

# 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 set_union(all(a), all(b), back_inserter(d));
8 // 聯集
9
10 set_intersection(all(a), all(b),
11   back_inserter(c)); //交集
12
13
14 /** 全排列要先 sort !!! */
15 next_permutation(num.begin(), num.end());
16 prev_permutation(num.begin(), num.end());
17 //用binary search找大於或等於val的最小值的位
18   置
19 vector<int>::iterator it = lower_bound(v.
20   begin(), v.end(), val);
21 //用binary search找大於val的最小值的位置
22 vector<int>::iterator it = upper_bound(v.
23   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
20   範圍內最大值
21 max_element(v.begin(), v.end()); // vector
22   中最大值
23 /*找到範圍裏面的最大元素*/
24 min_element(n, n + len); // n到n+len
25   範圍內最小值
26 min_element(v.begin(), v.end()); // vector
27   中最小值
28
29 /*queue*/
30 queue<datatype> q;
31 front(); /*取出最前面的值(沒有移除掉)*/
32 back(); /*取出最後面的值(沒有移除掉)*/
33 pop(); /*移掉最前面的值*/
34 push(); /*新增值到最後面*/
35 empty(); /*回傳bool,檢查是不是空的queue*/
36 size(); /*queue 的大小*/
37
38 /*stack*/
39 stack<datatype> s;
40 top(); /*取出最上面的值(沒有移除掉)*/
41 pop(); /*移掉最上面的值*/
42 push(); /*新增值到最上面*/
43 empty(); /*bool 檢查是不是空*/
44 size(); /*stack 的大小*/
45
46 /*unordered_set*/
47 unordered_set<datatype> s;
48 unordered_set<datatype> s(arr, arr + n);
49 /*initial with array*/
50 insert(); /*插入值*/
51 erase(); /*刪除值*/
52 empty(); /*bool 檢查是不是空*/
53 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 在我的理解裡是視题目的要求而定的
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)
```

6 換言之，我認為可以理解成 k 的意義就是題目今  
天所關注的重點，就是老師說的題目所規定的  
運算

## 2.2 Knapsack Bounded

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

## 2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
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
11     cout << "最高的價值為" << c[w];

```

## 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)

```

```

9         if (Ans[i - 1] == num[j - 1])
10             LCS[i][j] = LCS[i - 1][j -
11               1] + 1;
12         else
13             LCS[i][j] = max(LCS[i - 1][j],
14               LCS[i][j - 1]);
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];

```

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

```

## 2.9 Max\_subarray

```

1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int> &nums) {
3     int local_max = nums[0], global_max =
4       nums[0];

```

```

4     for (int i = 1; i < nums.size(); i++) {
5         local_max = max(nums[i], nums[i] +
6           local_max);
7         global_max = max(local_max,
8           global_max);
9     }
10     return global_max;

```

## 2.10 Money problem

```

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

```

```

42     if (c[m] & (1 << i))
43         cout << "用" << i << "個錢幣可湊
44         得價位" << m;

```

## 3 Flow & matching

### 3.1 Dinic

```

1 const long long INF = 1LL<<60;
2 struct Dinic { //O(VVE), with minimum cut
3     static const int MAXN = 5003;
4     struct Edge{
5         int u, v;
6         long long cap, rest;
7     };
8     int n, m, s, t, d[MAXN], cur[MAXN];
9     vector<Edge> edges;
10    vector<int> G[MAXN];
11    void init(){
12        edges.clear();
13        for ( int i = 0 ; i < n ; i++ ) G[i]
14            .clear();
15        n = 0;
16        // min cut start
17        bool side[MAXN];
18        void cut(int u) {
19            side[u] = 1;
20            for ( int i : G[u] ) {
21                if ( !side[ edges[i].v ] &&
22                    edges[i].rest )
23                    cut(edges[i].v);
24            }
25            // min cut end
26            int add_node(){
27                return n++;
28            }
29            void add_edge(int u, int v, long long
30                cap){
31                edges.push_back( {u, v, cap, cap} );
32                edges.push_back( {v, u, 0, 0LL} );
33                m = edges.size();
34                G[u].push_back(m-2);
35                G[v].push_back(m-1);
36            }
37            bool bfs(){
38                fill(d,d+n,-1);
39                queue<int> que;
40                que.push(s); d[s]=0;
41                while (!que.empty()){
42                    int u = que.front(); que.pop();
43                    for (int ei : G[u]){
44                        Edge &e = edges[ei];
45                        if (d[e.v] < 0 && e.rest >
46                            0){
47                            d[e.v] = d[u] + 1;
48                            que.push(e.v);
49                        }
50                    }
51                }
52                return d[t] >= 0;
53            }
54            long long dfs(int u, long long a){
55                if ( u == t || a == 0 ) return a;
56                long long flow = 0, f;
57                for ( int &i=cur[u]; i < (int)G[u].
58                    size(); i++ ) {
59                    Edge &e = edges[ G[u][i] ];
60                    if ( d[u] + 1 != d[e.v] )
61                        continue;
62                    f = dfs(e.v, min(a, e.rest) );
63                    if ( f > 0 ) {
64                        e.rest -= f;
65                        edges[ G[u][i]^1 ].rest += f;
66                        flow += f;
67                        a -= f;
68                        if ( a == 0 ) break;
69                    }
70                }
71                return flow;
72            }
73            long long maxflow(int _s, int _t){
74                s = _s, t = _t;
75                long long flow = 0, mf;
76                while ( bfs() ){
77                    fill(cur,cur+n,0);
78                    while ( (mf = dfs(s, INF)) )
79                        flow += mf;
80                }
81                return flow;
82            }
83        } 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     }

```

```

22     }
23     }
24     if(bottleneck[t] == 0) break;
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     ans += bottleneck[t];
32     }
33     return ans;
34 }
35 int main(){
36     int testcase = 1;
37     int n;
38     while(cin>>n){
39         if(n == 0)
40             break;
41         vector<vector<int>> capacity(n+1, vector
42             <int>(n+1, 0));
43         int s, t, c;
44         cin >> s >> t >> c;
45         int a, b, bandwidth;
46         for(int i = 0 ; i < c ; ++i){
47             cin >> a >> b >> bandwidth;
48             capacity[a][b] += bandwidth;
49             capacity[b][a] += bandwidth;
50         }
51         cout << "Network " << testcase++ << endl
52             ;
53         cout << "The bandwidth is " <<
54             getMaxFlow(capacity, s, t, n) << "."
55             << endl;
56         cout << endl;
57     }
58     return 0;
59 }

```

### 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    }

```

```

20     return false;
21 }
22 int solve(){
23     int res = 0;
24     memset(match,-1,sizeof(match));
25     for (int i=0; i<n; i++){
26         if (match[i]==-1){
27             memset(vis,0,sizeof(vis));
28             if ( dfs(i) ) res++;
29         }
30     }
31     return res;
32 }
33 } graph;

```

### 3.4 Maximum\_matching

```

1 /*bipartite - maximum matching*/
2 bool dfs(vector<vector<bool>> res,int node,
3     vector<int>& x, vector<int>& y, vector<
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(n, -1);
31         for (int i = 0; i < n; i++){
32             vector<bool> pass(n, 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

```

```

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 = 1000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];
13
14    void Clear()
15    {
16        tot = 0;
17        tim = 0;
18        for (int i = 1; i <= n; ++i)
19            first[i] = -1;
20    }
21    void Add(int from, int to, ll cp, ll flw)
22    {
23        u[tot] = from;
24        v[tot] = to;
25        cap[tot] = cp;
26        flow[tot] = flw;
27        next[tot] = first[u[tot]];
28        first[u[tot]] = tot;
29        ++tot;
30    }
31    bool bfs()
32    {
33        ++tim;
34        dis[s] = 0;
35        vi[s] = tim;
36
37        int head, tail;
38        head = tail = 1;
39        que[head] = s;
40        while (head <= tail)
41        {
42            for (int i = first[que[head]]; i
43                != -1; i = next[i])
44            {
45                if (vi[v[i]] != tim && cap[i]
46                    > flow[i])
47                {
48                    vi[v[i]] = tim;
49                    dis[v[i]] = dis[que[head]] + 1;
50                    que[++tail] = v[i];
51                }
52            }
53            ++head;
54            return vi[t] == tim;
55        }

```

```

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

```

```

11    P operator-(P b) { return P(x - b.x, y -
12        b.y); }
13    P operator*(double b) { return P(x * b,
14        y * b); }
15    P operator/(double b) { return P(x / b,
16        y / b); }
17    double operator*(P b) { return x * b.x +
18        y * b.y; }
19    // double operator^(P b) { return x * b.
20        y - y * b.x; }
21    double abs() { return hypot(x, y); }
22    P unit() { return *this / abs(); }
23    P rot(double o)
24    {
25        double c = cos(o), s = sin(o);
26        return P(c * x - s * y, s * x + c *
27            y);
28    }
29    double angle() { return atan2(y, x); }
30 }
31 struct C
32 {
33     P c;
34     double r;
35     C(P c = P(0, 0), double r = 0) : c(c), r
36         (r) {}
37 }
38 vector<P> Intersect(C a, C b)
39 {
40     if (a.r > b.r)
41         swap(a, b);
42     double d = (a.c - b.c).abs();
43     vector<P> p;
44     if (same(a.r + b.r, d))
45         p.pb(a.c + (b.c - a.c).unit() * a.r)
46         ;
47     else if (a.r + b.r > d && d + a.r >= b.r
48         )
49     {
50         double o = acos((sqrt(a.r) + sqrt(d)
51             - sqrt(b.r)) / (2 * a.r * d));
52         P i = (b.c - a.c).unit();
53         p.pb(a.c + i.rot(o) * a.r);
54         p.pb(a.c + i.rot(-o) * a.r);
55     }
56     return p;
57 }

```

### 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));

```

```

vector<int> vec;
for (int i = m; i >= 1 && 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;
}

```

### 4.3 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);

```

```

38     return tmp * tmp / v.abs2();
39 }
40 T seg_dis2(const line<T> &l) const
41 { //兩線段距離平方
42     return min({dis2(l.p1, 1), dis2(l.p2
43         , 1), l.dis2(p1, 1), l.dis2(p2,
44             1)});
45 }
46 point<T> projection(const point<T> &p)
47     const
48 { //點對直線的投影
49     point<T> n = (p2 - p1).normal();
50     return p - n * (p - p1).dot(n) / n.
51         abs2();
52 }
53 point<T> mirror(const point<T> &p) const
54 {
55     //點對直線的鏡射，要先呼叫pton轉成一般式
56     point<T> R;
57     T d = a * a + b * b;
58     R.x = (b * b * p.x - a * a * p.x - 2
59         * a * b * p.y - 2 * a * c) / d;
60     R.y = (a * a * p.y - b * b * p.y - 2
61         * a * b * p.x - 2 * b * c) / d;
62     return R;
63 }
64 bool equal(const line &l) const
65 { //直線相等
66     return ori(l.p1) == 0 && ori(l.p2)
67         == 0;
68 }
69 bool parallel(const line &l) const
70 {
71     return (p1 - p2).cross(l.p1 - l.p2)
72         == 0;
73 }
74 bool cross_seg(const line &l) const
75 {
76     return (p2 - p1).cross(l.p1 - p1) *
77         (p2 - p1).cross(l.p2 - p1) <= 0;
78     //直線是否交線段
79 }
80 int line_intersect(const line &l) const
81 { //直線相交情況，-1無限多點、1交於一點、0不相交
82     return parallel(l) ? (ori(l.p1) == 0
83         ? -1 : 0) : 1;
84 }
85 int seg_intersect(const line &l) const
86 {
87     T c1 = ori(l.p1), c2 = ori(l.p2);
88     T c3 = l.ori(p1), c4 = l.ori(p2);
89     if (c1 == 0 && c2 == 0)
90     { //共線
91         bool b1 = btw(l.p1) >= 0, b2 =
92             btw(l.p2) >= 0;
93         T a3 = l.btw(p1), a4 = l.btw(p2)
94             ;
95         if (b1 && b2 && a3 == 0 && a4 >=
96             0)
97             return 2;
98         if (b1 && b2 && a3 >= 0 && a4 ==
99             0)
100             return 1;
101         return 0;
102     }
103     else if (c1 * c2 <= 0 && c3 * c4 <=
104         0)
105         return 1;
106     else if (c1 * c2 > 0 && c3 * c4 > 0)
107         return 0;
108     return 0;
109 }
110 }
111 };

```

#### 4.4 Point

```

1 const double PI = atan2(0.0, -1.0);
2 template <typename T>
3 struct point
4 {
5     T x, y;
6     point() {}
7     point(const T &x, const T &y) : x(x), y(y) {}
8     point operator+(const point &b) const
9     {
10         return point(x + b.x, y + b.y);
11     }
12     point operator-(const point &b) const
13     {
14         return point(x - b.x, y - b.y);
15     }
16     point operator*(const T &b) const
17     {
18         return point(x * b, y * b);
19     }
20     point operator/(const T &b) const
21     {
22         return point(x / b, y / b);
23     }
24     bool operator==(const point &b) const
25     {

```

```

26         return 3;
27         if (b1 && b2 && a3 >= 0 && a4 >=
28             0)
29             return 0;
30         return -1; //無限交點
31     }
32     else if (c1 * c2 <= 0 && c3 * c4 <=
33         0)
34         return 1;
35     else if (c1 * c2 > 0 && c3 * c4 > 0)
36         return 0; //不相交
37 }
38 point<T> line_intersection(const line &l
39     ) const
40 { //直線交點
41     point<T> a = p2 - p1, b = l.p2 - l.
42         p1, s = l.p1 - p1;
43     //if(a.cross(b)==0)return INF;
44     return p1 + a * (s.cross(b) / a.
45         cross(b));
46 }
47 point<T> seg_intersection(const line &l)
48     const
49 { //線段交點
50     int res = seg_intersect(l);
51     if (res <= 0)
52         assert(0);
53     if (res == 2)
54         return p1;
55     if (res == 3)
56         return p2;
57     return line_intersection(l);
58 }
59 };

```

#### 4.5 Polygon

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}
5     vector<point<T>> p; //逆時針順序
6     T area() const
7     { //面積
8         T ans = 0;
9         for (int i = p.size() - 1, j = 0; j
10             < (int)p.size(); i = j++)
11             ans += p[i].cross(p[j]);
12         return ans / 2;
13     }
14     point<T> center_of_mass() const
15     { //重心
16         T cx = 0, cy = 0, w = 0;
17         for (int i = p.size() - 1, j = 0; j
18             < (int)p.size(); i = j++)
19         {
20             T a = p[i].cross(p[j]);
21             cx += (p[i].x + p[j].x) * a;
22             cy += (p[i].y + p[j].y) * a;
23             w += a;
24         }
25         return point<T>(cx / 3 / w, cy / 3 /

```

```

26     }
27     char ahas(const point<T> &t) const
28     { //點是否在簡單多邊形內，是的話回傳1，
29         //在邊上回傳-1，否則回傳0
30         bool c = 0;
31         for (int i = 0, j = p.size() - 1; i
32             < p.size(); j = i++)
33             if (line<T>(p[i], p[j]).
34                 point_on_segment(t))
35                 return -1;
36             else if ((p[i].y > t.y) != (p[j]
37                 .y > t.y) &&
38                 t.x < (p[j].x - p[i].x) *
39                     (t.y - p[i].y) /
40                     (p[j].y - p[i].y) +
41                     p[i].x)
42                 c = !c;
43         return c;
44     }
45     char point_in_convex(const point<T> &x)
46         const
47     {
48         int l = 1, r = (int)p.size() - 2;
49         while (l <= r)
50         { //點是否在凸多邊形內，是的話回傳1，
51             //在邊上回傳-1，否則回傳0
52             int mid = (l + r) / 2;
53             T a1 = (p[mid] - p[0]).cross(x -
54                 p[0]);
55             T a2 = (p[mid + 1] - p[0]).cross
56                 (x - p[0]);
57             if (a1 >= 0 && a2 <= 0)
58             {
59                 T res = (p[mid + 1] - p[mid]
60                     ).cross(x - p[mid]);
61                 return res > 0 ? 1 : (res >=
62                     0 ? -1 : 0);
63             }
64             else if (a1 < 0)
65                 r = mid - 1;
66             else
67                 l = mid + 1;
68         }
69         return 0;
70     }
71     vector<T> getA() const
72     { //凸包邊對x軸的夾角
73         vector<T> res; //一定是遞增的
74         for (size_t i = 0; i < p.size(); ++i)
75             res.push_back((p[(i + 1) % p.
76                 size()] - p[i]).getA());
77         return res;
78     }
79     bool line_intersect(const vector<T> &A,
80         const line<T> &l) const
81     { //O(logN)
82         int f1 = upper_bound(A.begin(), A.
83             end(), (l.p1 - l.p2).getA()) -
84             A.begin();
85         int f2 = upper_bound(A.begin(), A.
86             end(), (l.p2 - l.p1).getA()) -
87             A.begin();
88         return l.cross_seg(line<T>(p[f1], p[
89             f2]));
90     }

```



```

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   ;
94   p.resize(s.size() + 1);
95   int m = 0;
96   // cross >= 0 順時針，cross <= 0 逆
    時針旋轉
97   for (size_t i = 0; i < s.size(); ++i
98   )
99   {
100     while (m >= 2 && (p[m - 1] - p[m
      - 2]).cross(s[i] - p[m - 1]) <= 0)
101       --m;
102     p[m++] = s[i];
103   }
104   for (int i = s.size() - 2, t = m +
    1; i >= 0; --i)
105   {
106     while (m >= t && (p[m - 1] - p[m
      - 2]).cross(s[i] - p[m - 1]) <= 0)
107       --m;
108     p[m++] = s[i];
109   }
110   if (s.size() > 1) // 重複點一次需扣
    掉
111     --m;
112   p.resize(m);
113   T diam()
114   { //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]
121       ) > now.cross(p[t] - p[i]))
122     {
123       t = (t + 1) % n;
124       ans = max(ans, (p[i] - p[t]).
        abs2());
125     }
126     return p.pop_back(), ans;
127   }
128   T min_cover_rectangle()
129   { //最小覆蓋矩形
130     int n = p.size(), t = 1, r = 1, l;
131     if (n < 3)
132       return 0; //也可以做最小周長矩形
133     T ans = 1e99;
134     p.push_back(p[0]);
135     for (int i = 0; i < n; i++)
136     {
137       point<T> now = p[i + 1] - p[i];
138       while (now.cross(p[t + 1] - p[i]
139         ) > now.cross(p[t] - p[i]))
140       {
141         t = (t + 1) % n;
142         while (now.dot(p[r + 1] - p[i]
143           ) > now.dot(p[r] - p[i]))
144         {
145           r = (r + 1) % n;
146           if (!i)
147             l = r;
148           while (now.dot(p[l + 1] - p[i]
149             ) <= now.dot(p[l] - p[i]))
150             l = (l + 1) % n;
151           T d = now.abs2();
152           T tmp = now.cross(p[t] - p[i]) *
            (now.dot(p[r] - p[i]) - now
              .dot(p[l] - p[i])) / d;
153           ans = min(ans, tmp);
154         }
155       }
156       return p.pop_back(), ans;
157     }
158     T dis2(polygon &pl)
159     { //凸包最近距離平方
160       vector<point<T>> &P = p, &Q = pl.p;
161       int n = P.size(), m = Q.size(), l =
        0, r = 0;
162       for (int i = 0; i < n; ++i)
163         if (P[i].y < P[l].y)
164           l = i;
165       for (int i = 0; i < m; ++i)
166         if (Q[i].y < Q[r].y)
167           r = i;
168       P.push_back(P[0]), Q.push_back(Q[0])
169       ;
170       T ans = 1e99;
171       for (int i = 0; i < n; ++i)
172       {
173         while ((P[l] - P[l + 1]).cross(Q
174           [r + 1] - Q[r]) < 0)
175           r = (r + 1) % m;
176         ans = min(ans, line<T>(P[l], P[l
177           + 1]).seg_dis2(line<T>(Q[r]
178             , Q[r + 1])));
179       }
180     }
181   }
182   }
183   }
184   }
185   }
186   }
187   }
188   }
189   }
190   }
191   }
192   }
193   }
194   }
195   }
196   }
197   }
198   }
199   }
200   }
201   }
202   }
203   }
204   }
205   }
206   }
207   }
208   }
209   }
210   }
211   }
212   }
213   }

```

```

1   l = (l + 1) % n;
2   }
3   return P.pop_back(), Q.pop_back(),
    ans;
4   }
5   static char sign(const point<T> &t)
6   {
7     return (t.y == 0 ? t.x : t.y) < 0;
8   }
9   static bool angle_cmp(const line<T> &A,
    const line<T> &B)
10  {
11    point<T> a = A.p2 - A.p1, b = B.p2 -
    B.p1;
12    return sign(a) < sign(b) || (sign(a)
    == sign(b) && a.cross(b) > 0);
13  }
14  int halfplane_intersection(vector<line<T>
    >> &s)
15  { //半平面交
16    sort(s.begin(), s.end(), angle_cmp);
17    //線段左側為該線段半平面
18    int L, R, n = s.size();
19    vector<point<T>> px(n);
20    vector<line<T>> q(n);
21    q[L = R = 0] = s[0];
22    for (int i = 1; i < n; ++i)
23    {
24      while (L < R && s[i].ori(px[R -
        1]) <= 0)
25        --R;
26      while (L < R && s[i].ori(px[L])
        <= 0)
27        ++L;
28      q[++R] = s[i];
29      if (q[R].parallel(q[R - 1]))
30      {
31        --R;
32        if (q[R].ori(s[i].p1) > 0)
33          q[R] = s[i];
34      }
35      if (L < R)
36        px[R - 1] = q[R - 1].
        line_intersection(q[R]);
37    }
38    while (L < R && q[L].ori(px[R - 1])
    <= 0)
39      --R;
40    p.clear();
41    if (R - L <= 1)
42      return 0;
43    px[R] = q[R].line_intersection(q[L])
44    ;
45    for (int i = L; i <= R; ++i)
46      p.push_back(px[i]);
47    return R - L + 1;
48  }
49  }
50  }
51  }
52  }
53  }
54  }
55  }
56  }
57  }
58  }
59  }
60  }
61  }
62  }
63  }
64  }
65  }
66  }
67  }
68  }
69  }
70  }
71  }
72  }
73  }
74  }
75  }
76  }
77  }
78  }
79  }
80  }
81  }
82  }
83  }
84  }
85  }
86  }
87  }
88  }
89  }
90  }
91  }
92  }
93  }
94  }
95  }
96  }
97  }
98  }
99  }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }

```

```

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

```

## 5 Graph

### 5.1 Bellman-Ford

```

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){

```

## 4.6 Triangle

1 | template <typename T>

```

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

```

## 5.2 BFS-queue

```

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

```

```

16        {
17            p.push(edges[i].second);
18            result[edges[i].second] = count;
19        }
20    }
21    else if (edges[i].second == q.front() && pass[edges[i].first] == 0)
22    {
23        p.push(edges[i].first);
24        result[edges[i].first] = count;
25    }
26    else
27        newedges.push_back(edges[i]);
28    }
29    edges = newedges;
30    newedges.clear();
31    q.pop();
32    if (q.empty() == true)
33    {
34        q = p;
35        queue<int> tmp;
36        p = tmp;
37        count++;
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

```

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 &&
10            (*iter).second == 0 && pass[(*

```

```

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

```

## 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, ancestor;
7 void dijkstra(int s)
8 {
9     priority_queue<pii, vector<pii>, greater<pii>> pq;
10    pq.push(pii(0, s));
11    ancestor[s] = -1;
12    while (!pq.empty())
13    {
14        int u = pq.top().second;
15        pq.pop();
16
17        isDone[u] = true;
18
19        for (auto &pr : weight[u])
20        {
21            int v = pr.first, w = pr.second;
22
23            if (!isDone[v] && dist[u] + w < dist[v])
24            {
25                dist[v] = dist[u] + w;
26                pq.push(pii(dist[v], v));

```

```

27                ancestor[v] = u;
28            }
29        }
30    }
31 }
32 // weight[a - 1].push_back(pii(b - 1, w));
33 // weight[b - 1].push_back(pii(a - 1, w));
34 // dist.resize(n, inf);
35 // ancestor.resize(n, -1);
36 // dist[0] = 0;
37 // dijkstra(0);

```

## 5.5 Euler circuit

```

1 /*Euler circuit*/
2 /*From NTU kiseki*/
3 /*G is graph, vis is visited, la is path*/
4 bool vis[N];
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>> &distance, vector<vector<int>> &ancestor, int n)
12 {
13     for (int k = 0; k < n; k++)
14     {
15         for (int i = 0; i < n; i++)
16         {
17             for (int j = 0; j < n; j++)
18             {

```

```

19     if (distance[i][k] +
20         distance[k][j] <
21         distance[i][j])
22     {
23         distance[i][j] =
24             distance[i][k] +
25             distance[k][j];
26         ancestor[i][j] =
27             ancestor[k][j];
28     }
29 }
30 vector<vector<int>> distance(n, vector<int>(
31     n, INF));
32 vector<vector<int>> ancestor(n, vector<int>(
33     n, -1));
34 distance[a][b] = w;
35 ancestor[a][b] = w;
36 floyd_warshall(distance, ancestor, n);
37 /*Negative cycle detection*/
38 for (int i = 0; i < n; i++)
39 {
40     if (distance[i][i] < 0)
41     {
42         cout << "Negative cycle!" << endl;
43         break;
44     }
45 }

```

```

26 while(cin>>n){
27     int a,b;
28     bool end = false;
29     vector<vector<int>> gp(n+1,vector<
30         int>(n+1,0));
31     while(cin>>a>>b){
32         if(a == 0 && b == 0)
33             break;
34         gp[a][b] = 1;
35         gp[b][a] = 1;
36     }
37     vector<int> solution(n + 1, -1);
38     vector<bool> pass(n + 1, false);
39     solution[1] = 0;
40     pass[1] = true;
41     bool flag = false;
42     hamilton(gp, 1,1 ,solution,pass,flag
43         );
44     if(!flag)
45         cout << "N" << endl;
46     return 0;
47 }
48 /*
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,
3     vector<int>& solution,vector<
4     bool> pass,bool& flag){
5     if(k == gp.size()-1){
6         if(gp[cur][1] == 1){
7             cout << 1 << " ";
8             while(cur != 1){
9                 cout << cur << " ";
10                cur = solution[cur];
11            }
12            cout << cur << endl;
13            flag = true;
14            return;
15        }
16    }
17    for (int i = 0; i < gp[cur].size() && !
18        flag; i++){
19        if(gp[cur][i] == 1 && !pass[i]){
20            pass[i] = true;
21            solution[i] = cur;
22            hamilton(gp, k + 1, i, solution,
23                pass,flag);
24            pass[i] = false;
25        }
26    }
27 }
28 int main(){
29     int n;

```

## 5.8 Kruskal

```

1 /*mst - Kruskal*/
2 struct edges{
3     int from;
4     int to;
5     int weight;
6     friend bool operator < (edges a, edges b
7     ){
8         return a.weight > b.weight;
9     }
10 };
11 int find(int x,vector<int>& union_set){
12     if(x != union_set[x])
13         union_set[x] = find(union_set[x],
14             union_set);
15     return union_set[x];
16 }
17 void merge(int a,int b,vector<int>&
18     union_set){
19     int pa = find(a, union_set);
20     int pb = find(b, union_set);
21     if(pa != pb)
22         union_set[pa] = pb;
23 }
24 void kruskal(priority_queue<edges> pq,int n)

```

```

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

```

## 5.9 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                     weight = min(mp[k][i] +
27                         d[i][j] + mp[j][k],
28                         weight);
29         for (int i = 0; i < n; ++i)
30             for (int j = 0; j < n; ++j)
31                 d[i][j] = min(d[i][k] + d[k]
32                     ][j], d[i][j]);
33         if (weight == INF)
34             cout << "Back to jail\n";
35         else
36             cout << weight << endl;
37     }
38     w.resize(n, vector<int>(n, INF));
39     d.resize(n, vector<int>(n, INF));
40     p.resize(n, vector<int>(n));
41     cycle.resize(n);
42     //Edge input
43     w[a][b] = w;

```



```

80 d[a][b] = w;
81 p[a][b] = b;
82 init(n);
83 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
    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            */
44         tmp = llabs(r * xy.first + s1 * p) +
            llabs(r * xy.second - s1 * q);
45         ans = min(ans, tmp);
46         cout << ans << endl;
47     }
48     return 0;
49 }

```

## 6.4 Fermat

- $a^{(p-1)} \equiv 1 \pmod{p} \Leftrightarrow a * a^{(p-2)} \equiv 1$ 
  - $a^{(p-2)} \equiv 1/a$
- 同餘因數定理
  - $a \equiv b \pmod{p} \Leftrightarrow k|a - b$
- 同餘加法性質
  - $a \equiv b \pmod{p}$  and  $c \equiv d \pmod{p}$   
 $\Leftrightarrow a + c \equiv b + d \pmod{p}$
- 同餘相乘性質
  - $a \equiv b \pmod{p}$  and  $c \equiv d \pmod{p}$   
 $\Leftrightarrow ac \equiv bd \pmod{p}$
- 同餘次方性質
  - $a \equiv b \pmod{p} \Leftrightarrow a^n \equiv b^n \pmod{p}$
- 同餘倍方性質
  - $a \equiv b \pmod{p} \Leftrightarrow am \equiv bm \pmod{p}$

## 6.6 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);
}
```

## 6.5 Hex to Dec

```
1 int HextoDec(string num) //16 to 10
2 {
3     int base = 1;
4     int temp = 0;
5     for (int i = num.length() - 1; i >= 0; i--)
6     {
7         if (num[i] >= '0' && num[i] <= '9')
8         {
9             temp += (num[i] - '0') * base;
10            base = base * 16;
11        }
12        else if (num[i] >= 'A' && num[i] <= 'F')
13        {
14            temp += (num[i] - 'A' + 10) * base;
15            base = base * 16;
16        }
17    }
18    return temp;
19 }
20 void DecToHex(int p) //10 to 16
21 {
22     char *l = new (char);
23     sprintf(l, "%X", p);
24     //int l_intResult = stoi(l);
25     cout << l << "\n";
26     //return l_intResult;
27 }
```

## 6.7 Mod

```
1 int pow_mod(int a, int n, int m) // a ^ n
2 {
3     mod m;
4     // a, n, m
5     < 10 ^ 9
6     if (n == 0)
7     {
8         return 1;
9     }
10    int x = pow_mid(a, n / 2, m);
11    long long ans = (long long)x * x % m;
12    if (n % 2 == 1)
13    {
14        ans = ans * a % m;
15    }
16    return (int)ans;
17 }
18 int inv(int a, int n, int p) // n = p-2
19 {
20     long long res = 1;
21     for (; n >= 1; (a *= a) %= p)
22     {
23         if (n & 1)
24             (res *= a) %= p;
25     }
26     return res;
27 }
```

## 6.8 Mod 性質

加法： $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$   
 減法： $(a - b) \bmod p = (a \bmod p - b \bmod p + p) \bmod p$   
 乘法： $(a * b) \bmod p = (a \bmod p * b \bmod p) \bmod p$   
 次方： $(a^b) \bmod p = ((a \bmod p)^b) \bmod p$   
 加法結合律： $((a + b) \bmod p + c) \bmod p = (a + (b + c)) \bmod p$   
 乘法結合律： $((a * b) \bmod p * c) \bmod p = (a * (b * c)) \bmod p$   
 加法交換律： $(a + b) \bmod p = (b + a) \bmod p$   
 乘法交換律： $(a * b) \bmod p = (b * a) \bmod p$   
 結合律： $((a + b) \bmod p * c) = ((a * c) \bmod p + (b * c) \bmod p) \bmod p$   
 如果  $a \equiv b \pmod{m}$ ，我們會說  $a, b$  在模  $m$  下同餘。  
 以下為性質：

- 整除性： $a \equiv b \pmod{m} \Rightarrow c * m = a - b, c \in \mathbb{Z}$   
 $\Rightarrow a \equiv b \pmod{m} \Rightarrow m | a - b$
- 遞移性：若  $a \equiv b \pmod{c}, b \equiv d \pmod{c}$   
 則  $a \equiv d \pmod{c}$
- 保持基本運算：

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

- 放大縮小模數：  
 $k \in \mathbb{Z}^+, a \equiv b \pmod{m} \Leftrightarrow k * a \equiv k * b \pmod{k * m}$   
 模逆元是取模下的反元素，即為找到  $a^{-1}$  使得  $aa^{-1} \equiv 1 \pmod{c}$ 。

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

模逆元如果存在會有無限個，任意兩相鄰模逆元相差  $c$ 。

費馬小定理

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

歐拉定理

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

Wilson's theorem

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

## 6.9 PI

```
1 #define PI acos(-1)
2 #define PI M_PI
```

## 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 >= 1; a = (a + a) % mod)
6         if (b & 1)
7             ret = (ret + a) % mod;
8     return ret;
9 }
10 ll qpow(ll x, ll u, ll mod)
11 {
12     ll ret = 1;
13     for (; 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
46         || n == 11)
47         return 1;
48     if (n == 1 || !(n % 2) || !(n % 3) || !(
49         n % 5) || !(n % 7) || !(n % 11))
50         return 0;
51     while (!(u & 1))
52     {
53         k++;
54         u >>= 1;
55     }
56     srand((long long)12234336);
57     for (i = 1; i <= 50; i++)
58     {
59         x = rand() % (n - 2) + 2;
60         if (!(n % x))
61             return 0;
62         x = qpow(x, u, n);
63         pre = x;
64         for (j = 1; j <= k; j++)
65         {
66             x = modmul(x, x, n);
67             if (x == 1 && pre != 1 && pre !=
68                 n - 1)
69                 return 0;
70             pre = x;
71         }
72         if (x != 1)
73             return 0;
74     }
75     // if (Miller_Rabin(n)) puts("Prime");

```

## 6.12 Round(小數)

```

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

```

## 6.13 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()

```

```

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

```

## 6.14 公式

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

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

## 6.15 四則運算

```

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

```

```

31 if (s[le] == '(' && s[ri] == ')')
32     return DFS(le + 1, ri - 1); //去除左
33     右兩邊空格
34 if (s[le] == '(')
35     return DFS(le + 1, ri); //去除左邊空
36     格
37 if (s[ri] == ')')
38     return DFS(le, ri - 1); //去除右邊空
39     格
40 long long int num = 0;
41 for (int i = le; i <= ri; i++)
42     num = num * 10 + s[i] - '0';
43 return num;

```

## 6.16 因數表

```

1 vector<vector<int>> arr(10000000);
2 const int limit = 10e7;
3 for (int i = 1; i <= limit; i++)
4 {
5     for (int j = i; j <= limit; j += i)
6         arr[j].pb(i); // i 為因數
7 }

```

## 6.17 數字乘法組合

```

1 void dfs(int j, int old, int num, vector<int>
2     > com, vector<vector<int>> &ans)
3 {
4     for (int i = j; i <= sqrt(num); i++)
5     {
6         if (old == num)
7             com.clear();
8         if (num % i == 0)
9         {
10            vector<int> a;
11            a = com;
12            a.push_back(i);
13            finds(i, old, num / i, a, ans);
14            a.push_back(num / i);
15            ans.push_back(a);
16        }
17    }
18 }
19 vector<vector<int>> ans;
20 vector<int> zero;
21 dfs(2, num, num, zero, ans);
22 /*num 為 input 數字*/
23 for (int i = 0; i < ans.size(); i++)
24 {
25     for (int j = 0; j < ans[i].size() - 1; j
26         ++
27         cout << ans[i][j] << " ";
28     cout << ans[i][ans[i].size() - 1] <<
29     endl;
30 }

```

## 6.18 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
2     &out, vector<vector<int>> &ans)
3 {
4     if (n == 0)
5     {
6         for (int i : out)
7             if (i > m)
8                 return;
9         ans.push_back(out);
10    }
11    for (int j = i; j <= n; j++)
12    {
13        out.push_back(j);
14        recur(j, n - j, m, out, ans);
15        out.pop_back();
16    }
17 }
18 vector<vector<int>> ans;
19 vector<int> zero;
20 recur(1, 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         ++
26         cout << ans[i][j] << " ";
27     cout << ans[i][ans[i].size() - 1] <<
28     endl;
29 }

```

## 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[
39                 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)

```

```

98        sum -= lehmer_pi(w / prime[j]) -
99            (j - 1);
100    }
101    return sum;
102 } // lehmer_pi(n)

```

## 7 Other

### 7.1 binary search 三類變化

```

1 // 查找和目標值完全相等的數
2 int find(vector<int> &nums, int target)
3 {
4     int left = 0, right = nums.size();
5     while (left < right)
6     {
7         int mid = left + (right - left) / 2;
8         if (nums[mid] == target)
9             return mid;
10        else if (nums[mid] < target)
11            left = mid + 1;
12        else
13            right = mid;
14    }
15    return -1;
16 }
17 // 找第一個不小於目標值的數 == 找最後一個小
18 // 於目標值的數
19 /*(lower_bound)*/
20 int find(vector<int> &nums, int target)
21 {
22     int left = 0, right = nums.size();
23     while (left < right)
24     {
25         int mid = left + (right - left) / 2;
26         if (nums[mid] < target)
27             left = mid + 1;
28        else
29            right = mid;
30    }
31    return right;
32 }
33 // 找第一個大於目標值的數 == 找最後一個不大
34 // 於目標值的數
35 /*(upper_bound)*/
36 int find(vector<int> &nums, int target)
37 {
38     int left = 0, right = nums.size();
39     while (left < right)
40     {
41         int mid = left + (right - left) / 2;
42         if (nums[mid] <= target)
43             left = mid + 1;
44        else
45            right = mid;
46    }
47    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;

```

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
13   for (int i = front; i <= end; i++)
14   {
15     if (LeftSub[idxLeft] <= RightSub[
16       idxRight])
17     {
18       arr[i] = LeftSub[idxLeft];
19       idxLeft++;
20     }
21     else
22     {
23       arr[i] = RightSub[idxRight];
24       idxRight++;
25     }
26   }
27 }
28 void MergeSort(vector<int> &arr, int front,
29   int end)
30 {
31   // front = 0, end = arr.size() - 1
32   if (front < end)
33   {
34     int mid = (front + end) / 2;
35     MergeSort(arr, front, mid);
36     MergeSort(arr, mid + 1, end);
37     Merge(arr, front, mid, end);
38   }
39 }

```

## 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  }

```

```

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 -
30       1]);
31   }
32   int result = table[n - 1];
33   delete[] table;
34   return result;
35 }

```

## 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>> &boxs
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
18   if (board[rowNum][colNum] != 0)
19     return backtracking(board, rows,
20       cols, boxs, index + 1, n);
21
22   for (int i = 1; i <= n2; i++)
23   {
24     if (!rows[rowNum][i] && !cols[colNum
25       ][i] && !boxs[getSquareIndex(
26         rowNum, colNum, n)][i])
27     {
28       rows[rowNum][i] = true;
29       cols[colNum][i] = true;
30       boxs[getSquareIndex(rowNum,
31         colNum, n)][i] = true;
32       board[rowNum][colNum] = i;
33       if (backtracking(board, rows,
34         cols, boxs, index + 1, n))
35         return true;
36       board[rowNum][colNum] = 0;
37       rows[rowNum][i] = false;
38       cols[colNum][i] = false;
39       boxs[getSquareIndex(rowNum,
40         colNum, n)][i] = false;
41     }
42   }
43   return false;
44 }
45
46 /*用法 main*/
47 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
48 vector<vector<int>> board(n * n + 1, vector<
49   int>(n * n + 1, 0));
50 vector<vector<bool>> isRow(n * n + 1, vector<
51   bool>(n * n + 1, false));
52 vector<vector<bool>> isColumn(n * n + 1,
53   vector<bool>(n * n + 1, false));
54 vector<vector<bool>> isSquare(n * n + 1,
55   vector<bool>(n * n + 1, false));
56
57 for (int i = 0; i < n * n; ++i)
58 {
59   for (int j = 0; j < n * n; ++j)
60   {
61     int number;
62     cin >> number;
63     board[i][j] = number;
64     if (number == 0)
65       continue;
66     isRow[i][number] = true;
67     isColumn[j][number] = true;
68   }
69 }

```

```

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   /*解答*/

```

## 8 String

### 8.1 KMP

```

1 // 用在在一個 S 內查找一個詞 W 的出現位置
2 void ComputePrefix(string s, int next[])
3 {
4   int n = s.length();
5   int q, k;
6   next[0] = 0;
7   for (k = 0, q = 1; q < n; q++)
8   {
9     while (k > 0 && s[k] != s[q])
10      k = next[k];
11     if (s[k] == s[q])
12      k++;
13     next[q] = k;
14   }
15 }
16 void KMPMatcher(string text, string pattern)
17 {
18   int n = text.length();
19   int m = pattern.length();
20   int next[pattern.length()];
21   ComputePrefix(pattern, next);
22
23   for (int i = 0, q = 0; i < n; i++)
24   {
25     while (q > 0 && pattern[q] != text[i
26       ])
27       q = next[q];
28     if (pattern[q] == text[i])
29       q++;
30     if (q == m)
31     {
32       cout << "Pattern occurs with
33         shift " << i - m + 1 << endl;
34       q = 0;
35     }
36   }
37 }
38 // string s = "abdcabcdebcd";
39 // string p = "bcd";
40 // KMPMatcher(s, p);
41 // 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]), dp[i - 1][j - 1]);
18         }
19     }
20     return dp[m][n];
}

```

## 8.3 Sliding window

```

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

```

## 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     // 正數位數
7     int s;
8     int v1, v[LEN];
9     // vector<int> v;
10    Bigint() : s(1) { v1 = 0; }
11    Bigint(long long a)
12    {
13        s = 1;
14        v1 = 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        v1 = 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                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 v1; //return SZ(v);
52    }
53    bool empty() const { return len() == 0; }
54    void push_back(int x)
55    {
56        v[v1++] = x; //v.PB(x);
57    }
58    void pop_back()
59    {
60        v1--; //v.pop_back();
61    }
62    int back() const
63    {
64        return v[v1 - 1]; //return v.back();
65    }
66    void n()
67    {
68        while (!empty() && !back())
69            pop_back();
70    }
71    void resize(int nl)
72    {
73        v1 = nl; //v.resize(nl);
74        fill(v, v + v1, 0); //fill(ALL(v), 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;
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             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             ++j)
207         {
208             r.v[i + j] += v[i] * b.v[j];
209             if (r.v[i + j] >= BIGMOD)
210             {
211                 r.v[i + j + 1] += r.v[i
212                     + j] / BIGMOD;
213                 r.v[i + j] %= BIGMOD;
214             }
215         }
216     }
217     r.n();
218     return r;
219 }
220 Bigint operator/(const Bigint &b)
221 {
222     Bigint r;
223     r.resize(max(1, len() - b.len() + 1)
224 );
225     int oriS = s;
226     Bigint b2 = b; // b2 = abs(b)
227     s = b2.s * r.s = 1;
228     for (int i = r.len() - 1; i >= 0; i
229         --)
230     {
231         int d = 0, u = BIGMOD - 1;

```

## 9.2 DisjointSet

```

232         while (d < u)
233         {
234             int m = (d + u + 1) >> 1;
235             r.v[i] = m;
236             if ((r * b2) > (*this))
237                 u = m - 1;
238             else
239                 d = m;
240         }
241         r.v[i] = d;
242     }
243     s = oriS;
244     r.s = s * b.s;
245     r.n();
246     return r;
247 }
248 };
249
250 struct DisjointSet {
251     int p[maxn], sz[maxn], n, cc;
252     vector<pair<int*, int>> his;
253     vector<int> sh;
254     void init(int _n) {
255         n = _n; cc = n;
256         for (int i = 0; i < n; ++i) sz[i] =
257             1, p[i] = i;
258         sh.clear(); his.clear();
259     }
260     void assign(int *k, int v) {
261         his.emplace_back(k, *k);
262         *k = v;
263     }
264     void save() {
265         sh.push_back((int)his.size());
266     }
267     void undo() {
268         int last = sh.back(); sh.pop_back();
269         while (his.size() != last) {
270             int *k, v;
271             tie(k, v) = his.back(); his.
272                 pop_back();
273             *k = v;
274         }
275     }
276     int find(int x) {
277         if (x == p[x]) return x;
278         return find(p[x]);
279     }
280     void merge(int x, int y) {
281         x = find(x); y = find(y);
282         if (x == y) return;
283         if (sz[x] > sz[y]) swap(x, y);
284         assign(&sz[y], sz[x] + sz[y]);
285         assign(&p[x], y);
286         assign(&cc, cc - 1);
287     }
288 };

```

## 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
20                     ][j];
21         return rev;
22     }
23     matrix operator-(const matrix &a)
24     {
25         matrix rev(r, c);
26         for (int i = 0; i < r; ++i)
27             for (int j = 0; j < c; ++j)
28                 rev[i][j] = m[i][j] - a.m[i
29                     ][j];
30         return rev;
31     }
32     matrix operator*(const matrix &a)
33     {
34         matrix rev(r, a.c);
35         matrix tmp(a.c, a.r);
36         for (int i = 0; i < a.r; ++i)
37             for (int j = 0; j < a.c; ++j)
38                 tmp[j][i] = a.m[i][j];
39         for (int i = 0; i < r; ++i)
40             for (int j = 0; j < a.c; ++j)
41                 for (int k = 0; k < c; ++k)
42                     rev.m[i][j] += m[i][k] *
43                         tmp[j][k];
44         return rev;
45     }
46     bool inverse() //逆矩陣判斷
47     {
48         Matrix t(r, r + c);
49         for (int y = 0; y < r; y++)
50         {
51             t.m[y][c + y] = 1;
52             for (int x = 0; x < c; ++x)
53                 t.m[y][x] = m[y][x];
54         }
55         if (!t.gas())
56             return false;
57         for (int y = 0; y < r; y++)
58             for (int x = 0; x < c; ++x)
59                 m[y][x] = t.m[y][c + x] / t.
60                     m[y][y];
61         return true;
62     }
63     T gas() //行列式

```

```

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
81                         ][i] - m[i][k] * mx;
82             }
83             T det = sign ? -1 : 1;
84             for (int i = 0; i < r; ++i)
85             {
86                 det = det * m[i][i];
87                 det = det / lazy[i];
88                 for (auto &j : m[i])
89                     j /= lazy[i];
90             }
91             return det;
92         }
93     };

```

## 9.4 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         for(int j = 0; j < strlen(buf); j++){
52             out << buf[j];
53         }
54     }
55     return out;
56 }
57 istream& operator >> (istream &in,
58 BigInteger& x){
59     string s;
60     if(!(in >> s))
61         return in;
62     x = s;
63     return in;
64 }
65 struct Trie{
66     int c[5000005][10];
67     int val[5000005];
68     int sz;
69     int getIndex(char c){
70         return c - '0';
71     }
72     void init(){
73         memset(c[0], 0, sizeof(c[0]));
74         memset(val, -1, sizeof(val));
75         sz = 1;
76     }
77     void insert(BigInteger x, int v){
78         int u = 0;
79         int max_len_count = 0;
80         int firstNum = x.s.back();
81         char firstBuf[20];
82         sprintf(firstBuf, "%d", firstNum);
83         for(int j = 0; j < strlen(firstBuf);
84             j++){
85             int index = getIndex(firstBuf[j]);
86             if(!c[u][index]){
87                 memset(c[sz], 0, sizeof(c[
88                 sz]));
89                 val[sz] = v;
90                 c[u][index] = sz++;
91             }
92             u = c[u][index];
93             max_len_count++;
94         }
95         for(int i = x.s.size()-2; i >= 0; i
96         --){
97             char buf[20];
98             sprintf(buf, "%08d", x.s[i]);
99             for(int j = 0; j < strlen(buf);
100                 j++){
101                 int index = getIndex(buf[j]);
102                 if(!c[u][index]){
103                     memset(c[sz], 0, sizeof
104                     (c[sz]));
105                     val[sz] = v;
106                     c[u][index] = sz++;
107                 }
108                 u = c[u][index];
109                 max_len_count++;
110             }
111             if(max_len_count >= 50){
112                 break;
113             }
114         }
115     }
116     int find(const char* s){
117         int u = 0;
118         int n = strlen(s);
119         for(int i = 0; i < n; i++){
120             int index = getIndex(s[i]);
121             if(!c[u][index]){
122                 return -1;
123             }
124             u = c[u][index];
125         }
126         return val[u];
127     }
128 }
129
130 fraction operator-(const fraction &b)
131 const
132 {
133     return fraction(-n, d);
134 }
135 fraction operator+(const fraction &b)
136 const
137 {
138     return fraction(n * b.d + b.n * d, d * b
139     .d);
140 }
141 fraction operator-(const fraction &b)
142 const
143 {
144     return fraction(n * b.d - b.n * d, d * b
145     .d);
146 }
147 fraction operator*(const fraction &b)
148 const
149 {
150     return fraction(n * b.n, d * b.d);
151 }
152 fraction operator/(const fraction &b)
153 const
154 {
155     return fraction(n * b.d, d * b.n);
156 }
157 void print()
158 {
159     cout << n;
160     if (d != 1)
161         cout << "/" << d;
162 }
163 };
164
165 typedef long long ll;
166 struct fraction
167 {
168     ll n, d;
169     fraction(const ll &n = 0, const ll &d =
170     1) : n(_n), d(_d)
171     {
172         ll t = __gcd(n, d);
173         n /= t, d /= t;
174         if (d < 0)

```

## 9.5 分數

# 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 . . . . .	12
1.3 Code Template . . . . .	1	2.10 Money problem . . . . .	2	5.9 Minimum Weight Cycle . . .	8	6.21 質數數量 . . . . .	12
1.4 Python . . . . .	1			5.10 Prim . . . . .	9		
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 . . . . .	13
2.2 Knapsack Bounded . . . . .	1	3.5 MFlow Model . . . . .	4	6.3 Extended Euclidean . . . . .	9	7.5 Quick . . . . .	13
2.3 Knapsack sample . . . . .	1			6.4 Fermat . . . . .	10	7.6 Weighted Job Scheduling . . .	13
2.4 Knapsack Unbounded . . . . .	2	<b>4 Geometry</b>	<b>4</b>	6.5 Hex to Dec . . . . .	10	7.7 數獨解法 . . . . .	13
2.5 LCIS . . . . .	2	4.1 Circle . . . . .	4	6.6 Log . . . . .	10		
2.6 LCS . . . . .	2	4.2 Closest Pair . . . . .	4	6.7 Mod . . . . .	10	<b>8 String</b>	<b>13</b>
		4.3 Line . . . . .	4	6.8 Mod 性質 . . . . .	10	8.1 KMP . . . . .	13
		4.4 Point . . . . .	5	6.9 PI . . . . .	10	8.2 Min Edit Distance . . . . .	14
		4.5 Polygon . . . . .	5	6.10 Prime table . . . . .	10	8.3 Sliding window . . . . .	14
		4.6 Triangle . . . . .	6	6.11 Prime 判斷 . . . . .	10	8.4 Split . . . . .	14
		<b>5 Graph</b>	<b>6</b>	6.12 Round(小數) . . . . .	11		
		5.1 Bellman-Ford . . . . .	6	6.13 二分逼近法 . . . . .	11	<b>9 data structure</b>	<b>14</b>
		5.2 BFS-queue . . . . .	7	6.14 公式 . . . . .	11	9.1 Bigint . . . . .	14
		5.3 DFS-rec . . . . .	7	6.15 四則運算 . . . . .	11	9.2 DisjointSet . . . . .	15
		5.4 Dijkstra . . . . .	7	6.16 因數表 . . . . .	11	9.3 Matirx . . . . .	15
		5.5 Euler circuit . . . . .	7	6.17 數字乘法組合 . . . . .	11	9.4 Trie . . . . .	15
						9.5 分數 . . . . .	16