

# 1 Basic

## 1.1 Basic 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
```

## 1.2 Basic 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
```

## 1.3 Code Template

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef unsigned long long ull;
5 #define pb push_back
6 #define len length()
7 #define all(p) p.begin(), p.end()
8 #define endl '\n'
9 #define x first
10 #define y second
11 #define bug(k) cout << "value of " << #k <<
12   " is " << k << endl;
13 #define bugarr(k) \
14   for (auto i : k) \
15     cout << i << ' '; \
16   cout << endl;
17 int main()
18 {
19   ios::sync_with_stdio(0);
20   cin.tie(0);
21   return 0;
22 }
```

## 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
9   三個 int 變數
10
11 pow(a, b, c) // a ^ b % c
12
13 print(*objects, sep = ' ', end = '\n')
14 // objects -- 可以一次輸出多個對象
15 // sep -- 分開多個objects
16 // end -- 默認值是\n
17
18 // EOF break
19 try:
20   while True:
21     //input something
22 except EOFError:
23   pass
```

## 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
6   9223372036854775807)
7 unsigned long long (0 to
8   18446744073709551615)
```

## 1.6 Some Function

```
1 round(double f); // 四捨五入
2 ceil(double f); // 進入
3 floor(double f); // 捨去
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
11   置
12 vector<int>::iterator it = lower_bound(v.
13   begin(), v.end(), val);
14 //用binary search找大於val的最小值的位置
15 vector<int>::iterator it = upper_bound(v.
16   begin(), v.end(), val);
17
18 /**找到範圍裏面的最大元素*/
19 max_element(n,n+len); // n到n+len範圍內最大
20   值
```

```
17 max_element(v.begin(),v.end()); // vector 中
18   最大值
19 /*找到範圍裏面的最大元素*/
20 min_element(n,n+len); // n到n+len範圍內最小
21   值
22 min_element(v.begin(),v.end()); // vector 中
23   最小值
24
25 /*queue*/
26 queue<datatype> q;
27 front(); /*取出最前面的值(沒有移除掉)*/
28 back(); /*取出最後面的值(沒有移除掉)*/
29 pop(); /*移掉最前面的值*/
30 push(); /*新增值到最後面*/
31 empty(); /*回傳bool,檢查是不是空的queue*/
32 size(); /*queue 的大小*/
33
34 /*stack*/
35 stack<datatype> s;
36 top(); /*取出最上面的值(沒有移除掉)*/
37 pop(); /*移掉最上面的值*/
38 push(); /*新增值到最上面*/
39 empty(); /*bool 檢查是不是空*/
40 size(); /*stack 的大小*/
41
42 /*unordered_set*/
43 unordered_set<datatype> s;
44 unordered_set<datatype> s(arr, arr + n);
45 /*initial with array*/
46 insert(); /*插入值*/
47 erase(); /*刪除值*/
48 empty(); /*bool 檢查是不是空*/
49 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 <<
2   endl;
```

# 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
5   有多少個連續的箱子
6 所以每次區間消去的值就是(k+1) * (k+1)
7 換言之, 我認為可以理解成 k 的意義就是題目今
8   天所關注的重點, 就是老師說的題目所規定的
9   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>
2   value, int bag_Weight)
3 {
4   // vector<int> weight = {1, 3, 4};
5   // vector<int> value = {15, 20, 30};
6   // int bagWeight = 4;
7   vector<vector<int>> dp(weight.size(),
8     vector<int>(bagWeight + 1, 0));
9   for (int j = weight[0]; j <= bagWeight;
10     j++)
11     dp[0][j] = value[0];
12   // weight數組的大小就是物品個數
13   for (int i = 1; i < weight.size(); i++)
14   { // 遍歷物品
15     for (int j = 0; j <= bagWeight; j++)
16     { // 遍歷背包容量
17       if (j < weight[i]) dp[i][j] = dp
18         [i - 1][j];
19       else dp[i][j] = max(dp[i - 1][j],
20         dp[i - 1][j - weight[i]]
21         + value[i]);
22     }
23   }
24   cout << dp[weight.size() - 1][bagWeight]
25     << endl;
26 }
```

## 2.4 Knapsack Unbounded

```

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

```

## 2.5 LCIS

```

1 int LCIS_len(vector<int> arr1, vector<int>
2   arr2)
3 {
4     int n = arr1.size(), m = arr2.size();
5     vector<int> table(m, 0);
6     for (int j = 0; j < m; j++)
7         table[j] = 0;
8     for (int i = 0; i < n; i++)
9     {
10        int current = 0;
11        for (int j = 0; j < m; j++)
12        {
13            if (arr1[i] == arr2[j])
14                if (current + 1 > table[j])
15                    table[j] = current + 1;
16
17            if (arr1[i] > arr2[j])
18                if (table[j] > current)
19                    current = table[j];
20        }
21    }
22    int result = 0;
23    for (int i = 0; i < m; i++)
24        if (table[i] > result)
25            result = table[i];
26    return result;
27 }

```

## 2.6 LCS

```

1 int LCS(vector<string> Ans, vector<string>
2   num)
3 {
4     int N = Ans.size(), M = num.size();
5     vector<vector<int>> LCS(N + 1, vector<
6       int>(M + 1, 0));
7     for (int i = 1; i <= N; ++i)
8     {
9         for (int j = 1; j <= M; ++j)
10        {
11            if (Ans[i - 1] == num[j - 1])
12                LCS[i][j] = LCS[i - 1][j - 1] + 1;

```

```

11        else
12            LCS[i][j] = max(LCS[i - 1][j]
13                , LCS[i][j - 1]);
14        }
15    }
16    cout << LCS[N][M] << '\n';
17    //列印 LCS
18    int n = N, m = M;
19    vector<string> k;
20    while (n && m)
21    {
22        if (LCS[n][m] != max(LCS[n - 1][m],
23            LCS[n][m - 1]))
24        {
25            k.push_back(Ans[n - 1]);
26            n--;
27            m--;
28        }
29        else if (LCS[n][m] == LCS[n - 1][m])
30            n--;
31        else if (LCS[n][m] == LCS[n][m - 1])
32            m--;
33    }
34    reverse(k.begin(), k.end());
35    for (auto i : k)
36        cout << i << " ";
37    cout << endl;
38    return LCS[N][M];
39 }

```

## 2.7 LIS

```

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

```

```

30         index = i;
31     }
32 }
33 cout << res << endl; // length
34 printLIS(arr, pos, index);
35 for (int i = 0; i < ans.size(); i++)
36 {
37     cout << ans[i];
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]) &&
19            (i - x >= 0) && (i + 1 + x < n)) //even length
20            x++;
21        if (2 * x > maxlen)
22        {
23            maxlen = 2 * x;
24            l = i - x + 1;
25            r = i + x;
26        }
27    }
28    cout << maxlen << '\n'; // 最後長度
29    cout << l + 1 << ' ' << r + 1 << '\n';
30    //頭到尾
31 }

```

## 2.9 Max\_subarray

```

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

```

```

6         global_max = max(local_max,
7             global_max);
8     }
9     return global_max;
10 }

```

## 2.10 Money problem

```

1 //能否湊得某個價位
2 void change(vector<int> price, int limit)
3 {
4     vector<bool> c(limit + 1, 0);
5     c[0] = true;
6     for (int i = 0; i < price.size(); ++i)
7         // 依序加入各種面額
8         for (int j = price[i]; j <= limit;
9             ++j) // 由低價位逐步到高價位
10            c[j] = c[j] | c[j - price[i]];
11        // 湊、湊、湊
12        if (c[limit]) cout << "YES\n";
13        else cout << "NO\n";
14    }
15 // 湊得某個價位的湊法總共幾種
16 void change(vector<int> price, int limit)
17 {
18     vector<int> c(limit + 1, 0);
19     c[0] = true;
20     for (int i = 0; i < price.size(); ++i)
21         for (int j = price[i]; j <= limit;
22             ++j)
23            c[j] += c[j - price[i]];
24    cout << c[limit] << '\n';
25 }
26 // 湊得某個價位的最少錢幣用量
27 void change(vector<int> price, int limit)
28 {
29     vector<int> c(limit + 1, 0);
30     c[0] = true;
31     for (int i = 0; i < price.size(); ++i)
32         for (int j = price[i]; j <= limit;
33             ++j)
34            c[j] = min(c[j], c[j - price[i]] + 1);
35    cout << c[limit] << '\n';
36 }
37 //湊得某個價位的錢幣用量，有幾種可能性
38 void change(vector<int> price, int limit)
39 {
40     vector<int> c(limit + 1, 0);
41     c[0] = true;
42     for (int i = 0; i < price.size(); ++i)
43         for (int j = price[i]; j <= limit;
44             ++j)
45            c[j] |= c[j - price[i]] << 1; //
46        // 錢幣數量加一，每一種可能性都
47        // 加一。
48    for (int i = 1; i <= 63; ++i)
49        if (c[i] & (1 << i))
50            cout << "用" << i << "個錢幣可湊
51                得價位" << i << " ";
52    }

```

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]
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    };

```

```

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].
56            size(); i++) {
57            Edge &e = edges[ G[u][i] ];
58            if ( d[u] + 1 != d[e.v] )
59                continue;
60            f = dfs(e.v, min(a, e.rest) );
61            if ( f > 0 ) {
62                e.rest -= f;
63                edges[ G[u][i]^1 ].rest += f;
64                flow += f;
65                a -= f;
66                if ( a == 0 ) break;
67            }
68        }
69        return flow;
70    }
71    long long maxflow(int _s, int _t){
72        s = _s, t = _t;
73        long long flow = 0, mf;
74        while ( bfs() ){
75            fill(cur,cur+n,0);
76            while ( (mf = dfs(s, INF)) )
77                flow += mf;
78        }
79        return flow;
80    }
81    } 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    }

```

```

25    for(int cur = t; cur != s; cur = pre[cur
26        ]){
27        residual[pre[cur]][cur] +=
28            bottleneck[t];
29        residual[cur][pre[cur]] -=
30            bottleneck[t];
31    }
32    ans += bottleneck[t];
33    }
34    return ans;
35    }
36    int main(){
37        int testcase = 1;
38        int n;
39        while(cin>>n){
40            if(n == 0)
41                break;
42            vector<vector<int>> capacity(n+1, vector
43                <int>(n+1, 0));
44            int s, t, c;
45            cin >> s >> t >> c;
46            int a, b, bandwidth;
47            for(int i = 0 ; i < c ; ++i){
48                cin >> a >> b >> bandwidth;
49                capacity[a][b] += bandwidth;
50                capacity[b][a] += bandwidth;
51            }
52            cout << "Network " << testcase++ << endl
53                ;
54            cout << "The bandwidth is " <<
55                getMaxFlow(capacity, s, t, n) << "."
56                << endl;
57            cout << endl;
58        }
59        return 0;
60    }

```

### 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(){

```

```

23    int res = 0;
24    memset(match,-1,sizeof(match));
25    for (int i=0; i<n; i++){
26        if (match[i]==-1){
27            memset(vis,0,sizeof(vis));
28            if ( dfs(i) ) res++;
29        }
30    }
31    return res;
32    }
33    } graph;

```

### 3.4 Maximum\_matching

```

1 /*bipartite - maximum matching*/
2 bool dfs(vector<vector<bool>> res,int node,
3     vector<int>& x, vector<int>& y, vector<
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            return vi[t] == tim;
56        }
57        ll dfs(int x, ll a)
58        {
59            if (x == t || a == 0)
60                return a;
61            ll flw = 0, f;

```

```

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

```

## 4 Geometry

### 4.1 Circle

```

1 bool same(double a, double b)
2 {
3     return abs(a - b) < 0;
4 }
5 struct P
6 {
7     double x, y;
8     P() : x(0), y(0) {}
9     P(double x, double y) : x(x), y(y) {}
10    P operator+(P b) { return P(x + b.x, y +
11        b.y); }
12    P operator-(P b) { return P(x - b.x, y -
13        b.y); }
14    P operator*(double b) { return P(x * b,
15        y * b); }
16    P operator/(double b) { return P(x / b,
17        y / b); }

```

```

14 double operator*(P b) { return x * b.x +
15     y * b.y; }
16 // double operator^(P b) { return x * b.
17     y - y * b.x; }
18 double abs() { return hypot(x, y); }
19 P unit() { return *this / abs(); }
20 P rot(double o)
21 {
22     double c = cos(o), s = sin(o);
23     return P(c * x - s * y, s * x + c *
24         y);
25 }
26 double angle() { return atan2(y, x); }
27
28 struct C
29 {
30     P c;
31     double r;
32     C(P c = P(0, 0), double r = 0) : c(c), r
33         (r) {}
34
35     vector<P> Intersect(C a, C b)
36     {
37         if (a.r > b.r)
38             swap(a, b);
39         double d = (a.c - b.c).abs();
40         vector<P> p;
41         if (same(a.r + b.r, d))
42             p.pb(a.c + (b.c - a.c).unit() * a.r)
43             ;
44         else if (a.r + b.r > d && d + a.r >= b.r
45             )
46         {
47             double o = acos((sqrt(a.r) + sqrt(d)
48                 - sqrt(b.r)) / (2 * a.r * d));
49             P i = (b.c - a.c).unit();
50             p.pb(a.c + i.rot(o) * a.r);
51             p.pb(a.c + i.rot(-o) * a.r);
52         }
53         return p;
54     }
55 }

```

### 4.2 Closest Pair

```

1 //最近點對 (距離) //台大
2 vector<pair<double, double>> p;
3 double closest_pair(int l, int r)
4 {
5     // p 要對 x 軸做 sort
6     if (l == r)
7         return 1e9;
8     if (r - l == 1)
9         return dist(p[l], p[r]); // 兩點距離
10    int m = (l + r) >> 1;
11    double d = min(closest_pair(l, m),
12        closest_pair(m + 1, r));
13    vector<int> vec;
14    for (int i = m; i >= l && fabs(p[m].x -
15        p[i].x) < d; --i)
16        vec.push_back(i);
17    for (int i = m + 1; i <= r && fabs(p[m].
18        x - p[i].x) < d; ++i)

```

```

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

```

### 4.3 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &x, const point<T> &
8         y) : p1(x), p2(y) {}
9     void pton()
10    { //轉成一般式
11        a = p1.y - p2.y;
12        b = p2.x - p1.x;
13        c = -a * p1.x - b * p1.y;
14    }
15    T ori(const point<T> &p) const
16    { //點和有向直線的關係 · >0左邊 · =0在線上
17        <0右邊
18        return (p2 - p1).cross(p - p1);
19    }
20    T btw(const point<T> &p) const
21    { //點投影落在線段上 <=0
22        return (p1 - p).dot(p2 - p);
23    }
24    bool point_on_segment(const point<T> &p)
25    const
26    { //點是否在線段上
27        return ori(p) == 0 && btw(p) <= 0;
28    }
29    T dis2(const point<T> &p, bool
30        is_segment = 0) const
31    { //點跟直線/線段的距離平方
32        point<T> v = p2 - p1, v1 = p - p1;
33        if (is_segment)
34        {
35            point<T> v2 = p - p2;
36            if (v.dot(v1) <= 0)
37                return v1.abs2();
38            if (v.dot(v2) >= 0)
39                return v2.abs2();
40        }
41        T tmp = v.cross(v1);
42        return tmp * tmp / v.abs2();
43    }
44    T seg_dis2(const line<T> &l) const
45    { //兩線段距離平方
46        return min({dis2(l.p1, 1), dis2(l.p2
47            , 1), l.dis2(p1, 1), l.dis2(p2,

```

```

    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 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);
}
};

```

#### 4.4 Point

```

const double PI = atan2(0.0, -1.0);
template <typename T>
struct point
{
    T x, y;
    point() {}
    point(const T &x, const T &y) : x(x), y(y) {}
    point operator+(const point &b) const
    {
        return point(x + b.x, y + b.y);
    }
    point operator-(const point &b) const
    {
        return point(x - b.x, y - b.y);
    }
    point operator*(const T &b) const
    {
        return point(x * b, y * b);
    }
    point operator/(const T &b) const
    {
        return point(x / b, y / b);
    }
    bool operator==(const point &b) const
    {
        return x == b.x && y == b.y;
    }
    T dot(const point &b) const
    {
        return x * b.x + y * b.y;
    }
}

```

```

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

```

#### 4.5 Polygon

```

template <typename T>
struct polygon
{
    polygon() {}
    vector<point<T>> p; //逆時針順序
    T area() const
    { //面積
        T ans = 0;
        for (int i = p.size() - 1, j = 0; j
            < (int)p.size(); i = j++)
            ans += p[i].cross(p[j]);
        return ans / 2;
    }
    point<T> center_of_mass() const
    { //重心
        T cx = 0, cy = 0, w = 0;
        for (int i = p.size() - 1, j = 0; j
            < (int)p.size(); i = j++)
        {
            T a = p[i].cross(p[j]);
            cx += (p[i].x + p[j].x) * a;
            cy += (p[i].y + p[j].y) * a;
            w += a;
        }
        return point<T>(cx / 3 / w, cy / 3 /
            w);
    }
    char ahas(const point<T> &t) const
    { //點是否在簡單多邊形內，是的話回傳1、
        //在邊上回傳-1、否則回傳0
        bool c = 0;

```

```

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

```



```

72 for (int n = p.size(), i = n - 1, j 118
    = 0; j < n; i = j++) 119
73 { 120
74     if (l.ori(p[i]) >= 0) 121
75     { 122
76         ans.p.push_back(p[i]); 123
77         if (l.ori(p[j]) < 0) 124
78             ans.p.push_back(l. 125
                line_intersection( 126
                    line<T>(p[i], p[j]))); 127
79     } 128
80     else if (l.ori(p[j]) > 0) 129
81         ans.p.push_back(l. 130
            line_intersection(line<T> 131
                >(p[i], p[j]))); 132
82 } 133
83 return ans; 134
84 } 135
85 static bool graham_cmp(const point<T> &a 136
    , const point<T> &b) 137
86 { //凸包排序函數 // 起始點不同 138
87     // return (a.x < b.x) || (a.x == b.x 139
        && a.y < b.y); //最左下角開始 140
88     return (a.y < b.y) || (a.y == b.y && 141
        a.x < b.x); //Y最小開始 142
89 } 143
90 void graham(vector<point<T>> &s) 144
91 { //凸包 Convexhull 2D 145
92     sort(s.begin(), s.end(), graham_cmp) 146
93     ; 147
94     p.resize(s.size() + 1); 148
95     int m = 0; 149
96     // cross >= 0 順時針。cross <= 0 逆 150
        時針旋轉 151
97     for (size_t i = 0; i < s.size(); ++i 152
        ) 153
98     { 154
99         while (m >= 2 && (p[m - 1] - p[m 155
            - 2]).cross(s[i] - p[m - 156
            2]) <= 0) 157
100             --m; 158
101             p[m++] = s[i]; 159
102     } 160
103     for (int i = s.size() - 2, t = m + 161
        1; i >= 0; --i) 162
104     { 163
105         while (m >= t && (p[m - 1] - p[m 164
            - 2]).cross(s[i] - p[m - 165
            2]) <= 0) 166
106             --m; 167
107             p[m++] = s[i]; 168
108     } 169
109     if (s.size() > 1) // 重複頭一次需扣 170
        掉 171
110     p.resize(m); 172
111 } 173
112 T diam() 174
113 { //直徑 175
114     int n = p.size(), t = 1; 176
115     T ans = 0; 177
116     p.push_back(p[0]); 178
117     for (int i = 0; i < n; i++) 179

```

```

{ 180
    point<T> now = p[i + 1] - p[i]; 181
    while (now.cross(p[t + 1] - p[i] 182
        ]) > now.cross(p[t] - p[i])) 183
        t = (t + 1) % n; 184
    ans = max(ans, (p[i] - p[t]). 185
        abs2()); 186
} 187
return p.pop_back(), ans; 188
} 189
T min_cover_rectangle() 190
{ //最小覆蓋矩形 191
    int n = p.size(), t = 1, r = 1, l; 192
    if (n < 3) 193
        return 0; //也可以做最小周長矩形 194
    T ans = 1e99; 195
    p.push_back(p[0]); 196
    for (int i = 0; i < n; i++) 197
    { 198
        point<T> now = p[i + 1] - p[i]; 199
        while (now.cross(p[t + 1] - p[i] 200
            ]) > now.cross(p[t] - p[i])) 201
            t = (t + 1) % n; 202
        while (now.dot(p[r + 1] - p[i] 203
            ) > now.dot(p[r] - p[i])) 204
            r = (r + 1) % n; 205
        if (!i) 206
            l = r; 207
        while (now.dot(p[l + 1] - p[i] 208
            ) <= now.dot(p[l] - p[i])) 209
            l = (l + 1) % n; 210
        T d = now.abs2(); 211
        T tmp = now.cross(p[t] - p[i]) * 212
            (now.dot(p[r] - p[i]) - now 213
            .dot(p[l] - p[i])) / d; 214
        ans = min(ans, tmp); 215
    } 216
    return p.pop_back(), ans; 217
} 218
T dis2(polygon &p1) 219
{ //凸包最近距離平方 220
    vector<point<T>> &P = p, &Q = p1.p; 221
    int n = P.size(), m = Q.size(), l = 222
        0, r = 0; 223
    for (int i = 0; i < n; ++i) 224
        if (P[i].y < P[l].y) 225
            l = i; 226
    for (int i = 0; i < m; ++i) 227
        if (Q[i].y < Q[r].y) 228
            r = i; 229
    P.push_back(P[0]), Q.push_back(Q[0]) 230
    ; 231
    T ans = 1e99; 232
    for (int i = 0; i < n; ++i) 233
    { 234
        while ((P[l] - P[l + 1]).cross(Q 235
            [r + 1] - Q[r]) < 0) 236
            r = (r + 1) % m; 237
        ans = min(ans, line<T>(P[l], P[l 238
            + 1]).seg_dis2(line<T>(Q[r] 239
            , Q[r + 1]))); 240
        l = (l + 1) % n; 241
    } 242
    return P.pop_back(), Q.pop_back(), 243
        ans; 244
} 245

```

```

} 246
static char sign(const point<T> &t) 247
{ 248
    return (t.y == 0 ? t.x : t.y) < 0; 249
} 250
static bool angle_cmp(const line<T> &A, 251
    const line<T> &B) 252
{ 253
    point<T> a = A.p2 - A.p1, b = B.p2 - 254
        B.p1; 255
    return sign(a) < sign(b) || (sign(a) 256
        == sign(b) && a.cross(b) > 0); 257
} 258
int halfplane_intersection(vector<line<T> 259
    >> &s) 260
{ //半平面交 261
    sort(s.begin(), s.end(), angle_cmp); 262
    //線段左側為該線段半平面 263
    int L, R, n = s.size(); 264
    vector<point<T>> px(n); 265
    vector<line<T>> q(n); 266
    q[L = R = 0] = s[0]; 267
    for (int i = 1; i < n; ++i) 268
    { 269
        while (L < R && s[i].ori(px[R - 270
            1]) <= 0) 271
            --R; 272
        while (L < R && s[i].ori(px[L]) 273
            <= 0) 274
            ++L; 275
        q[++R] = s[i]; 276
        if (q[R].parallel(q[R - 1])) 277
        { 278
            --R; 279
            if (q[R].ori(s[i].p1) > 0) 280
                q[R] = s[i]; 281
        } 282
        if (L < R) 283
            px[R - 1] = q[R - 1]. 284
                line_intersection(q[R]); 285
    } 286
    while (L < R && q[L].ori(px[R - 1]) 287
        <= 0) 288
        --R; 289
    p.clear(); 290
    if (R - L <= 1) 291
        return 0; 292
    px[R] = q[R].line_intersection(q[L]) 293
    ; 294
    for (int i = L; i <= R; ++i) 295
        p.push_back(px[i]); 296
    return R - L + 1; 297
} 298
} 299

```

```

triangle(const point<T> &a, const point< 300
    T> &b, const point<T> &c) : a(a), b(b), 301
    c(c) {} 302
T area() const 303
{ 304
    T t = (b - a).cross(c - a) / 2; 305
    return t > 0 ? t : -t; 306
} 307
point<T> barycenter() const 308
{ //重心 309
    return (a + b + c) / 3; 310
} 311
point<T> circumcenter() const 312
{ //外心 313
    static line<T> u, v; 314
    u.p1 = (a + b) / 2; 315
    u.p2 = point<T>(u.p1.x - a.y + b.y, 316
        u.p1.y + a.x - b.x); 317
    v.p1 = (a + c) / 2; 318
    v.p2 = point<T>(v.p1.x - a.y + c.y, 319
        v.p1.y + a.x - c.x); 320
    return u.line_intersection(v); 321
} 322
point<T> incenter() const 323
{ //內心 324
    T A = sqrt((b - c).abs2()), B = sqrt 325
        ((a - c).abs2()), C = sqrt((a - 326
        b).abs2()); 327
    return point<T>(A * a.x + B * b.x + 328
        C * c.x, A * a.y + B * b.y + C * 329
        c.y) / (A + B + C); 330
} 331
point<T> perpercenter() const 332
{ //垂心 333
    return barycenter() * 3 - 334
        circumcenter() * 2; 335
} 336
} 337

```

## 5 Graph

### 5.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/ 338
2 #define inf 99999 //define by you maximum 339
    edges weight 340
3 vector<vector<int>> > edges; 341
4 vector<int> dist; 342
5 vector<int> ancestor; 343
6 void BellmanFord(int start, int node){ 344
7     dist[start] = 0; 345
8     for(int it = 0; it < node-1; it++){ 346
9         for(int i = 0; i < node; i++){ 347
10             for(int j = 0; j < node; j++){ 348
11                 if(edges[i][j] != -1){ 349
12                     if(dist[i] + edges[i][j] 350
                        < dist[j]){ 351
13                         dist[j] = dist[i] + 352
                            edges[i][j]; 353
14                         ancestor[j] = i; 354

```

### 4.6 Triangle

```

1 template <typename T> 355
2 struct triangle 356
3 { 357
4     point<T> a, b, c; 358
5     triangle() {} 359

```

```

15     }
16     }
17 }
18 }
19 }
20
21 for(int i = 0; i < node; i++) //
22     negative cycle detection
23     for(int j = 0; j < node; j++)
24         if(dist[i] + edges[i][j] < dist[
25             j])
26         {
27             cout<<"Negative cycle!"<<
28                 endl;
29             return;
30         }
31 }
32 int main(){
33     int node;
34     cin>>node;
35     edges.resize(node,vector<int>(node,inf))
36     ;
37     dist.resize(node,inf);
38     ancestor.resize(node,-1);
39     int a,b,d;
40     while(cin>>a>>b>>d){
41         /*input: source destination weight*/
42         if(a == -1 && b == -1 && d == -1)
43             break;
44         edges[a][b] = d;
45     }
46     int start;
47     cin>>start;
48     BellmanFord(start,node);
49     return 0;
50 }

```

## 5.2 BFS-queue

```

1 /*BFS - queue version*/
2 void BFS(vector<int> &result, vector<pair<
3     int, int>> edges, int node, int start)
4 {
5     vector<int> pass(node, 0);
6     queue<int> q;
7     queue<int> p;
8     q.push(start);
9     int count = 1;
10    vector<pair<int, int>> newedges;
11    while (!q.empty())
12    {
13        pass[q.front()] = 1;
14        for (int i = 0; i < edges.size(); i
15            ++){
16            if (edges[i].first == q.front()
17                && pass[edges[i].second] ==
18                    0)
19            {
20                p.push(edges[i].second);
21                result[edges[i].second] =
22                    count;
23            }
24        }
25    }
26 }

```

```

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 }
44 int main()
45 {
46     int node;
47     cin >> node;
48     vector<pair<int, int>> edges;
49     int a, b;
50     while (cin >> a >> b)
51     {
52         /*a = b = -1 means input edges ended
53         */
54         if (a == -1 && b == -1)
55             break;
56         edges.push_back(pair<int, int>(a, b)
57             );
58     }
59     vector<int> result(node, -1);
60     BFS(result, edges, node, 0);
61     return 0;
62 }

```

## 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 }

```

## 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];
5 size_t la[K];
6 void dfs(int u, vector<int> &vec)
7 {
8     while (la[u] < G[u].size())
9     {
10         if (vis[G[u][la[u]].second])
11         {
12             ++la[u];
13             continue;
14         }
15         int v = G[u][la[u]].first;
16         vis[G[u][la[u]].second] = true;
17         ++la[u];
18         dfs(v, vec);
19         vec.push_back(v);
20     }
21 }

```

## 5.6 Floyd-warshall

```

1 /*SPA - Floyd-Warshall*/
2 // 有向圖 · 正邊 0(V³)
3 // 有向圖 · 無負環 0(V³)
4 // 有向圖 · 有負環 不適用
5
6 // 無向圖 · 正邊 0(V³)
7 // 無向圖 · 無負環 不適用
8 // 無向圖 · 有負環 不適用
9 /*Find min weight cycle*/
10 #define inf 99999
11 void floyd_warshall(vector<vector<int>> &
12     distance, vector<vector<int>> &ancestor,
13     int n)
14 {
15     for (int k = 0; k < n; k++)
16     {
17         for (int i = 0; i < n; i++)
18         {
19             for (int j = 0; j < n; j++)
20             {
21                 if (distance[i][k] +
22                     distance[k][j] <
23                         distance[i][j])
24                 {
25                     distance[i][j] =
26                         distance[i][k] +
27                             distance[k][j];
28                     ancestor[i][j] = k;
29                 }
30             }
31         }
32     }
33 }

```

```

21 distance[i][j] =
    distance[i][k] +
    distance[k][j];
22 ancestor[i][j] =
    ancestor[k][j];
23 }
24 }
25 }
26 }
27 }
28 }
29 vector<vector<int>> distance(n, vector<int>(
    n, inf));
30 vector<vector<int>> ancestor(n, vector<int>(
    n, -1));
31 distance[a][b] = w;
32 ancestor[a][b] = w;
33 floyd_warshall(distance, ancestor, n);
34 /*Negative cycle detection*/
35 for (int i = 0; i < n; i++)
36 {
37     if (distance[i][i] < 0)
38     {
39         cout << "Negative cycle!" << endl;
40         break;
41     }
42 }

```

```

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 */

```

## 5.7 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;

```

## 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],
            union_set);
14     return union_set[x];
15 }
16 void merge(int a, int b, vector<int>&
    union_set){
17     int pa = find(a, union_set);
18     int pb = find(b, union_set);
19     if(pa != pb)
20         union_set[pa] = pb;
21 }
22 void kruskal(priority_queue<edges> pq, int n)
23 {
24     vector<int> union_set(n, 0);
25     for (int i = 0; i < n; i++)
26         union_set[i] = i;
27     int edge = 0;
28     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 Minimum Weight Cycle

```

1 // 最小環
2 // 圖上無負環 !!!!
3 #define INF 99999
4 vector<vector<int>> w, d, p;
5 vector<int> cycle;
6 int c = 0;
7 void trace(int i, int j)
8 {
9     cycle[c++] = i;
10    if (i != j)
11        trace(p[i][j], j);
12 }
13 void init(int n)
14 {
15     for (int i = 0; i < n; ++i)
16         d[i][i] = 0;
17 }
18 void minimum_cycle(int n)
19 {
20     int weight = 1e9;
21     for (int k = 0; k < n; ++k)
22     {
23         for (int i = 0; i < k; ++i)
24             for (int j = 0; j < k; ++j)
25                 if (i != j)

```

```

26     if (w[k][i] + d[i][j] +
        w[j][k] < weight)
27     {
28         weight = w[k][i] + d[i][j] + w[j][k];
29         c = 0;
30         trace(i, j);
31         cycle[c++] = k;
32     }
33 }
34 for (int i = 0; i < n; ++i)
35 {
36     for (int j = 0; j < n; ++j)
37     {
38         if (d[i][k] + d[k][j] < d[i][j])
39         {
40             d[i][j] = d[i][k] + d[k][j];
41             p[i][j] = p[i][k];
42         }
43     }
44 }
45 if (weight == 1e9)
46     cout << "No exist";
47 else
48 {
49     bug(weight);
50     bug(c);
51     bugarr(cycle);
52 }
53 }
54 w.resize(n, vector<int>(n, INF));
55 d.resize(n, vector<int>(n, INF));
56 p.resize(n, vector<int>(n));
57 cycle.resize(n);
58 //Edge input
59 w[a][b] = w;
60 d[a][b] = w;
61 p[a][b] = b;
62 init(n);
63 minimum_cycle(n);

```

## 5.10 Prim

```

1 /*mst - Prim*/
2 #define inf 99999
3 struct edges
4 {
5     int from;
6     int to;
7     int weight;
8     friend bool operator < (edges a, edges b)
9     {
10         return a.weight > b.weight;
11     };
12 };
13 void Prim(vector<vector<int>> gp, int n, int
    start)
14 {
15     vector<bool> pass(n, false);

```



```

16 int edge = 0;
17 int cost = 0; //evaluate cost of mst
18 priority_queue<edges> pq;
19 for (int i = 0; i < n; i++)
20 {
21     if (gp[start][i] != inf)
22     {
23         edges tmp;
24         tmp.from = start;
25         tmp.to = i;
26         tmp.weight = gp[start][i];
27         pq.push(tmp);
28     }
29 }
30 pass[start] = true;
31 while (!pq.empty() && edge < n - 1)
32 {
33     edges cur = pq.top();
34     pq.pop();
35     if (!pass[cur.to])
36     {
37         for (int i = 0; i < n; i++)
38         {
39             if (gp[cur.to][i] != inf)
40             {
41                 edges tmp;
42                 tmp.from = cur.to;
43                 tmp.to = i;
44                 tmp.weight = gp[cur.to][i];
45                 pq.push(tmp);
46             }
47         }
48         pass[cur.to] = true;
49         edge += 1;
50         cost += cur.weight;
51     }
52 }
53 if (edge < n - 1)
54     cout << "No mst" << endl;
55 else
56     cout << cost << endl;
57 }
58 int main()
59 {
60     int n;
61     cin >> n;
62     int a, b, d;
63     vector<vector<int>> gp(n, vector<int>(n, inf));
64     while (cin >> a >> b >> d)
65     {
66         if (a == -1 && b == -1 && d == -1)
67             break;
68         if (gp[a][b] > d)
69             gp[a][b] = d;
70     }
71     Prim(gp, n, 0);
72     return 0;
73 }

```

## 5.11 Union\_find

```

1 // union_find from 台大
2 vector<int> father;
3 vector<int> people;
4 void init(int n)
5 {
6     for (int i = 0; i < n; i++)
7     {
8         father[i] = i;
9         people[i] = 1;
10    }
11 }
12 int Find(int x)
13 {
14     if (x != father[x])
15         father[x] = Find(father[x]);
16     return father[x];
17 }
18 void Union(int x, int y)
19 {
20     int m = Find(x);
21     int n = Find(y);
22     if (m != n)
23     {
24         father[n] = m;
25         people[m] += people[n];
26     }
27 }
28 }

```

## 6 Mathematics

### 6.1 Catalan

#### Catalan number

- 0~19項的catalan number
  - 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190
- 公式:  $C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

### 6.2 Combination

```

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

```

## 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)
6         return {1, 0};
7     long long k = a / b;
8     pair<long long, long long> p = extgcd(b,
9         a - k * b);
10    //cout << p.first << " " << p.second <<
11        endl;
12    //cout << "商數(k)= " << k << endl <<
13        endl;
14    return {p.second, p.first - k * p.second
15        };
16 }
17 int main()
18 {
19     int a, b;
20     cin >> a >> b;
21     pair<long long, long long> xy = extgcd(a,
22         b); //(x0,y0)
23     cout << xy.first << " " << xy.second <<
24         endl;
25     cout << xy.first << " * " << a << " + "
26         << xy.second << " * " << b << endl;
27     return 0;
28 }
29 // ax + by = gcd(a,b) * r
30 /*find |x|+|y| -> min*/
31 int main()
32 {
33     long long r, p, q; /*px+qy = r*/
34     int cases;
35     cin >> cases;
36     while (cases--)
37     {
38         cin >> r >> p >> q;
39         pair<long long, long long> xy =
40             extgcd(q, p); //(x0,y0)
41         long long ans = 0, tmp = 0;
42         double k, k1;
43         long long s, s1;
44         k = 1 - (double)(r * xy.first) / p;
45         s = round(k);
46         ans = llabs(r * xy.first + s * p) +
47             llabs(r * xy.second - s * q);
48         k1 = -(double)(r * xy.first) / p;
49         s1 = round(k1);
50         /*cout << k << endl << k1 << endl;
51             cout << s << endl << s1 << endl;
52             */
53         tmp = llabs(r * xy.first + s1 * p) +
54             llabs(r * xy.second - s1 * q);
55         ans = min(ans, tmp);
56         cout << ans << endl;
57     }
58     return 0;
59 }

```

## 6.4 Fermat

- $a^{(p-1)} \equiv 1 \pmod{p} \Leftrightarrow a * a^{(p-2)} \equiv 1/a$ 
  - $a^{(p-2)} \equiv 1/a$
- 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);
}
```

## 6.8 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： $a \equiv b \pmod{m} \Rightarrow c * m = a - b, c \in \mathbb{Z}$   
 $\Rightarrow a \equiv b \pmod{m} \Rightarrow m \mid a - b$

- 遞移性：若  $a \equiv b \pmod{c}, b \equiv d \pmod{c}$   
 則  $a \equiv d \pmod{c}$

- 保持基本運算：

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

- 放大縮小模數：

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

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

整數  $a$  在模  $c$  下要有模反元素的充分必要條件為  $a, c$  互質。

模逆元如果存在會有無限個，任意兩相鄰模逆元相差  $c$ 。

費馬小定理

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

歐拉定理

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

## 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 typedef long long ll;
2 ll modmul(ll a, ll b, ll mod)
3 {
4     ll ret = 0;
5     for (; b; b >>= 1, a = (a + a) % mod)
6         if (b & 1)
7             ret = (ret + a) % mod;
8     return ret;
9 }
10 ll qpowl(ll x, ll u, ll mod)
11 {
12     ll ret = 1ll;
13     for (; u; u >>= 1, x = modmul(x, x, mod))
14         if (u & 1)
15             ret = modmul(ret, x, mod);
16     return ret;
17 }
18 ll gcd(ll a, ll b)
19 {
20     return b ? gcd(b, a % b) : a;
21 }
22 ll Pollard_Rho(ll n, ll c)
23 {
24     ll i = 1, j = 2, x = rand() % (n - 1) + 1, y = x;
25     while (1)
26     {
27         i++;
28         x = (modmul(x, x, n) + c) % n;
29         ll p = gcd((y - x + n) % n, n);
30         if (p != 1 && p != n)
31             return p;
32         if (y == x)
33             return n;
34         if (i == j)
35         {
36             y = x;
37             j <<= 1;
38         }
39     }
40 }
```

```
39     }
40 }
41 bool Miller_Rabin(ll n)
42 {
43     ll x, pre, u = n - 1;
44     int i, j, k = 0;
45     if (n == 2 || n == 3 || n == 5 || n == 7 || n == 11)
46         return 1;
47     if (n == 1 || !(n % 2) || !(n % 3) || !(n % 5) || !(n % 7) || !(n % 11))
48         return 0;
49     while (!(u & 1))
50     {
51         k++;
52         u >>= 1;
53     }
54     srand((long long)12234336);
55     for (i = 1; i <= 50; i++)
56     {
57         x = rand() % (n - 2) + 2;
58         if (!(n % x))
59             return 0;
60         x = qpowl(x, u, n);
61         pre = x;
62         for (j = 1; j <= k; j++)
63         {
64             x = modmul(x, x, n);
65             if (x == 1 && pre != 1 && pre != n - 1)
66                 return 0;
67             pre = x;
68         }
69         if (x != 1)
70             return 0;
71     }
72     return 1;
73 }
74 // if (Miller_Rabin(n)) puts("Prime");
```

## 6.12 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));
```

## 6.13 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("%.31f\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,
   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 }

```

## 6.16 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>
   > 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 }

```

## 6.18 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 }

```

## 6.19 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 LL ans;
2 void find(LL n, LL c) // 配合質數判斷
3 {
4     if (n == 1)
5         return;
6     if (Miller_Rabin(n))
7     {
8         ans = min(ans, n);
9         // bug(ans); //質因數
10        return;
11    }
12    LL x = n, k = c;
13    while (x == n)
14        x = Pollard_Rho(x, c--);
15    find(n / x, k);
16    find(x, k);
17 }

```

## 6.21 質數數量

```

1 // 10 ^ 11 左右
2 #define LL long long
3 const int N = 5e6 + 2;
4 bool np[N];
5 int prime[N], pi[N];
6 int getprime()
7 {
8     int cnt = 0;
9     np[0] = np[1] = true;
10    pi[0] = pi[1] = 0;
11    for (int i = 2; i < N; ++i)
12    {
13        if (!np[i])
14            prime[++cnt] = i;
15        pi[i] = cnt;
16        for (int j = 1; j <= cnt && i *
   prime[j] < N; ++j)
17        {
18            np[i * prime[j]] = true;
19            if (i % prime[j] == 0)
20                break;
21        }
22    }
23    return cnt;
24 }
25 const int M = 7;
26 const int PM = 2 * 3 * 5 * 7 * 11 * 13 * 17;
27 int phi[PM + 1][M + 1], sz[M + 1];
28 void init()
29 {
30     getprime();
31     sz[0] = 1;
32     for (int i = 0; i <= PM; ++i)
33         phi[i][0] = i;
34     for (int i = 1; i <= M; ++i)
35     {
36         sz[i] = prime[i] * sz[i - 1];
37         for (int j = 1; j <= PM; ++j)
38             phi[j][i] = phi[j][i - 1] - phi[
   j / prime[i]][i - 1];
39     }
40 }

```

```

41 int sqrt2(LL x)
42 {
43     LL r = (LL)sqrt(x - 0.1);
44     while (r * r <= x)
45         ++r;
46     return int(r - 1);
47 }
48 int sqrt3(LL x)
49 {
50     LL r = (LL)cbrt(x - 0.1);
51     while (r * r * r <= x)
52         ++r;
53     return int(r - 1);
54 }
55 LL getphi(LL x, int s)
56 {
57     if (s == 0)
58         return x;
59     if (s <= M)
60         return phi[x % sz[s]][s] + (x / sz[s]
61             ) * phi[sz[s]][s];
62     if (x <= prime[s] * prime[s])
63         return pi[x] - s + 1;
64     if (x <= prime[s] * prime[s] * prime[s]
65         && x < N)
66     {
67         int s2x = pi[sqrt2(x)];
68         LL ans = pi[x] - (s2x + s - 2) * (
69             s2x - s + 1) / 2;
70         for (int i = s + 1; i <= s2x; ++i)
71             ans += pi[x / prime[i]];
72         return ans;
73     }
74     return getphi(x, s - 1) - getphi(x /
75         prime[s], s - 1);
76 }
77 LL getpi(LL x)
78 {
79     if (x < N)
80         return pi[x];
81     LL ans = getphi(x, pi[sqrt3(x)]) + pi[
82         sqrt3(x)] - 1;
83     for (int i = pi[sqrt3(x)] + 1, ed = pi[
84         sqrt2(x)]; i <= ed; ++i)
85         ans -= getpi(x / prime[i]) - i + 1;
86     return ans;
87 }
88 LL lehmer_pi(LL x)
89 {
90     if (x < N)
91         return pi[x];
92     int a = (int)lehmer_pi(sqrt2(sqrt2(x)));
93     int b = (int)lehmer_pi(sqrt2(x));
94     int c = (int)lehmer_pi(sqrt3(x));
95     LL sum = getphi(x, a) + (LL)(b + a - 2)
96         * (b - a + 1) / 2;
97     for (int i = a + 1; i <= b; i++)
98     {
99         LL w = x / prime[i];
100         sum -= lehmer_pi(w);
101         if (i > c)
102             continue;
103         LL lim = lehmer_pi(sqrt2(w));
104         for (int j = i; j <= lim; j++)
105             sum -= lehmer_pi(w / prime[j]) -
106                 (j - 1);
107     }
108 }

```

## 7 Other

### 7.1 binary search 三類變化

```

99     }
100     return sum;
101 }
102 // lehmer_pi(n)

// 查找和目標值完全相等的數
int find(vector<int> &nums, int target)
{
    int left = 0, right = nums.size();
    while (left < right)
    {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target)
            return mid;
        else if (nums[mid] < target)
            left = mid + 1;
        else
            right = mid;
    }
    return -1;
}

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

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

```

### 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
11        largest = root;
12    if (right <= length && array[right] >
13        array[largest])
14        largest = right;
15    if (largest != root)
16    {
17        swap(array[largest], array[root]);
18        MaxHeapify(array, largest, length);
19    }
20 }
21 void HeapSort(vector<int> &array)
22 {
23     array.insert(array.begin(), 0);
24     for (int i = (int)array.size() / 2; i >=
25         1; i--)
26         MaxHeapify(array, i, (int)array.size
27             () - 1);
28     int size = (int)array.size() - 1;
29     for (int i = (int)array.size() - 1; i >=
30         2; i--)
31     {
32         swap(array[1], array[i]);
33         size--;
34         MaxHeapify(array, 1, size);
35     }
36     array.erase(array.begin());
37 }

```

### 7.3 Josephus

```

1 /*n people kill k for each turn*/
2 int josephus(int n, int k)
3 {
4     int s = 0;
5     for (int i = 2; i <= n; i++)
6     {
7         s = (s + k) % i;
8     }
9     /*index start from 1 -> s+1*/
10    return s + 1;
11 }
12 /*died at kth*/
13 int kth(int n, int m, int k)
14 {
15     if (m == 1)
16         return n - 1;
17     for (k = k * m + m - 1; k >= n; k = k -
18         n + (k - n) / (m - 1))
19         ;
20     return k;
21 }

```

20 | }

### 7.4 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++)
13    {
14        if (LeftSub[idxLeft] <= RightSub[
15            idxRight])
16        {
17            arr[i] = LeftSub[idxLeft];
18            idxLeft++;
19        }
20        else
21        {
22            arr[i] = RightSub[idxRight];
23            idxRight++;
24        }
25    }
26 }
27 void MergeSort(vector<int> &arr, int front,
28     int end)
29 {
30     // front = 0 , end = arr.size() - 1
31     if (front < end)
32     {
33         int mid = (front + end) / 2;
34         MergeSort(arr, front, mid);
35         MergeSort(arr, mid + 1, end);
36         Merge(arr, front, mid, end);
37     }
38 }

```

### 7.5 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 }

```

```

13 i++;
14 swap(arr[i], arr[end]);
15 return i;
16 }
17 void QuickSort(vector<int> &arr, int front,
18 int end)
19 {
20 // front = 0, end = arr.size() - 1
21 if (front < end)
22 {
23 int pivot = Partition(arr, front,
24 end);
25 QuickSort(arr, front, pivot - 1);
26 QuickSort(arr, pivot + 1, end);
27 }
28 }

```

## 7.6 Weighted Job Scheduling

```

1 struct Job
2 {
3 int start, finish, profit;
4 };
5 bool jobComparataor(Job s1, Job s2)
6 {
7 return (s1.finish < s2.finish);
8 }
9 int latestNonConflict(Job arr[], int i)
10 {
11 for (int j = i - 1; j >= 0; j--)
12 {
13 if (arr[j].finish <= arr[i].start)
14 return j;
15 }
16 return -1;
17 }
18 int findMaxProfit(Job arr[], int n)
19 {
20 sort(arr, arr + n, jobComparataor);
21 int *table = new int[n];
22 table[0] = arr[0].profit;
23 for (int i = 1; i < n; i++)
24 {
25 int inclProf = arr[i].profit;
26 int l = latestNonConflict(arr, i);
27 if (l != -1)
28 inclProf += table[l];
29 table[i] = max(inclProf, table[i - 1]);
30 }
31 int result = table[n - 1];
32 delete[] table;
33 return result;
34 }

```

## 7.7 數獨解法

```

1 int getSquareIndex(int row, int column, int
2 n)

```

```

2 {
3 return row / n * n + column / n;
4 }
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);
20 for (int i = 1; i <= n2; i++)
21 {
22 if (!rows[rowNum][i] && !cols[colNum
23 ][i] && !boxes[getSquareIndex(
24 rowNum, colNum, n)][i])
25 {
26 rows[rowNum][i] = true;
27 cols[colNum][i] = true;
28 boxes[getSquareIndex(rowNum,
29 colNum, n)][i] = true;
30 board[rowNum][colNum] = i;
31 if (backtracking(board, rows,
32 cols, boxes, index + 1, n))
33 return true;
34 board[rowNum][colNum] = 0;
35 rows[rowNum][i] = false;
36 cols[colNum][i] = false;
37 boxes[getSquareIndex(rowNum,
38 colNum, n)][i] = false;
39 }
40 }
41 return false;
42 }
43 /*用法 main*/
44 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
45 vector<vector<int>> board(n * n + 1, vector<
46 int>(n * n + 1, 0));
47 vector<vector<bool>> isRow(n * n + 1, vector<
48 bool>(n * n + 1, false));
49 vector<vector<bool>> isColumn(n * n + 1,
50 vector<bool>(n * n + 1, false));
51 vector<vector<bool>> isSquare(n * n + 1,
52 vector<bool>(n * n + 1, false));
53 for (int i = 0; i < n * n; ++i)
54 {
55 for (int j = 0; j < n * n; ++j)
56 {
57 int number;
58 cin >> number;
59 board[i][j] = number;
60 if (number == 0)
61 continue;
62 isRow[i][number] = true;
63 isColumn[j][number] = true;
64 }
65 }

```

```

53 isSquare[getSquareIndex(i, j, n)][
54 number] = true;
55 }
56 if (backtracking(board, isRow, isColumn,
57 isSquare, 0, n))
58 /*有解答*/
59 else
60 /*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 q = next[q];
26 if (pattern[q] == text[i])
27 q++;
28 if (q == m)
29 {
30 cout << "Pattern occurs with
31 shift " << i - m + 1 << endl;
32 q = 0;
33 }
34 }
35 }
36 // string s = "abcdabcdebc";
37 // string p = "bcd";
38 // KMPMatcher(s, p);
39 // cout << endl;

```

### 8.2 Min Edit Distance

```

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

```

### 8.3 Sliding window

```

1 string minWindow(string s, string t)
2 {
3 unordered_map<char, int> letterCnt;
4 for (int i = 0; i < t.length(); i++)
5 letterCnt[t[i]]++;
6 int minLength = INT_MAX, minStart = -1;
7 int left = 0, matchCnt = 0;
8 for (int i = 0; i < s.length(); i++)
9 {
10 if (--letterCnt[s[i]] >= 0)
11 matchCnt++;
12 while (matchCnt == t.length())
13 {
14 if (i - left + 1 < minLength)
15 {
16 minLength = i - left + 1;
17 minStart = left;
18 }
19 if (++letterCnt[s[left]] > 0)
20 matchCnt--;
21 left++;
22 }
23 }
24 return minLength == INT_MAX ? "" : s.
25 substr(minStart, minLength);

```

### 8.4 Split

```

1 vector<string> mysplit(string s, string d)
2 {

```



```

3 int ps = 0, pe, dl = d.length();
4 string token;
5 vector<string> res;
6 while ((pe = s.find(d, ps)) != string::npos)
7 {
8     token = s.substr(ps, pe - ps);
9     ps = pe + dl;
10    res.push_back(token);
11 }
12 res.push_back(s.substr(ps));
13 return res;
14 }

```

## 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     static const int maxLEN
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    }
26    Bigint(string str)
27    {
28        s = 1;
29        vl = 0;
30        int stPos = 0, num = 0;
31        if (!str.empty() && str[0] == '-')
32        {
33            stPos = 1;
34            s = -1;
35        }
36        for (int i = str.length() - 1, q = 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    {
57        v[vl++] = x; //v.PB(x);
58    }
59    void pop_back()
60    {
61        vl--; //v.pop_back();
62    }
63    int back() const
64    {
65        return v[vl - 1]; //return v.back();
66    }
67    void n()
68    {
69        while (!empty() && !back())
70            pop_back();
71    }
72    void resize(int nl)
73    {
74        vl = nl;
75        fill(v, v + vl, 0); //fill(ALL(v), 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::ostream &out, const Bigint &a)
91    {
92        if (a.empty())
93        {
94            out << "0";
95            return out;
96        }
97        if (a.s == -1)
98            out << "-";
99        out << a.back();
100        for (int i = a.len() - 2; i >= 0; i--)
101            out << a.v[i];
102        char str[10];

```

```

103        snprintf(str, 5, "%.4d", a.v[i]);
104        out << str;
105    }
106    return out;
107 }
108 int cp3(const Bigint &b) const
109 {
110     if (s != b.s)
111         return s - b.s;
112     if (s == -1)
113         return -(*this).cp3(-b);
114     if (len() != b.len())
115         return len() - b.len(); //int
116     for (int i = len() - 1; i >= 0; i--)
117         if (v[i] != b.v[i])
118             return v[i] - b.v[i];
119     return 0;
120 }
121 bool operator<(const Bigint &b) const
122 {
123     return cp3(b) < 0;
124 }
125 bool operator<=(const Bigint &b) const
126 {
127     return cp3(b) <= 0;
128 }
129 bool operator==(const Bigint &b) const
130 {
131     return cp3(b) == 0;
132 }
133 bool operator!=(const Bigint &b) const
134 {
135     return cp3(b) != 0;
136 }
137 bool operator>(const Bigint &b) const
138 {
139     return cp3(b) > 0;
140 }
141 bool operator>=(const Bigint &b) const
142 {
143     return cp3(b) >= 0;
144 }
145 Bigint operator-(const Bigint &b) const
146 {
147     Bigint r = (*this);
148     r.s = -r.s;
149     return r;
150 }
151 Bigint operator+(const Bigint &b) const
152 {
153     if (s == -1)
154         return -(*this) + (-b);
155     if (b.s == -1)
156         return (*this) - (-b);
157     Bigint r;
158     int nl = max(len(), b.len());
159     r.resize(nl + 1);
160     for (int i = 0; i < nl; i++)
161     {
162         if (i < len())
163             r.v[i] += v[i];
164         if (i < b.len())
165             r.v[i] += b.v[i];
166         if (r.v[i] >= BIGMOD)

```

```

167         r.v[i + 1] += r.v[i] /
168         BIGMOD;
169         r.v[i] %= BIGMOD;
170     }
171     r.n();
172     return r;
173 }
174 Bigint operator-(const Bigint &b) const
175 {
176     if (s == -1)
177         return -(*this) - (-b);
178     if (b.s == -1)
179         return (*this) + (-b);
180     if ((*this) < b)
181         return -b - (*this);
182     Bigint r;
183     r.resize(len());
184     for (int i = 0; i < len(); i++)
185     {
186         r.v[i] += v[i];
187         if (i < b.len())
188             r.v[i] -= b.v[i];
189         if (r.v[i] < 0)
190         {
191             r.v[i] += BIGMOD;
192             r.v[i + 1]--;
193         }
194     }
195     r.n();
196     return r;
197 }
198 Bigint operator*(const Bigint &b)
199 {
200     Bigint r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j++)
206         {
207             r.v[i + j] += v[i] * b.v[j];
208             if (r.v[i + j] >= BIGMOD)
209             {
210                 r.v[i + j + 1] += r.v[i + j] / BIGMOD;
211                 r.v[i + j] %= BIGMOD;
212             }
213         }
214     }
215     r.n();
216     return r;
217 }
218 Bigint operator/(const Bigint &b)
219 {
220     Bigint r;
221     r.resize(max(1, len() - b.len() + 1));
222     int oriS = s;
223     Bigint b2 = b; // b2 = abs(b)
224     s = b2.s * r.s = 1;
225     for (int i = r.len() - 1; i >= 0; i--)
226     {
227         int d = 0, u = BIGMOD - 1;

```

```

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

```

## 9.2 DisjointSet

```

1 struct DisjointSet {
2     int p[maxn], sz[maxn], n, cc;
3     vector<pair<int*, int>> his;
4     vector<int> sh;
5     void init(int _n) {
6         n = _n; cc = n;
7         for (int i = 0; i < n; ++i) sz[i] =
8             1, p[i] = i;
9         sh.clear(); his.clear();
10    }
11    void assign(int *k, int v) {
12        his.emplace_back(k, *k);
13        *k = v;
14    }
15    void save() {
16        sh.push_back((int)his.size());
17    }
18    void undo() {
19        int last = sh.back(); sh.pop_back();
20        while (his.size() != last) {
21            int *k, v;
22            tie(k, v) = his.back(); his.
23                pop_back();
24            *k = v;
25        }
26    }
27    int find(int x) {
28        if (x == p[x]) return x;
29        return find(p[x]);
30    }
31    void merge(int x, int y) {
32        x = find(x); y = find(y);
33        if (x == y) return;
34        if (sz[x] > sz[y]) swap(x, y);
35        assign(&sz[y], sz[x] + sz[y]);
36        assign(&p[x], y);
37        assign(&cc, cc - 1);
38    }
39 };

```

## 9.3 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         for (int y = 0; y < r; y++)
56             for (int x = 0; x < c; ++x)
57                 m[y][x] = t.m[y][c + x] / t.
58                     m[y][y];
59         return true;
60     }
61     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 };

```

## 9.4 Trie

```

1 // biginter字典數
2 struct BigInteger{
3     static const int BASE = 100000000;
4     static const int WIDTH = 8;
5     vector<int> s;
6     BigInteger(long long num = 0){
7         *this = num;
8     }
9     BigInteger operator = (long long num){
10         s.clear();
11         do{
12             s.push_back(num % BASE);
13             num /= BASE;
14         }while(num > 0);
15         return *this;
16     }
17     BigInteger operator = (const string& str)
18     {
19         s.clear();
20         int x, len = (str.length() - 1) /
21             WIDTH + 1;
22         for(int i = 0; i < len; i++){

```

```

21         int end = str.length() - i*WIDTH
22             ;
23         int start = max(0, end-WIDTH);
24         sscanf(str.substr(start, end-
25             start).c_str(), "%d", &x);
26         s.push_back(x);
27     }
28     return *this;
29 }
30 BigInteger operator + (const BigInteger&
31     b) const{
32     BigInteger c;
33     c.s.clear();
34     for(int i = 0, g = 0; i < s.size(); i++){
35         if(g == 0 && i >= s.size() && i
36             >= b.s.size()) break;
37         int x = g;
38         if(i < s.size()) x+=s[i];
39         if(i < b.s.size()) x+=b.s[i];
40         c.s.push_back(x % BASE);
41         g = x / BASE;
42     }
43     return c;
44 }
45 ostream& operator << (ostream &out, const
46     BigInteger& x){
47     out << x.s.back();
48     for(int i = x.s.size()-2; i >= 0; i--){
49         char buf[20];
50         sprintf(buf, "%08d", x.s[i]);
51         out << buf;
52     }
53     return out;
54 }
55 istream& operator >> (istream &in,
56     BigInteger& x){
57     string s;
58     if(!(in >> s))
59         return in;
60     x = s;
61     return in;
62 }
63 struct Trie{
64     int c[500005][10];
65     int val[500005];
66     int sz;
67     int getIndex(char c){
68         return c - '0';
69     }
70     void init(){
71         memset(c[0], 0, sizeof(c[0]));
72         memset(val, -1, sizeof(val));
73         sz = 1;
74     }
75     void insert(BigInteger x, int v){
76         int u = 0;
77         int max_len_count = 0;
78         int firstNum = x.s.back();
79         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         int index = getIndex(s[i]);
115         if(!c[u][index]){
116             return -1;
117         }
118         u = c[u][index];
119     }
120     return val[u];
121 }
122 }
123 }

```

## 9.5 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)
17     const
18 {
19     return fraction(n * b.d + b.n * d, d * b
20         .d);
21 }
22 fraction operator-(const fraction &b)
23     const
24 {
25     return fraction(n * b.d - b.n * d, d * b
26         .d);
27 }
28 fraction operator*(const fraction &b)
29     const
30 {
31     return fraction(n * b.n, d * b.d);
32 }
33 fraction operator/(const fraction &b)
34     const
35 {
36     return fraction(n * b.d, d * b.n);
37 }
38 void print()
39 {
40     cout << n;
41     if (d != 1)
42         cout << "/" << d;
43 }
44 };

```

# TO DO WRITING NOT THINKING

## Contents

<b>1 Basic</b>	<b>1</b>	2.7 LIS . . . . .	2	5.6 Floyd-warshall . . . . .	7	6.18 數字加法組合 . . . . .	11
1.1 Basic codeblock setting . . .	1	2.8 LPS . . . . .	2	5.7 Hamilton_cycle . . . . .	8	6.19 羅馬數字 . . . . .	11
1.2 Basic vim setting . . . . .	1	2.9 Max_subarray . . . . .	2	5.8 Kruskal . . . . .	8	6.20 質因數分解 . . . . .	11
1.3 Code Template . . . . .	1	2.10 Money problem . . . . .	2	5.9 Minimum Weight Cycle . . .	8	6.21 質數數量 . . . . .	11
1.4 Python . . . . .	1			5.10 Prim . . . . .	8		
1.5 Range data . . . . .	1	<b>3 Flow &amp; matching</b>	<b>3</b>	5.11 Union_find . . . . .	9	<b>7 Other</b>	<b>12</b>
1.6 Some Function . . . . .	1	3.1 Dinic . . . . .	3			7.1 binary search 三類變化 . . . .	12
1.7 Time . . . . .	1	3.2 Edmonds_karp . . . . .	3	<b>6 Mathematics</b>	<b>9</b>	7.2 heap sort . . . . .	12
<b>2 DP</b>	<b>1</b>	3.3 hungarian . . . . .	3	6.1 Catalan . . . . .	9	7.3 Josephus . . . . .	12
2.1 3 維 DP 思路 . . . . .	1	3.4 Maximum_matching . . . . .	3	6.2 Combination . . . . .	9	7.4 Merge sort . . . . .	12
2.2 Knapsack Bounded . . . . .	1	3.5 MFlow Model . . . . .	4	6.3 Extended Euclidean . . . . .	9	7.5 Quick . . . . .	12
2.3 Knapsack sample . . . . .	1			6.4 Fermat . . . . .	9	7.6 Weighted Job Scheduling . . .	13
2.4 Knapsack Unbounded . . . . .	1	<b>4 Geometry</b>	<b>4</b>	6.5 Hex to Dec . . . . .	9	7.7 數獨解法 . . . . .	13
2.5 LCIS . . . . .	2	4.1 Circle . . . . .	4	6.6 Log . . . . .	10	<b>8 String</b>	<b>13</b>
2.6 LCS . . . . .	2	4.2 Closest Pair . . . . .	4	6.7 Mod . . . . .	10	8.1 KMP . . . . .	13
		4.3 Line . . . . .	4	6.8 Mod 性質 . . . . .	10	8.2 Min Edit Distance . . . . .	13
		4.4 Point . . . . .	5	6.9 PI . . . . .	10	8.3 Sliding window . . . . .	13
		4.5 Polygon . . . . .	5	6.10 Prime table . . . . .	10	8.4 Split . . . . .	13
		4.6 Triangle . . . . .	6	6.11 Prime 判斷 . . . . .	10		
		<b>5 Graph</b>	<b>6</b>	6.12 Round(小數) . . . . .	10	<b>9 data structure</b>	<b>14</b>
		5.1 Bellman-Ford . . . . .	6	6.13 二分逼近法 . . . . .	10	9.1 Bigint . . . . .	14
		5.2 BFS-queue . . . . .	7	6.14 公式 . . . . .	11	9.2 DisjointSet . . . . .	15
		5.3 DFS-rec . . . . .	7	6.15 四則運算 . . . . .	11	9.3 Matirx . . . . .	15
		5.4 Dijkstra . . . . .	7	6.16 因數表 . . . . .	11	9.4 Trie . . . . .	15
		5.5 Euler circuit . . . . .	7	6.17 數字乘法組合 . . . . .	11	9.5 分數 . . . . .	16