1   Basic

1.1   A function

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

vector<int>::iterator it = lower_bound(v.
    begin(), v.end(), val);
//用binary search找大於或等於val的最小值的位
    置
vector<int>::iterator it = upper_bound(v.
    begin(), v.end(), val);
//用binary search找大於val的最小值的位置

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

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

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

1.2   Codeblock setting

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

1.3   data range

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

1.4   IO_fast

```cpp
void io()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
}
```

2   DP

2.1   3 維 DP 思路

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

2.2   Knapsack Bounded

```cpp
const int N = 100, W = 100000;
int cost[N], weight[N], number[N];
int c[W + 1];
void knapsack(int n, int w)
{
    for (int i = 0; i < n; ++i)
    {
        int num = min(number[i], w / weight[
            i]);
        for (int k = 1; num > 0; k *= 2)
        {
            if (k > num)
                k = num;
            num -= k;
            for (int j = w; j >= weight[i] *
                k; --j)
                c[j] = max(c[j], c[j -
                    weight[i] * k] + cost[i]
                    * k);
        }
    }
    cout << "Max Prince" << c[w];
}
```

2.3   Knapsack sample

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

2.4   Knapsack Unbounded

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

2.5   LCIS

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

            if (arr1[i] > arr2[j])
                if (table[j] > current)
                    current = table[j];
        }
    }
    int result = 0;
    for (int i = 0; i < m; i++)
        if (table[i] > result)
            result = table[i];
    return result;
}
```

2.6   LCS

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

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

2.7 LIS

```
1  void getMaxElementAndPos(vector<int> &LISTbl
     , vector<int> &LISLen, int tNum, int
     tlen, int tStart, int &num, int &pos)
2  {
3    int max = numeric_limits<int>::min();
4    int maxPos;
5    for (int i = tStart; i >= 0; i--)
6    {
7      if (LISLen[i] == tlen && LISTbl[i] <
           tNum)
8      {
9        if (LISTbl[i] > max)
10       {
11         max = LISTbl[i];
12         maxPos = i;
13       }
14     }
15   }
16   num = max;
17   pos = maxPos;
18 }
19 int LIS(vector<int> &LISTbl)
20 {
21   if (LISTbl.size() == 0)
22     return 0;
23   vector<int> LISLen(LISTbl.size(), 1);
24   for (int i = 1; i < LISTbl.size(); i++)
25     for (int j = 0; j < i; j++)
26       if (LISTbl[j] < LISTbl[i])
27         LISLen[i] = max(LISLen[i],
               LISLen[j] + 1);
28   int maxlen = *max_element(LISLen.begin()
         , LISLen.end());
29   int num, pos;
30   vector<int> buf;
31   getMaxElementAndPos(LISTbl, LISLen,
         numeric_limits<int>::max(), maxlen,
         LISTbl.size() - 1, num, pos);
32   buf.push_back(num);
33   for (int len = maxlen - 1; len >= 1; len
         --)
34   {
35     int tnum = num;
36     int tpos = pos;
37     getMaxElementAndPos(LISTbl, LISLen,
           tnum, len, tpos - 1, num, pos);
38     buf.push_back(num);
39   }
40   reverse(buf.begin(), buf.end());
```

2.8 LPS

```
1  void LPS(string s)
2  {
3    int maxlen = 0, l, r;
4    int n = n;
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
           length
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';
          //頭到尾
29 }
```

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);
```

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

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

3 Flow & matching

3.1 Dinic

```
1  const long long INF = 1LL<<60;
2  struct Dinic { //O(VVE), with minimum cut
3      static const int MAXN = 5003;
4      struct Edge{
5          int u, v;
6          long long cap, rest;
7      };
8      int n, m, s, t, d[MAXN], cur[MAXN];
9      vector<Edge> edges;
10     vector<int> G[MAXN];
11     void init(){
12         edges.clear();
13         for ( int i = 0 ; i < n ; i++ ) G[i
                 ].clear();
14         n = 0;
15     }
16     // min cut start
17     bool side[MAXN];
18     void cut(int u) {
19         side[u] = 1;
20         for ( int i : G[u] ) {
21             if ( !side[ edges[i].v ] &&
                     edges[i].rest )
22                 cut(edges[i].v);
23         }
24     }
25     // min cut end
26     int add_node(){
27         return n++;
28     }
29     void add_edge(int u, int v, long long
             cap){
30         edges.push_back( {u, v, cap, cap} );
31         edges.push_back( {v, u, 0, 0LL} );
32         m = edges.size();
33         G[u].push_back(m-2);
34         G[v].push_back(m-1);
35     }
36     bool bfs(){
37         fill(d,d+n,-1);
38         queue<int> que;
39         que.push(s); d[s]=0;
40         while (!que.empty()){
41             int u = que.front(); que.pop();
42             for (int ei : G[u]){
43                 Edge &e = edges[ei];
44                 if (d[e.v] < 0 && e.rest >
                         0){
45                     d[e.v] = d[u] + 1;
46                     que.push(e.v);
47                 }
48             }
49         }
50         return d[t] >= 0;
51     }
52     long long dfs(int u, long long a){
53         if ( u == t || a == 0 ) return a;
54         long long flow = 0, f;
```

```
55        for ( int &i=cur[u]; i < (int)G[u].
              size() ; i++) {
56            Edge &e = edges[ G[u][i] ];
57            if ( d[u] + 1 != d[e.v] )
                  continue;
58            f = dfs(e.v, min(a, e.rest) );
59            if ( f > 0 ) {
60                e.rest -= f;
61                edges[ G[u][i]^1 ].rest += f;
62                flow += f;
63                a -= f;
64                if ( a == 0 ) break;
65            }
66        }
67        return flow;
68    }
69    long long maxflow(int _s, int _t){
70        s = _s, t = _t;
71        long long flow = 0, mf;
72        while ( bfs() ){
73            fill(cur,cur+n,0);
74            while ( (mf = dfs(s, INF)) )
                  flow += mf;
75        }
76        return flow;
77    }
78 } dinic;
```

3.2   Edmonds__karp

```
1 /*Flow - Edmonds-karp*/
2 /*Based on UVa820*/
3 #define inf 1000000
4 int getMaxFlow(vector<vector<int>> &capacity
     , int s, int t, int n){
5   int ans = 0;
6   vector<vector<int>> residual(n+1, vector<
       int>(n+1, 0)); //residual network
7   while(true){
8     vector<int> bottleneck(n+1, 0);
9     bottleneck[s] = inf;
10    queue<int> q;
11    q.push(s);
12    vector<int> pre(n+1, 0);
13    while(!q.empty() && bottleneck[t] == 0){
14      int cur = q.front();
15      q.pop();
16      for(int i = 1; i <= n ; i++){
17        if(bottleneck[i] == 0 && capacity[
             cur][i] > residual[cur][i]){
18          q.push(i);
19          pre[i] = cur;
20          bottleneck[i] = min(bottleneck[cur
               ], capacity[cur][i] - residual
               [cur][i]);
21        }
22      }
23    }
24    if(bottleneck[t] == 0) break;
25    for(int cur = t; cur != s; cur = pre[cur
         ]){
26      residual[pre[cur]][cur] +=
             bottleneck[t];
```

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

3.3   hungarian

```
1 /*bipartite - hungarian*/
2 struct Graph{
3     static const int MAXN = 5003;
4     vector<int> G[MAXN];
5     int n, match[MAXN], vis[MAXN];
6     void init(int _n){
7         n = _n;
8         for (int i=0; i<n; i++) G[i].clear()
             ;
9     }
10    bool dfs(int u){
11        for (int v:G[u]){
12            if (vis[v]) continue;
13            vis[v]=true;
14            if (match[v]==-1 || dfs(match[v
                 ])){
15                match[v] = u;
16                match[u] = v;
17                return true;
18            }
19        }
20        return false;
21    }
22    int solve(){
23        int res = 0;
24        memset(match,-1,sizeof(match));
25        for (int i=0; i<n; i++){
26            if (match[i]==-1){
```

```
27                memset(vis,0,sizeof(vis));
28                if ( dfs(i) ) res++;
29            }
30        }
31        return res;
32    }
33 } graph;
```

3.4   Maximum__matching

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

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 = 1000000000000000LL;
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
                 != -1; i = next[i])
43            {
44                if (vi[v[i]] != tim && cap[i
                     ] > flow[i])
45                {
46                    vi[v[i]] = tim;
47                    dis[v[i]] = dis[que[head
                         ]] + 1;
48                    que[++tail] = v[i];
49                }
50            }
51            ++head;
52        }
53        return vi[t] == tim;
54    }
55    ll dfs(int x, ll a)
56    {
57        if (x == t || a == 0)
58            return a;
59        ll flw = 0, f;
```

```cpp
        int &i = cur[x];
        for (i = first[x]; i != -1; i = next[i])
        {
            if (dis[x] + 1 == dis[v[i]] && (
                f = dfs(v[i], min(a, cap[i]
                - flow[i]))) > 0)
            {
                flow[i] += f;
                flow[i ^ 1] -= f;
                a -= f;
                flw += f;
                if (a == 0)
                    break;
            }
        }
        return flw;
    }
    ll MaxFlow(int s, int t)
    {
        this->s = s;
        this->t = t;
        ll flw = 0;
        while (bfs())
        {
            for (int i = 1; i <= n; ++i)
                cur[i] = 0;
            flw += dfs(s, oo);
        }
        return flw;
    }
};
// MF Net;
// Net.n = n;
// Net.Clear();
// a 到 b (注意從1開始!!!!)
// Net.Add(a, b, w, 0);
// Net.MaxFlow(s, d);
// s 到 d 的 MF
```

4 Geometry

4.1 Closest Pair

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

4.2 Line

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

4.3 Point

```cpp
template <typename T>
struct point
{
    T x, y;
    point() {}
    point(const T &x, const T &y) : x(x), y(
        y) {}
    point operator+(const point &b) const
    {
        return point(x + b.x, y + b.y);
    }
    point operator-(const point &b) const
    {
        return point(x - b.x, y - b.y);
    }
    point operator*(const T &b) const
    {
        return point(x * b, y * b);
    }
    point operator/(const T &b) const
    {
        return point(x / b, y / b);
    }
    bool operator==(const point &b) const
    {
        return x == b.x && y == b.y;
    }
    T dot(const point &b) const
    {
        return x * b.x + y * b.y;
    }
```

```cpp
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)), dot(b)));
46     }
47     T getA() const
48     {
                       //對x軸的弧度
49         T A = atan2(y, x); //超過180度會變負的
50         if (A <= -PI / 2)
51             A += PI * 2;
52         return A;
53     }
54 };
```

4.4 Polygon

```cpp
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 < (int)p.size(); i = j++)
10             ans += p[i].cross(p[j]);
11         return ans / 2;
12     }
13     point<T> center_of_mass() const
14     { //重心
15         T cx = 0, cy = 0, w = 0;
16         for (int i = p.size() - 1, j = 0; j < (int)p.size(); i = j++)
17         {
18             T a = p[i].cross(p[j]);
19             cx += (p[i].x + p[j].x) * a;
20             cy += (p[i].y + p[j].y) * a;
21             w += a;
22         }
23         return point<T>(cx / 3 / w, cy / 3 / w);
24     }
25     char ahas(const point<T> &t) const
26     { //點是否在簡單多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
27         bool c = 0;
28         for (int i = 0, j = p.size() - 1; i < p.size(); j = i++)
29             if (line<T>(p[i], p[j]).point_on_segment(t))
30                 return -1;
31             else if ((p[i].y > t.y) != (p[j].y > t.y) &&
32                      t.x < (p[j].x - p[i].x) * (t.y - p[i].y) / (p[j].y - p[i].y) + p[i].x)
33                 c = !c;
34         return c;
35     }
36     char point_in_convex(const point<T> &x) const
37     {
38         int l = 1, r = (int)p.size() - 2;
39         while (l <= r)
40         { //點是否在凸多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
41             int mid = (l + r) / 2;
42             T a1 = (p[mid] - p[0]).cross(x - p[0]);
43             T a2 = (p[mid + 1] - p[0]).cross(x - p[0]);
44             if (a1 >= 0 && a2 <= 0)
45             {
46                 T res = (p[mid + 1] - p[mid]).cross(x - p[mid]);
47                 return res > 0 ? 1 : (res >= 0 ? -1 : 0);
48             }
49             else if (a1 < 0)
50                 r = mid - 1;
51             else
52                 l = mid + 1;
53         }
54         return 0;
55     }
56     vector<T> getA() const
57     {//凸包邊對x軸的夾角
58         vector<T> res; //一定是遞增的
59         for (size_t i = 0; i < p.size(); ++i)
60             res.push_back((p[(i + 1) % p.size()] - p[i]).getA());
61         return res;
62     }
63     bool line_intersect(const vector<T> &A, const line<T> &l) const
64     { //O(logN)
65         int f1 = upper_bound(A.begin(), A.end(), (l.p1 - l.p2).getA()) - A.begin();
66         int f2 = upper_bound(A.begin(), A.end(), (l.p2 - l.p1).getA()) - A.begin();
67         return l.cross_seg(line<T>(p[f1], p[f2]));
68     }
69     polygon cut(const line<T> &l) const
70     { //凸包對直線切割，得到直線l左側的凸包
71         polygon ans;
72         for (int n = p.size(), i = n - 1, j = 0; j < n; i = j++)
73         {
74             if (l.ori(p[i]) >= 0)
75             {
76                 ans.p.push_back(p[i]);
77                 if (l.ori(p[j]) < 0)
78                     ans.p.push_back(l.line_intersection(line<T>(p[i], p[j])));
79             }
80             else if (l.ori(p[j]) > 0)
81                 ans.p.push_back(l.line_intersection(line<T>(p[i], p[j])));
82         }
83         return ans;
84     }
85     static bool graham_cmp(const point<T> &a, const point<T> &b)
86     { //凸包排序函數  // 起始點不同
87         // return (a.x < b.x) || (a.x == b.x && a.y < b.y);  //最左下角開始
88         return (a.y < b.y) || (a.y == b.y && a.x < b.x);  //Y最小開始
89     }
90     void graham(vector<point<T>> &s)
91     { //凸包 Convexhull 2D
92         sort(s.begin(), s.end(), graham_cmp);
93         p.resize(s.size() + 1);
94         int m = 0;
95         // cross >= 0 順時針。cross <= 0 逆時針旋轉
96         for (size_t i = 0; i < s.size(); ++i)
97         {
98             while (m >= 2 && (p[m - 1] - p[m - 2]).cross(s[i] - p[m - 2]) <= 0)
99                 --m;
100            p[m++] = s[i];
101        }
102        for (int i = s.size() - 2, t = m + 1; i >= 0; --i)
103        {
104            while (m >= t && (p[m - 1] - p[m - 2]).cross(s[i] - p[m - 2]) <= 0)
105                --m;
106            p[m++] = s[i];
107        }
108        if (s.size() > 1) // 重複頭一次需扣掉
109            --m;
110        p.resize(m);
111    }
112    T diam()
113    { //直徑
114        int n = p.size(), t = 1;
115        T ans = 0;
116        p.push_back(p[0]);
117        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 &pl)
151    { //凸包最近距離平方
152        vector<point<T>> &P = p, &Q = pl.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;
```

```cpp
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);
             //線段左側為該線段半平面
183        int L, R, n = s.size();
184        vector<point<T>> px(n);
185        vector<line<T>> q(n);
186        q[L = R = 0] = s[0];
187        for (int i = 1; i < n; ++i)
188        {
189            while (L < R && s[i].ori(px[R -
                 1]) <= 0)
190                --R;
191            while (L < R && s[i].ori(px[L])
                 <= 0)
192                ++L;
193            q[++R] = s[i];
194            if (q[R].parallel(q[R - 1]))
195            {
196                --R;
197                if (q[R].ori(s[i].p1) > 0)
198                    q[R] = s[i];
199            }
200            if (L < R)
201                px[R - 1] = q[R - 1].
                     line_intersection(q[R]);
202        }
203        while (L < R && q[L].ori(px[R - 1])
             <= 0)
204            --R;
205        p.clear();
206        if (R - L <= 1)
207            return 0;
208        px[R] = q[R].line_intersection(q[L])
             ;
209        for (int i = L; i <= R; ++i)
210            p.push_back(px[i]);
211        return R - L + 1;
212    }
213 };
```

4.5   Triangle

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

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

5   Graph

5.1   Bellman-Ford

```cpp
1 /*SPA - Bellman-Ford*/
2 #define inf 99999 //define by you maximum
    edges weight
3 vector<vector<int> > edges;
4 vector<int> dist;
5 vector<int> ancestor;
6 void BellmanFord(int start,int node){
7     dist[start] = 0;
8     for(int it = 0; it < node-1; it++){
9         for(int i = 0; i < node; i++){
10            for(int j = 0; j < node; j++){
11                if(edges[i][j] != -1){
12                    if(dist[i] + edges[i][j]
                         < dist[j]){
13                        dist[j] = dist[i] +
                             edges[i][j];
14                        ancestor[j] = i;
```

```cpp
15                    }
16                }
17            }
18        }
19    }
20    for(int i = 0; i < node; i++)  //
         negative cycle detection
21        for(int j = 0; j < node; j++)
22            if(dist[i] + edges[i][j] < dist[
                 j])
23            {
24                cout<<"Negative cycle!"<<
                     endl;
25                return;
26            }
27 }
28 int main(){
29    int node;
30    cin>>node;
31    edges.resize(node,vector<int>(node,inf))
         ;
32    dist.resize(node,inf);
33    ancestor.resize(node,-1);
34    int a,b,d;
35    while(cin>>a>>b>>d){
36        /*input: source destination weight*/
37        if(a == -1 && b == -1 && d == -1)
38            break;
39        edges[a][b] = d;
40    }
41    int start;
42    cin>>start;
43    BellmanFord(start,node);
44    return 0;
45 }
```

5.2   BFS-queue

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

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

5.3   DFS-rec

```cpp
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.
         end(); iter++){
9         if((*iter).first.first == start &&
             (*iter).second == 0 && pass[(*
             iter).first.second] == 0){
10            route.push_back((*iter).first.
                 second);
11            DFS((*iter).first.second);
12        }
```

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

5.4 Dijkstra

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

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

5.5 Euler circuit

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

5.6 Floyd-warshall

```
1    /*SPA - Floyd-Warshall*/
2    #define inf 99999
3    void floyd_warshall(vector<vector<int>>&
        distance, vector<vector<int>>& ancestor,
        int n){
4        for (int k = 0; k < n; k++){
5            for (int i = 0; i < n; i++){
6                for (int j = 0; j < n; j++){
7                    if(distance[i][k] + distance
                        [k][j] < distance[i][j])
                        {
8                        distance[i][j] =
                            distance[i][k] +
                            distance[k][j];
9                        ancestor[i][j] =
                            ancestor[k][j];
10                   }
11               }
12           }
13       }
14    }
15    int main(){
16        int n;
17        cin >> n;
```

```
18        int a, b, d;
19        vector<vector<int>> distance(n, vector<
            int>(n,99999));
20        vector<vector<int>> ancestor(n, vector<
            int>(n,-1));
21        while(cin>>a>>b>>d){
22            if(a == -1 && b == -1 && d == -1)
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!" <<
                    endl;
34                break;
35            }
36        }
37        return 0;
38    }
```

5.7 Hamilton_cycle

```
1    /*find hamilton cycle*/
2    void hamilton(vector<vector<int>> gp, int k,
        int cur, vector<int>& solution,vector<
        bool> pass,bool& flag){
3        if(k == gp.size()-1){
4            if(gp[cur][1] == 1){
5                cout << 1 << " ";
6                while(cur != 1){
7                    cout << cur << " ";
8                    cur = solution[cur];
9                }
10               cout << cur << endl;
11               flag = true;
12               return;
13           }
14       }
15       for (int i = 0; i < gp[cur].size() && !
            flag; i++){
16           if(gp[cur][i] == 1 && !pass[i]){
17               pass[i] = true;
18               solution[i] = cur;
19               hamilton(gp, k + 1, i, solution,
                    pass,flag);
20               pass[i] = false;
21           }
22       }
23    }
24    int main(){
25        int n;
26        while(cin>>n){
27            int a,b;
28            bool end = false;
29            vector<vector<int>> gp(n+1,vector<
                int>(n+1,0));
30            while(cin>>a>>b){
31                if(a == 0 && b == 0)
```

```
32                    break;
33                gp[a][b] = 1;
34                gp[b][a] = 1;
35            }
36            vector<int> solution(n + 1, -1);
37            vector<bool> pass(n + 1, false);
38            solution[1] = 0;
39            pass[1] = true;
40            bool flag = false;
41            hamilton(gp, 1,1 ,solution,pass,flag
                );
42            if(!flag)
43                cout << "N" << endl;
44        }
45        return 0;
46    }
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            return a.weight > b.weight;
8        }
9    };
10   int find(int x,vector<int>& union_set){
11       if(x != union_set[x])
12           union_set[x] = find(union_set[x],
                union_set);
13       return union_set[x];
14   }
15   void merge(int a,int b,vector<int>&
        union_set){
16       int pa = find(a, union_set);
17       int pb = find(b, union_set);
18       if(pa != pb)
19           union_set[pa] = pb;
20   }
21   void kruskal(priority_queue<edges> pq,int n)
        {
22       vector<int> union_set(n, 0);
23       for (int i = 0; i < n; i++)
24           union_set[i] = i;
25       int edge = 0;
26       int cost = 0; //evaluate cost of mst
27       while(!pq.empty() && edge < n - 1){
28           edges cur = pq.top();
29           int from = find(cur.from, union_set)
                ;
```

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

5.9 Prim

```
1  /*mst - Prim*/
2  #define inf 99999
3  struct edges{
4      int from;
5      int to;
6      int weight;
7      friend bool operator < (edges a, edges b
           ){
8          return a.weight > b.weight;
9      }
10 };
11 void Prim(vector<vector<int>> gp,int n,int
       start){
12     vector<bool> pass(n,false);
13     int edge = 0;
14     int cost = 0; //evaluate cost of mst
15     priority_queue<edges> pq;
16     for (int i = 0; i < n; i++){
17         if(gp[start][i] != inf){
18             edges tmp;
19             tmp.from = start;
20             tmp.to = i;
21             tmp.weight = gp[start][i];
22             pq.push(tmp);
23         }
24     }
25     pass[start] = true;
26     while(!pq.empty() && edge < n-1){
27         edges cur = pq.top();
28         pq.pop();
```

```
29         if(!pass[cur.to]){
30             for (int i = 0; i < n; i++){
31                 if(gp[cur.to][i] != inf){
32                     edges tmp;
33                     tmp.from = cur.to;
34                     tmp.to = i;
35                     tmp.weight = gp[cur.to][
                           i];
36                     pq.push(tmp);
37                 }
38             }
39             pass[cur.to] = true;
40             edge += 1;
41             cost += cur.weight;
42         }
43     }
44     if(edge < n-1)
45         cout << "No mst" << endl;
46     else
47         cout << cost << endl;
48 }
49 int main(){
50     int n;
51     cin >> n;
52     int a, b, d;
53     vector<vector<int>> gp(n,vector<int>(n,
           inf));
54     while(cin>>a>>b>>d){
55         if(a == -1 && b == -1 && d == -1)
56             break;
57         if(gp[a][b] > d)
58             gp[a][b] = d;
59     }
60     Prim(gp,n,0);
61     return 0;
62 }
```

5.10 Union_find

```
1  int find(int x, vector<int> &union_set)
2  {
3      if (union_set[x] != x)
4          union_set[x] = find(union_set[x],
               union_set); //compress path
5      return union_set[x];
6  }
7  void merge(int x, int y, vector<int> &
       union_set, vector<int> &rank)
8  {
9      int rx, ry;
10     rx = find(x, union_set);
11     ry = find(y, union_set);
12     if (rx == ry)
13         return;
14     /*merge by rank -> always merge small
           tree to big tree*/
15     if (rank[rx] > rank[ry])
16         union_set[ry] = rx;
17     else
18     {
19         union_set[rx] = ry;
20         if (rank[rx] == rank[ry])
21             ++rank[ry];
```

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

6 Mathematics

6.1 Combination

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

6.2 Extended Euclidean

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

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

6.3 Hex to Dec

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

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

6.4 log

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

6.5 Mod

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

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

6.6 Permutation

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

6.7 PI

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

6.8 Prime table

```
1 const int maxn = sqrt(INT_MAX);
2 vector<int> p;
3 bitset<maxn> is_notp;
4 void PrimeTable()
5 {
6     is_notp.reset();
7     is_notp[0] = is_notp[1] = 1;
8     for (int i = 2; i <= maxn; ++i)
9     {
10         if (!is_notp[i])
11             p.push_back(i);
12         for (int j = 0; j < (int)p.size();
             ++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 }
```

6.9 primeBOOL

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

6.10 Round(小數)

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

6.11 二分逼近法

```
1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10         else
11             R = M;
12     }
13     printf("%.3lf\n", R);
14 }
```

6.12 四則運算

```
1 string s = ""; //開頭是負號要補0
2 long long int DFS(int le, int ri) // (0,
     string final index)
3 {
4     int c = 0;
5     for (int i = ri; i >= le; i--)
6     {
7         if (s[i] == ')')
8             c++;
9         if (s[i] == '(')
10             c--;
11         if (s[i] == '+' && c == 0)
12             return DFS(le, i - 1) + DFS(i +
                 1, ri);
```

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

6.13 數字乘法組合

```
1  void dfs(int j, int old, int num, vector<int
      > com, vector<vector<int>> &ans)
2  {
3      for (int i = j; i <= sqrt(num); i++)
4      {
5          if (old == num)
6              com.clear();
7          if (num % i == 0)
8          {
9              vector<int> a;
10             a = com;
11             a.push_back(i);
12             finds(i, old, num / i, a, ans);
13             a.push_back(num / i);
14             ans.push_back(a);
15         }
16     }
17 }
18 vector<vector<int>> ans;
19 vector<int> zero;
20 dfs(2, num, num, zero, ans);
```

```
21 /*/num 為 input 數字*/
22 for (int i = 0; i < ans.size(); i++)
23 {
24     for (int j = 0; j < ans[i].size() - 1; j
         ++)
25         cout << ans[i][j] << " ";
26     cout << ans[i][ans[i].size() - 1] <<
         endl;
27 }
```

6.14 數字加法組合

```
1  void recur(int i, int n, int m, vector<int>
      &out, vector<vector<int>> &ans)
2  {
3      if (n == 0)
4      {
5          for (int i : out)
6              if (i > m)
7                  return;
8          ans.push_back(out);
9      }
10     for (int j = i; j <= n; j++)
11     {
12         out.push_back(j);
13         recur(j, n - j, m, out, ans);
14         out.pop_back();
15     }
16 }
17 vector<vector<int>> ans;
18 vector<int> zero;
19 recur(1, num, num, zero, ans);
20 // num 為 input 數字
21 for (int i = 0; i < ans.size(); i++)
22 {
23     for (int j = 0; j < ans[i].size() - 1; j
         ++)
24         cout << ans[i][j] << " ";
25     cout << ans[i][ans[i].size() - 1] <<
         endl;
26 }
```

6.15 羅馬數字

```
1  int romanToInt(string s)
2  {
3      unordered_map<char, int> T;
4      T['I'] = 1;
5      T['V'] = 5;
6      T['X'] = 10;
7      T['L'] = 50;
8      T['C'] = 100;
9      T['D'] = 500;
10     T['M'] = 1000;
11
12     int sum = T[s.back()];
13     for (int i = s.length() - 2; i >= 0; --i
         )
14     {
```

```
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19     }
20     return sum;
21 }
```

6.16 質因數分解

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

7 Other

7.1 binary search 三類變化

```
1  // 查找和目標值完全相等的數
2  int find(vector<int> &nums, int target)
3  {
4      int left = 0, right = nums.size();
5      while (left < right)
6      {
7          int mid = left + (right - left) / 2;
8          if (nums[mid] == target)
9              return mid;
10         else if (nums[mid] < target)
11             left = mid + 1;
12         else
13             right = mid;
14     }
15     return -1;
16 }
17 // 找第一個不小於目標值的數 == 找最後一個小
       於目標值的數
18 /*(lower_bound)*/
19 int find(vector<int> &nums, int target)
20 {
21     int left = 0, right = nums.size();
22     while (left < right)
23     {
24         int mid = left + (right - left) / 2;
25         if (nums[mid] < target)
```

```
26             left = mid + 1;
27         else
28             right = mid;
29     }
30     return right;
31 }
32 // 找第一個大於目標值的數 == 找最後一個不大
       於目標值的數
33 /*(upper_bound)*/
34 int find(vector<int> &nums, int target)
35 {
36     int left = 0, right = nums.size();
37     while (left < right)
38     {
39         int mid = left + (right - left) / 2;
40         if (nums[mid] <= target)
41             left = mid + 1;
42         else
43             right = mid;
44     }
45     return right;
46 }
```

7.2 heap sort

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

7.3 Merge sort

```cpp
void Merge(vector<int> &arr, int front, int
    mid, int end)
{
    vector<int> LeftSub(arr.begin() + front,
        arr.begin() + mid + 1);
    vector<int> RightSub(arr.begin() + mid +
        1, arr.begin() + end + 1);
    LeftSub.insert(LeftSub.end(), INT_MAX);
    RightSub.insert(RightSub.end(), INT_MAX)
        ;
    int idxLeft = 0, idxRight = 0;

    for (int i = front; i <= end; i++)
    {

        if (LeftSub[idxLeft] <= RightSub[
            idxRight])
        {
            arr[i] = LeftSub[idxLeft];
            idxLeft++;
        }
        else
        {
            arr[i] = RightSub[idxRight];
            idxRight++;
        }
    }
}
void MergeSort(vector<int> &arr, int front,
    int end)
{
    // front = 0 , end = arr.size() - 1
    if (front < end)
    {
        int mid = (front + end) / 2;
        MergeSort(arr, front, mid);
        MergeSort(arr, mid + 1, end);
        Merge(arr, front, mid, end);
    }
}
```

7.4 python

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

print(*objects, sep=' ', end='\n', file=sys.
    stdout, flush=False)
objects -- /*表示可以一次輸出多個對象。輸出
    多個對象時，需要使用、複數請求。*/
sep -- /*分開多個對象，默認值是一個整體。*/
end -- /*默認值是換行符 '\n'，我們可以使用其
    他字符串。*/
file -- /*要寫入的文件對象。*/
flush -- /*輸出是否被刷新通常決定於文件，但
    如果flush關鍵字參數為True，流會被強制刷
```

新。*/

7.5 Quick

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

7.6 Weighted Job Scheduling

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

    return result;
}
```

7.7 數獨解法

```cpp
int getSquareIndex(int row, int column, int
    n)
{
    return row / n * n + column / n;
}

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

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

    for (int i = 1; i <= n2; i++)
    {
        if (!rows[rowNum][i] && !cols[colNum
            ][i] && !boxs[getSquareIndex(
            rowNum, colNum, n)][i])
        {
            rows[rowNum][i] = true;
            cols[colNum][i] = true;
            boxs[getSquareIndex(rowNum,
                colNum, n)][i] = true;
            board[rowNum][colNum] = i;
            if (backtracking(board, rows,
                cols, boxs, index + 1, n))
                return true;
            board[rowNum][colNum] = 0;
            rows[rowNum][i] = false;
            cols[colNum][i] = false;
            boxs[getSquareIndex(rowNum,
                colNum, n)][i] = false;
        }
    }
    return false;
}
/*用法 main*/
int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
```

```cpp
vector<vector<int>> board(n * n + 1, vector<
    int>(n * n + 1, 0));
vector<vector<bool>> isRow(n * n + 1, vector
    <bool>(n * n + 1, false));
vector<vector<bool>> isColumn(n * n + 1,
    vector<bool>(n * n + 1, false));
vector<vector<bool>> isSquare(n * n + 1,
    vector<bool>(n * n + 1, false));

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

8 String

8.1 KMP

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

    for (int i = 0, q = 0; i < n; i++)
    {
        while (q > 0 && pattern[q] != text[i
            ])
```

```
26          q = next[q];
27      if (pattern[q] == text[i])
28          q++;
29      if (q == m)
30      {
31          cout << "Pattern occurs with
                shift " << i - m + 1 << endl
                ;
32          q = 0;
33      }
34  }
35 }
36 // string s = "abcdabcdebcd";
37 // string p = "bcd";
38 // KMPMatcher(s, p);
39 // cout << endl;
```

8.2   Min Edit Distance

```
1 int EditDistance(string a, string b)
2 {
3     vector<vector<int>> dp(a.size() + 1,
           vector<int>(b.size() + 1, 0));
4     int m = a.length(), n = b.length();
5     for (int i = 0; i < m + 1; i++)
6     {
7         for (int j = 0; j < n + 1; j++)
8         {
9             if (i == 0)
10                dp[i][j] = j;
11            else if (j == 0)
12                dp[i][j] = i;
13            else if (a[i - 1] == b[j - 1])
14                dp[i][j] = dp[i - 1][j - 1];
15            else
16                dp[i][j] = 1 + min(min(dp[i
                    - 1][j], dp[i][j - 1]),
                    dp[i - 1][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.
           substr(minStart, minLength);
25 }
```

8.4   Split

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

9   data structure

9.1   Bigint

```
1 //台大
2 struct Bigint
3 {
4     static const int LEN = 60;        //
           maxLEN
5     static const int BIGMOD = 10000; //10為
           正常位數
6     int s;
7     int vl, v[LEN];
8     //  vector<int> v;
9     Bigint() : s(1) { vl = 0; }
10    Bigint(long long a)
11    {
12        s = 1;
13        vl = 0;
14        if (a < 0)
15        {
16            s = -1;
17            a = -a;
18        }
19        while (a)
20        {
21            push_back(a % BIGMOD);
22            a /= BIGMOD;
23        }
24    }
25    Bigint(string str)
26    {
27        s = 1;
28        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 =
               1; i >= stPos; i--)
36        {
37            num += (str[i] - '0') * q;
38            if ((q *= 10) >= BIGMOD)
39            {
40                push_back(num);
41                num = 0;
42                q = 1;
43            }
44        }
45        if (num)
46            push_back(num);
47        n();
48    }
49    int len() const
50    {
51        return vl; //return SZ(v);
52    }
53    bool empty() const { return len() == 0;
          }
54    void push_back(int x)
55    {
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),
               0);
75    }
76    void print() const
77    {
78        if (empty())
79        {
80            putchar('0');
81            return;
82        }
83        if (s == -1)
84            putchar('-');
85        printf("%d", back());
86        for (int i = len() - 2; i >= 0; i--)
87            printf("%.4d", v[i]);
88    }
89    friend std::ostream &operator<<(std::
           ostream &out, const Bigint &a)
90    {
91        if (a.empty())
92        {
93            out << "0";
94            return out;
95        }
96        if (a.s == -1)
97            out << "-";
98        out << a.back();
99        for (int i = a.len() - 2; i >= 0; i
               --)
100       {
101           char str[10];
102           snprintf(str, 5, "%.4d", a.v[i])
                  ;
103           out << str;
104       }
105       return out;
106   }
107   int cp3(const Bigint &b) const
108   {
109       if (s != b.s)
110           return s - b.s;
111       if (s == -1)
112           return -(-*this).cp3(-b);
113       if (len() != b.len())
114           return len() - b.len(); //int
115       for (int i = len() - 1; i >= 0; i--)
116           if (v[i] != b.v[i])
117               return v[i] - b.v[i];
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;
```

```cpp
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] /
                      BIGMOD;
168              r.v[i] %= BIGMOD;
169          }
170      }
171      r.n();
172      return r;
173  }
174  Bigint operator-(const Bigint &b) const
175  {
176      if (s == -1)
177          return -(-(*this) - (-b));
178      if (b.s == -1)
179          return (*this) + (-b);
180      if ((*this) < b)
181          return -(b - (*this));
182      Bigint r;
183      r.resize(len());
184      for (int i = 0; i < len(); i++)
185      {
186          r.v[i] += v[i];
187          if (i < b.len())
188              r.v[i] -= b.v[i];
189          if (r.v[i] < 0)
190          {
191              r.v[i] += BIGMOD;
192              r.v[i + 1]--;
193          }
194      }
195      r.n();
196      return r;
197  }
198  Bigint operator*(const Bigint &b) const
199  {
200      Bigint r;
201      r.resize(len() + b.len() + 1);
202      r.s = s * b.s;
203      for (int i = 0; i < len(); i++)
204      {
205          for (int j = 0; j < b.len(); j
                  ++)
206          {
207              r.v[i + j] += v[i] * b.v[j];
208              if (r.v[i + j] >= BIGMOD)
209              {
210                  r.v[i + j + 1] += r.v[i
                          + j] / BIGMOD;
211                  r.v[i + j] %= BIGMOD;
212              }
213          }
214      }
215      r.n();
216      return r;
217  }
218  Bigint operator/(const Bigint &b)
219  {
220      Bigint r;
221      r.resize(max(1, len() - b.len() + 1)
              );
222      int oriS = s;
223      Bigint b2 = b; // b2 = abs(b)
224      s = b2.s = r.s = 1;
225      for (int i = r.len() - 1; i >= 0; i
              --)
226      {
227          int d = 0, u = BIGMOD - 1;
228          while (d < u)
229          {
230              int m = (d + u + 1) >> 1;
231              r.v[i] = m;
232              if ((r * b2) > (*this))
233                  u = m - 1;
234              else
235                  d = m;
236          }
237          r.v[i] = d;
238      }
239      s = oriS;
240      r.s = s * b.s;
241      r.n();
242      return r;
243  }
244  Bigint operator%(const Bigint &b)
245  {
246      return (*this) - (*this) / b * b;
247  }
248  };
```

9.2 matirx

```cpp
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,
            rt(c)) {}
10      Matrix(mt a) { m = a, r = a.size(), c =
            a[0].size(); }
11      rt &operator[](int i) { return m[i]; }
12      matrix operator+(const matrix &a)
13      {
14          matrix rev(r, c);
15          for (int i = 0; i < r; ++i)
16              for (int j = 0; j < c; ++j)
17                  rev[i][j] = m[i][j] + a.m[i
                      ][j];
18          return rev;
19      }
20      matrix operator-(const matrix &a)
21      {
22          matrix rev(r, c);
23          for (int i = 0; i < r; ++i)
24              for (int j = 0; j < c; ++j)
25                  rev[i][j] = m[i][j] - a.m[i
                      ][j];
26          return rev;
27      }
28      matrix operator*(const matrix &a)
29      {
30          matrix rev(r, a.c);
31          matrix tmp(a.c, a.r);
32          for (int i = 0; i < a.r; ++i)
33              for (int j = 0; j < a.c; ++j)
34                  tmp[j][i] = a.m[i][j];
35          for (int i = 0; i < r; ++i)
36              for (int j = 0; j < a.c; ++j)
37                  for (int k = 0; k < c; ++k)
38                      rev.m[i][j] += m[i][k] *
                          tmp[j][k];
39          return rev;
40      }
41      bool inverse() //逆矩陣判斷
42      {
43          Matrix t(r, r + c);
44          for (int y = 0; y < r; y++)
45          {
46              t.m[y][c + y] = 1;
47              for (int x = 0; x < c; ++x)
48                  t.m[y][x] = m[y][x];
49          }
50          if (!t.gas())
51              return false;
52          for (int y = 0; y < r; y++)
53              for (int x = 0; x < c; ++x)
54                  m[y][x] = t.m[y][c + x] / t.
                      m[y][y];
55          return true;
56      }
57      T gas() //行列式
58      {
59          vector<T> lazy(r, 1);
60          bool sign = false;
61          for (int i = 0; i < r; ++i)
62          {
63              if (m[i][i] == 0)
64              {
65                  int j = i + 1;
66                  while (j < r && !m[j][i])
67                      j++;
68                  if (j == r)
69                      continue;
70                  m[i].swap(m[j]);
71                  sign = !sign;
72              }
73              for (int j = 0; j < r; ++j)
74              {
75                  if (i == j)
76                      continue;
77                  lazy[j] = lazy[j] * m[i][i];
78                  T mx = m[j][i];
79                  for (int k = 0; k < c; ++k)
80                      m[j][k] = m[j][k] * m[i
                          ][i] - m[i][k] * mx;
81              }
82          }
83          T det = sign ? -1 : 1;
84          for (int i = 0; i < r; ++i)
85          {
86              det = det * m[i][i];
87              det = det / lazy[i];
88              for (auto &j : m[i])
89                  j /= lazy[i];
90          }
91          return det;
92      }
93  };
```

9.3 Trie

```cpp
1   // biginter字典數
2   struct BigInteger{
3       static const int BASE = 100000000;
4       static const int WIDTH = 8;
5       vector<int> s;
6       BigInteger(long long num = 0){
7           *this = num;
8       }
9       BigInteger operator = (long long num){
10          s.clear();
11          do{
12              s.push_back(num % BASE);
13              num /= BASE;
14          }while(num > 0);
15          return *this;
16      }
17      BigInteger operator = (const string& str
            ){
18          s.clear();
19          int x, len = (str.length() - 1) /
                WIDTH + 1;
20          for(int i = 0; i < len;i++){
21              int end = str.length() - i*WIDTH
                    ;
22              int start = max(0, end-WIDTH);
23              sscanf(str.substr(start, end-
                    start).c_str(), "%d", &x);
24              s.push_back(x);
25          }
26          return *this;
27      }
28
29      BigInteger operator + (const BigInteger&
            b) const{
30          BigInteger c;
31          c.s.clear();
32          for(int i = 0, g = 0;;i++){
33              if(g == 0 && i >= s.size() && i
                    >= b.s.size()) break;
34              int x = g;
```

```
35            if(i < s.size()) x+=s[i];
36            if(i < b.s.size()) x+=b.s[i];
37            c.s.push_back(x % BASE);
38            g = x / BASE;
39        }
40        return c;
41    }
42 };
43
44 ostream& operator << (ostream &out, const
       BigInteger& x){
45     out << x.s.back();
46     for(int i = x.s.size()-2; i >= 0;i--){
47         char buf[20];
48         sprintf(buf, "%08d", x.s[i]);
49         for(int j = 0; j< strlen(buf);j++){
50             out << buf[j];
51         }
52     }
53     return out;
54 }
55
56 istream& operator >> (istream &in,
       BigInteger& x){
57     string s;
58     if(!(in >> s))
59         return in;
60     x = s;
61     return in;
62 }
63
64 struct Trie{
65     int c[5000005][10];
66     int val[5000005];
67     int sz;
68     int getIndex(char c){
69         return c - '0';
70     }
71     void init(){
72         memset(c[0], 0, sizeof(c[0]));
73         memset(val, -1, sizeof(val));
74         sz = 1;
75     }
76     void insert(BigInteger x, int v){
77         int u = 0;
78         int max_len_count = 0;
79         int firstNum = x.s.back();
80         char firstBuf[20];
81         sprintf(firstBuf, "%d", firstNum);
82         for(int j = 0; j < strlen(firstBuf);
            j++){
83             int index = getIndex(firstBuf[j
                ]);
84             if(!c[u][index]){
85                 memset(c[sz], 0 , sizeof(c[
                    sz]));
86                 val[sz] = v;
87                 c[u][index] = sz++;
88             }
89             u = c[u][index];
90             max_len_count++;
91         }
92         for(int i = x.s.size()-2; i >= 0;i
            --){
93             char buf[20];
94             sprintf(buf, "%08d", x.s[i]);
```

```
95             for(int j = 0; j < strlen(buf)
                && max_len_count < 50;j++){
96                 int index = getIndex(buf[j])
                    ;
97                 if(!c[u][index]){
98                     memset(c[sz], 0 , sizeof
                        (c[sz]));
99                     val[sz] = v;
100                    c[u][index] = sz++;
101                }
102                u = c[u][index];
103                max_len_count++;
104            }
105            if(max_len_count >= 50){
106                break;
107            }
108        }
109    }
110    int find(const char* s){
111        int u = 0;
112        int n = strlen(s);
113        for(int i = 0 ; i < n;++i)
114        {
115            int index = getIndex(s[i]);
116            if(!c[u][index]){
117                return -1;
118            }
119            u = c[u][index];
120        }
121        return val[u];
122    }
123 }
```

9.4 分數

```
1 typedef long long ll;
2 struct fraction
3 {
4   ll n, d;
5   fraction(const ll &_n = 0, const ll &_d =
        1) : n(_n), d(_d)
6   {
7     ll t = __gcd(n, d);
8     n /= t, d /= t;
9     if (d < 0)
10      n = -n, d = -d;
11  }
12  fraction operator-() const
13  {
14    return fraction(-n, d);
15  }
16  fraction operator+(const fraction &b)
        const
17  {
18    return fraction(n * b.d + b.n * d, d * b
          .d);
19  }
20  fraction operator-(const fraction &b)
        const
21  {
22    return fraction(n * b.d - b.n * d, d * b
          .d);
23  }
```

```
24  fraction operator*(const fraction &b)
        const
25  {
26    return fraction(n * b.n, d * b.d);
27  }
28  fraction operator/(const fraction &b)
        const
29  {
30    return fraction(n * b.d, d * b.n);
31  }
32  void print()
33  {
34    cout << n;
35    if (d != 1)
36      cout << "/" << d;
37  }
38 };
```

To do writing not thinking

Contents