num);
48            n();
49        }
50        int len() const
51        {
52            return vl; //return SZ(v);
53        }
54        bool empty() const { return len() == 0; }
55        void push_back(int x)
56        {

```

```

56        v[vl++] = x; //v.PB(x);
57    }
58    void pop_back()
59    {
60        vl--; //v.pop_back();
61    }
62    int back() const
63    {
64        return v[vl - 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            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        return 0;
119    }
120    bool operator<(const Bigint &b) const
121    {
122        return cp3(b) < 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 -(b - (*this));
183        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            {
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)

```


9.2 Matirx

```

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,
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][j];
20         return rev;
21     }
22     matrix operator-(const matrix &a)
23     {
24         matrix rev(r, c);
25         for (int i = 0; i < r; ++i)
26             for (int j = 0; j < c; ++j)
27                 rev[i][j] = m[i][j] - a.m[i][j];
28         return rev;
29     }
30     matrix operator*(const matrix &a)
31     {
32         matrix rev(r, a.c);
33         matrix tmp(a.c, a.r);
34         for (int i = 0; i < a.r; ++i)
35             for (int j = 0; j < a.c; ++j)
36                 tmp[j][i] = a.m[i][j];
37         for (int i = 0; i < r; ++i)
38             for (int j = 0; j < a.c; ++j)
39                 for (int k = 0; k < c; ++k)
40                     rev.m[i][j] += m[i][k] *
41                         tmp[j][k];
42         return rev;
43     }
44     bool inverse() //逆矩陣判斷
45     {
46         Matrix t(r, r + c);
47         for (int y = 0; y < r; y++)
48         {
49             t.m[y][c + y] = 1;
50             for (int x = 0; x < c; ++x)
51                 t.m[y][x] = m[y][x];
52         }
53         if (!t.gas())
54             return false;
55     }
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

```

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.
55                 m[y][y];
56     return true;
57 }
58 T gas() //行列式
59 {
60     vector<T> lazy(r, 1);
61     bool sign = false;
62     for (int i = 0; i < r; ++i)
63     {
64         if (m[i][i] == 0)
65         {
66             int j = i + 1;
67             while (j < r && !m[j][i])
68                 j++;
69             if (j == r)
70                 continue;
71             m[i].swap(m[j]);
72             sign = !sign;
73         }
74         for (int j = 0; j < r; ++j)
75         {
76             if (i == j)
77                 continue;
78             lazy[j] = lazy[j] * m[i][i];
79             T mx = m[j][i];
80             for (int k = 0; k < c; ++k)
81                 m[j][k] = m[j][k] * m[i][i] - m[i][k] * mx;
82         }
83     }
84     T det = sign ? -1 : 1;
85     for (int i = 0; i < r; ++i)
86     {
87         det = det * m[i][i];
88         det = det / lazy[i];
89         for (auto &j : m[i])
90             j /= lazy[i];
91     }
92     return det;
93 }
94 };

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;

```

```

74     }
75     BigInteger operator = (const string& str
76     ){
77         s.clear();
78         int x, len = (str.length() - 1) /
79             WIDTH + 1;
80         for(int i = 0; i < len; i++){
81             int end = str.length() - i*WIDTH
82                 ;
83             int start = max(0, end-WIDTH);
84             sscanf(str.substr(start, end-
85                 start).c_str(), "%d", &x);
86             s.push_back(x);
87         }
88         return *this;
89     }
90     BigInteger operator + (const BigInteger&
91         b) const{
92         BigInteger c;
93         c.s.clear();
94         for(int i = 0, g = 0; i < s.size(); i++){
95             if(g == 0 && i >= s.size() && i
96                 >= b.s.size()) break;
97             int x = g;
98             if(i < s.size()) x+=s[i];
99             if(i < b.s.size()) x+=b.s[i];
100             c.s.push_back(x % BASE);
101             g = x / BASE;
102         }
103         return c;
104     }
105 };

106 ostream& operator << (ostream &out, const
107     BigInteger& x){
108     out << x.s.back();
109     for(int i = x.s.size()-2; i >= 0; i--){
110         char buf[20];
111         sprintf(buf, "%08d", x.s[i]);
112         for(int j = 0; j < strlen(buf); j++){
113             out << buf[j];
114         }
115     }
116     return out;
117 }

118 istream& operator >> (istream &in,
119     BigInteger& x){
120     string s;
121     if(!(in >> s))
122         return in;
123     x = s;
124     return in;
125 }

126 struct Trie{
127     int c[5000005][10];
128     int val[5000005];
129     int sz;
130     int getIndex(char c){
131         return c - '0';
132     }
133     void init(){
134         memset(c[0], 0, sizeof(c[0]));
135         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);
83         j++){
84         int index = getIndex(firstBuf[j]
85             );
86         if(!c[u][index]){
87             memset(c[sz], 0, sizeof(c[
88                 sz]));
89             val[sz] = v;
90             c[u][index] = sz++;
91         }
92         u = c[u][index];
93         max_len_count++;
94     }
95     for(int i = x.s.size()-2; i >= 0; i
96         --){
97         char buf[20];
98         sprintf(buf, "%08d", x.s[i]);
99         for(int j = 0; j < strlen(buf)
100             && max_len_count < 50; j++){
101             int index = getIndex(buf[j])
102                 ;
103             if(!c[u][index]){
104                 memset(c[sz], 0, sizeof
105                     (c[sz]));
106                 val[sz] = v;
107                 c[u][index] = sz++;
108             }
109             u = c[u][index];
110             max_len_count++;
111         }
112         if(max_len_count >= 50){
113             break;
114         }
115     }
116 }
117 int find(const char* s){
118     int u = 0;
119     int n = strlen(s);
120     for(int i = 0; i < n; ++i)
121     {
122         int index = getIndex(s[i]);
123         if(!c[u][index]){
124             return -1;
125         }
126         u = c[u][index];
127     }
128     return val[u];
129 }

```

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