

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 /** 全排列要先 sort !!! **/
14 next_permutation(num.begin(), num.end());
15 prev_permutation(num.begin(), num.end());
16 //用binary search找第一個大於或等於val的位置
17 vector<int>::iterator it = lower_bound(v.
18   begin(), v.end(), val);
19 //用binary search找第一個大於val的位置
20 vector<int>::iterator it = upper_bound(v.
21   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*/
54
55 /*tuple*/
56 tuple<datatype,datatype,datatype> t;
57 std::get<0>(t) /*Get first element of tuple
58   */
59 std::get<1>(t) /*Get second element of tuple
60   */
61 std::get<2>(t) /*Get third element of tuple
62   */
```

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

2.2 Knapsack Bounded

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

2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
2   value, int bag_Weight)
3 {
4   // vector<int> weight = {1, 3, 4};
5   // vector<int> value = {15, 20, 30};
6   // int bagWeight = 4;
7   vector<vector<int>> dp(weight.size(),
8     vector<int>(bagWeight + 1, 0));
9   for (int j = weight[0]; j <= bagWeight;
10     j++)
11     dp[0][j] = value[0];
12   // weight數組的大小就是物品個數
13   for (int i = 1; i < weight.size(); i++)
```

```

11 { // 遍歷物品
12   for (int j = 0; j <= bagWeight; j++)
13   { // 遍歷背包容量
14     if (j < weight[i]) dp[i][j] = dp[i - 1][j];
15     else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - weight[i]] + value[i]);
16   }
17 }
18 cout << dp[weight.size() - 1][bagWeight] << endl;
19 }

```

2.4 Knapsack Unbounded

```

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

```

2.5 LCIS

```

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

```

2.6 LCS

```

1 int LCS(vector<string> Ans, vector<string> num)
2 {
3   int N = Ans.size(), M = num.size();
4   vector<vector<int>> LCS(N + 1, vector<int>(M + 1, 0));
5   for (int i = 1; i <= N; ++i)
6   {
7     for (int j = 1; j <= M; ++j)
8     {
9       if (Ans[i - 1] == num[j - 1])
10        LCS[i][j] = LCS[i - 1][j - 1] + 1;
11      else
12        LCS[i][j] = max(LCS[i - 1][j], LCS[i][j - 1]);
13    }
14  }
15  cout << LCS[N][M] << '\n';
16  // 列印 LCS
17  int n = N, m = M;
18  vector<string> k;
19  while (n && m)
20  {
21    if (LCS[n][m] != max(LCS[n - 1][m], LCS[n][m - 1]))
22    {
23      k.push_back(Ans[n - 1]);
24      n--;
25      m--;
26    }
27    else if (LCS[n][m] == LCS[n - 1][m])
28      n--;
29    else if (LCS[n][m] == LCS[n][m - 1])
30      m--;
31  }
32  reverse(k.begin(), k.end());
33  for (auto i : k)
34    cout << i << " ";
35  cout << endl;
36  return LCS[N][M];
37 }

```

2.7 LIS O(Nlog(N))

```

1 int LIS(vector<int> &v) // O(n*log(n))
2 { // 需要求 LDS 請把 array reverse 反過來求 LIS
3   // 但必須注意 lower_bound or upper_bound
4   if (v.size() == 0)
5     return 0;
6   vector<int> dp(v.size(), 0);
7   int length = 1;
8   dp[0] = v[0];
9   for (int i = 1; i < v.size(); i++)
10   {
11     auto b = dp.begin(), e = dp.begin() + length;
12     // auto it = lower_bound(b, e, v[i])
13     ; // 後面 >= 前面

```

```

13   auto it = upper_bound(b, e, v[i]);
14   // 後面 > 前面
15   if (it == dp.begin() + length)
16     dp[length++] = v[i];
17   else
18     *it = v[i];
19 }
20 return length;
21 }

```

2.8 LIS

```

1 vector<int> ans;
2 void LIS(vector<int> &arr)
3 {
4   vector<int> dp(arr.size(), 1);
5   vector<int> pos(arr.size(), -1);
6   int res = INT_MIN, index = 0;
7   for (int i = 0; i < arr.size(); ++i)
8   {
9     for (int j = i + 1; j < arr.size(); ++j)
10    {
11      if (arr[j] > arr[i])
12      {
13        if (dp[i] + 1 > dp[j])
14        {
15          dp[j] = dp[i] + 1;
16          pos[j] = i;
17        }
18      }
19    }
20    if (dp[i] > res)
21    {
22      res = dp[i];
23      index = i;
24    }
25  }
26  cout << res << endl; // length
27  printLIS(arr, pos, index);
28  for (int i = 0; i < ans.size(); i++)
29  {
30    cout << ans[i];
31    if (i != ans.size() - 1)
32      cout << ' ';
33  }
34  cout << '\n';
35 }
36 void printLIS(vector<int> &arr, vector<int> &pos, int index)
37 {
38   if (pos[index] != -1)
39     printLIS(arr, pos, pos[index]);
40   ans.push_back(arr[index]);
41 }

```

2.9 LPS

```

1 void LPS(string s)
2 {
3   int maxlen = 0, l, r;
4   int n = s.size();
5   for (int i = 0; i < n; i++)
6   {
7     int x = 0;
8     while ((s[i - x] == s[i + x]) && (i - x >= 0) && (i + x < n)) // odd
9       x++;
10    x--;
11    if (2 * x + 1 > maxlen)
12    {
13      maxlen = 2 * x + 1;
14      l = i - x;
15      r = i + x;
16    }
17    x = 0;
18    while ((s[i - x] == s[i + 1 + x]) && (i - x >= 0) && (i + 1 + x < n)) // even length
19      x++;
20    if (2 * x > maxlen)
21    {
22      maxlen = 2 * x;
23      l = i - x;
24      r = i + x;
25    }
26  }
27  cout << maxlen << '\n'; // 最後長度
28  cout << l + 1 << ' ' << r + 1 << '\n'; // 頭到尾
29 }

```

2.10 Max_subarray

```

1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int> &nums) {
3   int local_max = nums[0], global_max = nums[0];
4   for (int i = 1; i < nums.size(); i++) {
5     local_max = max(nums[i], local_max + nums[i]);
6     global_max = max(local_max, global_max);
7   }
8   return global_max;
9 }

```

2.11 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

```

```

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

```

3 Flow & matching

3.1 Dinic

```

1  const long long INF = 1LL<<60;
2  struct Dinic { //O(VVE), with minimum cut
3      static const int MAXN = 5003;
4      struct Edge{
5          int u, v;
6          long long cap, rest;
7      };

```

```

8      int n, m, s, t, d[MAXN], cur[MAXN];
9      vector<Edge> edges;
10     vector<int> G[MAXN];
11     void init(){
12         edges.clear();
13         for (int i = 0; i < n; i++) G[i].clear();
14         n = 0;
15     }
16     // min cut start
17     bool side[MAXN];
18     void cut(int u) {
19         side[u] = 1;
20         for (int i : G[u]) {
21             if ( !side[ edges[i].v ] &&
22                 edges[i].rest )
23                 cut(edges[i].v);
24         }
25     }
26     // min cut end
27     int add_node(){
28         return n++;
29     }
30     void add_edge(int u, int v, long long 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 > 0){
46                     d[e.v] = d[u] + 1;
47                     que.push(e.v);
48                 }
49             }
50         }
51         return d[t] >= 0;
52     }
53     long long dfs(int u, long long a){
54         if ( u == t || a == 0 ) return a;
55         long long flow = 0, f;
56         for (int &i=cur[u]; i < (int)G[u].size(); i++) {
57             Edge &e = edges[ G[u][i] ];
58             if ( d[u] + 1 != d[e.v] ) continue;
59             f = dfs(e.v, min(a, e.rest) );
60             if ( f > 0 ) {
61                 e.rest -= f;
62                 edges[ G[u][i]^1 ].rest += f;
63                 a -= f;
64                 if ( a == 0 ) break;
65             }
66         }
67         return flow;
68     }

```

```

69     }
70     long long maxflow(int _s, int _t){
71         s = _s, t = _t;
72         long long flow = 0, mf;
73         while ( bfs() ){
74             fill(cur,cur+n,0);
75             while ( (mf = dfs(s, INF)) )
76                 flow += mf;
77         }
78     } dinic;

```

3.2 Edmonds_karp

```

1  /*Flow - Edmonds-karp*/
2  /*Based on UVa820*/
3  #define inf 1000000
4  int getMaxFlow(vector<vector<int>> &capacity, int s, int t, int n){
5      int ans = 0;
6      vector<vector<int>> residual(n+1, vector<int>(n+1, 0)); //residual network
7      while(true){
8          vector<int> bottleneck(n+1, 0);
9          bottleneck[s] = inf;
10         queue<int> q;
11         q.push(s);
12         vector<int> pre(n+1, 0);
13         while(!q.empty() && bottleneck[t] == 0){
14             int cur = q.front();
15             q.pop();
16             for(int i = 1; i <= n; i++){
17                 if(bottleneck[i] == 0 && capacity[cur][i] > residual[cur][i]){
18                     q.push(i);
19                     pre[i] = cur;
20                     bottleneck[i] = min(bottleneck[cur], capacity[cur][i] - residual[cur][i]);
21                 }
22             }
23             if(bottleneck[t] == 0) break;
24             for(int cur = t; cur != s; cur = pre[cur]){
25                 residual[pre[cur]][cur] += bottleneck[t];
26                 residual[cur][pre[cur]] -= bottleneck[t];
27             }
28             ans += bottleneck[t];
29         }
30     }
31     return ans;
32 }
33 int main(){
34     int testcase = 1;
35     int n;
36     while(cin>>n){
37         if(n == 0) break;
38         vector<vector<int>> capacity(n+1, vector<int>(n+1, 0));

```

```

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

```

3.3 hungarian

```

1  /*bipartite - hungarian*/
2  struct Graph{
3      static const int MAXN = 5003;
4      vector<int> G[MAXN];
5      int n, match[MAXN], vis[MAXN];
6      void init(int _n){
7          n = _n;
8          for (int i=0; i<n; i++) G[i].clear();
9      }
10     bool dfs(int u){
11         for (int v:G[u]){
12             if (vis[v]) continue;
13             vis[v]=true;
14             if (match[v]==-1 || dfs(match[v])){
15                 match[v] = u;
16                 match[u] = v;
17                 return true;
18             }
19         }
20         return false;
21     }
22     int solve(){
23         int res = 0;
24         memset(match,-1,sizeof(match));
25         for (int i=0; i<n; i++){
26             if (match[i]==-1){
27                 memset(vis,0,sizeof(vis));
28                 if ( dfs(i) ) res++;
29             }
30         }
31         return res;
32     }
33 } graph;

```

3.4 Maximum_matching

```

1  /*bipartite - maximum matching*/

```

```

2 bool dfs(vector<vector<bool>> res, int node,
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
47 3 1
48 answer is 3
49 */

```

3.5 MFlow Model

```

1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 1000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];

```

```

13 void Clear()
14 {
15     tot = 0;
16     tim = 0;
17     for (int i = 1; i <= n; ++i)
18         first[i] = -1;
19 }
20 void Add(int from, int to, ll cp, ll flw)
21 {
22     u[tot] = from;
23     v[tot] = to;
24     cap[tot] = cp;
25     flow[tot] = flw;
26     next[tot] = first[u[tot]];
27     first[u[tot]] = tot;
28     ++tot;
29 }
30 bool bfs()
31 {
32     ++tim;
33     dis[s] = 0;
34     vi[s] = tim;
35
36     int head, tail;
37     head = tail = 1;
38     que[head] = s;
39     while (head <= tail)
40     {
41         for (int i = first[que[head]]; i
42             != -1; i = next[i])
43         {
44             if (vi[v[i]] != tim && cap[i]
45                 > flow[i])
46             {
47                 vi[v[i]] = tim;
48                 dis[v[i]] = dis[que[head]]
49                     + 1;
50                 que[++tail] = v[i];
51             }
52             ++head;
53         }
54         return vi[t] == tim;
55     }
56 }
57 ll dfs(int x, ll a)
58 {
59     if (x == t || a == 0)
60         return a;
61     ll flw = 0, f;
62     int &i = cur[x];
63     for (i = first[x]; i != -1; i = next[i])
64     {
65         if (dis[x] + 1 == dis[v[i]] && (
66             f = dfs(v[i], min(a, cap[i]
67                             - flow[i]))) > 0)
68         {
69             flow[i] += f;
70             flow[i ^ 1] -= f;
71             a -= f;
72             flw += f;
73             if (a == 0)
74                 break;
75         }
76     }
77 }

```

```

78 }
79 return flw;
80 }
81 ll MaxFlow(int s, int t)
82 {
83     this->s = s;
84     this->t = t;
85     ll flw = 0;
86     while (bfs())
87     {
88         for (int i = 1; i <= n; ++i)
89             cur[i] = 0;
90         flw += dfs(s, oo);
91     }
92     return flw;
93 }
94 // MF Net;
95 // Net.n = n;
96 // Net.Clear();
97 // a 到 b (注意從1開始!!!!)
98 // Net.Add(a, b, w, 0);
99 // Net.MaxFlow(s, d)
100 // s 到 d 的 MF

```

```

25 struct C
26 {
27     P c;
28     double r;
29     C(P c = P(0, 0), double r = 0) : c(c), r
30         (r) {}
31 };
32 vector<P> Intersect(C a, C b)
33 {
34     if (a.r > b.r)
35         swap(a, b);
36     double d = (a.c - b.c).abs();
37     vector<P> p;
38     if (same(a.c + b.r, d))
39         p.pb(a.c + (b.c - a.c).unit() * a.r);
40     else if (a.r + b.r > d && d + a.r >= b.r)
41     {
42         double o = acos((sqrt(a.r) + sqrt(d)
43                         - sqrt(b.r)) / (2 * a.r * d));
44         P i = (b.c - a.c).unit();
45         p.pb(a.c + i.rot(o) * a.r);
46         p.pb(a.c + i.rot(-o) * a.r);
47     }
48     return p;
49 }

```

4 Geometry

4.1 Circle Intersect

```

1 bool same(double a, double b)
2 {
3     return abs(a - b) < 0;
4 }
5 struct P
6 {
7     double x, y;
8     P() : x(0), y(0) {}
9     P(double x, double y) : x(x), y(y) {}
10    P operator+(P b) { return P(x + b.x, y +
11        b.y); }
12    P operator-(P b) { return P(x - b.x, y -
13        b.y); }
14    P operator*(double b) { return P(x * b,
15        y * b); }
16    P operator/(double b) { return P(x / b,
17        y / b); }
18    double operator*(P b) { return x * b.x +
19        y * b.y; }
20    // double operator^(P b) { return x * b.
21        y - y * b.x; }
22    double abs() { return hypot(x, y); }
23    P unit() { return *this / abs(); }
24    P rot(double o)
25    {
26        double c = cos(o), s = sin(o);
27        return P(c * x - s * y, s * x + c *
28            y);
29    }
30    double angle() { return atan2(y, x); }
31 };

```

4.2 Closest Pair

```

1 //最近點對 (距離) //台大
2 vector<pair<double, double>> p;
3 double closest_pair(int l, int r)
4 {
5     // p 要對 x 軸做 sort
6     if (l == r)
7         return 1e9;
8     if (r - l == 1)
9         return dist(p[l], p[r]); // 兩點距離
10    int m = (l + r) >> 1;
11    double d = min(closest_pair(l, m),
12        closest_pair(m + 1, r));
13    vector<int> vec;
14    for (int i = m; i >= l && fabs(p[m].x -
15        p[i].x) < d; --i)
16        vec.push_back(i);
17    for (int i = m + 1; i <= r && fabs(p[m].
18        x - p[i].x) < d; ++i)
19        vec.push_back(i);
20    sort(vec.begin(), vec.end(), [&](int a,
21        int b)
22    { return p[a].y < p[b].y; });
23    for (int i = 0; i < vec.size(); ++i)
24        for (int j = i + 1; j < vec.size()
25            && fabs(p[vec[j]].y - p[vec[i]].
26                y) < d; ++j)
27            d = min(d, dist(p[vec[i]], p[vec
28                [j]]));
29    return d;
30 }

```

4.3 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &x, const point<T> &
8           y) : p1(x), p2(y) {}
9     void pton()
10    { //轉成一般式
11        a = p1.y - p2.y;
12        b = p2.x - p1.x;
13        c = -a * p1.x - b * p1.y;
14    }
15    T ori(const point<T> &p) const
16    { //點和有向直線的關係 · >0左邊、=0在線上
17        <0右邊
18        return (p2 - p1).cross(p - p1);
19    }
20    T btw(const point<T> &p) const
21    { //點投影落在線段上 <=0
22        return (p1 - p).dot(p2 - p);
23    }
24    bool point_on_segment(const point<T> &p)
25    const
26    { //點是否在線段上
27        return ori(p) == 0 && btw(p) <= 0;
28    }
29    T dis2(const point<T> &p, bool
30           is_segment = 0) const
31    { //點跟直線/線段的距離平方
32    point<T> v = p2 - p1, v1 = p - p1;
33    if (is_segment)
34    {
35        point<T> v2 = p - p2;
36        if (v.dot(v1) <= 0)
37            return v1.abs2();
38        if (v.dot(v2) >= 0)
39            return v2.abs2();
40    }
41    T tmp = v.cross(v1);
42    return tmp * tmp / v.abs2();
43    }
44    T seg_dis2(const line<T> &l) const
45    { //兩線段距離平方
46    return min({dis2(l.p1, 1), dis2(l.p2
47        , 1), l.dis2(p1, 1), l.dis2(p2,
48        1)});
49    }
50    point<T> projection(const point<T> &p)
51    const
52    { //點對直線的投影
53    point<T> n = (p2 - p1).normal();
54    return p - n * (p - p1).dot(n) / n.
55        abs2();
56    }
57    point<T> mirror(const point<T> &p) const
58    {
59        //點對直線的鏡射 · 要先呼叫pton轉成一
60        般式
61        point<T> R;

```

```

53    T d = a * a + b * b;
54    R.x = (b * b * p.x - a * a * p.x - 2
55          * a * b * p.y - 2 * a * c) / d;
56    R.y = (a * a * p.y - b * b * p.y - 2
57          * a * b * p.x - 2 * b * c) / d;
58    return R;
59    }
60    bool equal(const line &l) const
61    { //直線相等
62    return ori(l.p1) == 0 && ori(l.p2)
63        == 0;
64    }
65    bool parallel(const line &l) const
66    {
67    return (p1 - p2).cross(l.p1 - l.p2)
68        == 0;
69    }
70    bool cross_seg(const line &l) const
71    {
72    return (p2 - p1).cross(l.p1 - p1) *
73        (p2 - p1).cross(l.p2 - p1) <= 0;
74    //直線是否交線段
75    }
76    int line_intersect(const line &l) const
77    { //直線相交情況 · -1無限多點、1交於一
78    點、0不相交
79    return parallel(l) ? (ori(l.p1) == 0
80        ? -1 : 0) : 1;
81    }
82    int seg_intersect(const line &l) const
83    {
84    T c1 = ori(l.p1), c2 = ori(l.p2);
85    T c3 = l.ori(p1), c4 = l.ori(p2);
86    if (c1 == 0 && c2 == 0)
87    { //共線
88        bool b1 = btw(l.p1) >= 0, b2 =
89            btw(l.p2) >= 0;
90        T a3 = l.btw(p1), a4 = l.btw(p2)
91            ;
92        if (b1 && b2 && a3 == 0 && a4 >=
93            0)
94            return 2;
95        if (b1 && b2 && a3 >= 0 && a4 ==
96            0)
97            return 3;
98        if (b1 && b2 && a3 >= 0 && a4 >=
99            0)
100            return 0;
101        return -1; //無限交點
102    }
103    else if (c1 * c2 <= 0 && c3 * c4 <=
104        0)
105        return 1;
106    return 0; //不相交
107    }
108    point<T> line_intersection(const line &l
109    ) const
110    { /*直線交點*/
111    point<T> a = p2 - p1, b = l.p2 - l.
112        p1, s = l.p1 - p1;
113    //if(a.cross(b)==0)return INF;
114    return p1 + a * (s.cross(b) / a.
115        cross(b));
116    }

```

```

117    point<T> seg_intersection(const line &l)
118    const
119    { //線段交點
120    int res = seg_intersect(l);
121    if (res <= 0)
122        assert(0);
123    if (res == 2)
124        return p1;
125    if (res == 3)
126        return p2;
127    return line_intersection(l);
128    }
129    };

```

4.4 max_cover_rectangle

```

1 const double PI = atan2(0.0, -1.0);
2 const double eps = 1e-10;
3 typedef point<double> p; // data type 依照題
4 目更改
5 int mycmp(double a) { return fabs(a) < eps ?
6 0 : (a < 0 ? -1 : 1); }
7 double length(p a) { return sqrt(a.dot(a)); }
8 p Rotate(p a, double rad) { return p(a.x *
9 cos(rad) - a.y * sin(rad), a.x * sin(rad)
10 + a.y * cos(rad)); }
11 double angle(p a) { return atan2(a.y, a.x); }
12 double angle(p a, p b) { return atan2(a.
13 cross(b), a.dot(b)); }
14 double turnAngle(p a, p b) { return mycmp(a.
15 dot(b)) == 1 ? angle(a, b) : PI + angle(
16 a, b); }
17 double distanceOfpAndLine(p a, p b, p c) {
18 return fabs((b - a).cross(c - a) /
19 length(b - a)); }
20 double Area(int a, int b, int c, int d, p ab
21 , p cd, polygon<double> po)
22 {
23     double h1 = distanceOfpAndLine(po.p[a],
24 po.p[b], po.p[b] + ab);
25     double h2 = distanceOfpAndLine(po.p[c],
26 po.p[d], po.p[d] + cd);
27     return h1 * h2;
28 }
29 double max_cover_rectangle(polygon<double>
30 po)
31 {
32     po.pb(po.p[0]);
33     int m = po.p.size();
34     if (m < 3)
35         return 0; // 沒凸包哪來外包矩形
36     double Max = -1;
37     double Minx = po.p[0].x, Miny = po.p[0].
38 y, Maxx = po.p[0].x, Maxy = po.p[0].
39 y;
40     int p1 = 0, p2 = 0, p3 = 0, p4 = 0;
41     p v1, v2, ori;
42     ori = v1 = p(1, 0);
43     v2 = p(0, 1);
44     for (int i = 1; i < m; i++)

```

```

30    {
31        if (mycmp(Minx - po.p[i].x) == 1)
32            Minx = po.p[i].x, p3 = i;
33        if (mycmp(Maxx - po.p[i].x) == -1)
34            Maxx = po.p[i].x, p4 = i;
35        if (mycmp(Miny - po.p[i].y) == 1)
36            Miny = po.p[i].y, p1 = i;
37        if (mycmp(Maxy - po.p[i].y) == -1)
38            Maxy = po.p[i].y, p2 = i;
39    }
40    while (mycmp(ori.cross(v1)) >= 0)
41    {
42        double minRad = 1e20;
43        minRad = min(minRad, turnAngle(v1,
44            po.p[p1 + 1] - po.p[p1]));
45        minRad = min(minRad, turnAngle(v1 *
46            (-1), po.p[p2 + 1] - po.p[p2]));
47        minRad = min(minRad, turnAngle(v2 *
48            (-1), po.p[p3 + 1] - po.p[p3]));
49        minRad = min(minRad, turnAngle(v2,
50            po.p[p4 + 1] - po.p[p4]));
51        double l = 0, r = minRad;
52        while (mycmp(l - r))
53        {
54            double len = (r - l) / 3;
55            double midl = l + len;
56            double midr = r - len;
57            if (mycmp(Area(p1, p2, p3, p4,
58                Rotate(v1, midl), Rotate(v2,
59                midl), po) - Area(p1, p2,
60                p3, p4, Rotate(v1, midr),
61                Rotate(v2, midr), po)) == 1)
62                r = midr;
63            else
64                l = midl;
65        }
66        Max = max(Max, Area(p1, p2, p3, p4,
67            Rotate(v1, l), Rotate(v2, l), po
68            ));
69        v1 = Rotate(v1, minRad);
70        v2 = Rotate(v2, minRad);
71        if (mycmp(angle(v1, po.p[p1 + 1] -
72            po.p[p1])) == 0)
73            p1 = (p1 + 1) % m;
74        if (mycmp(angle(v1 * (-1), po.p[p2 +
75            1] - po.p[p2])) == 0)
76            p2 = (p2 + 1) % m;
77        if (mycmp(angle(v2 * (-1), po.p[p3 +
78            1] - po.p[p3])) == 0)
79            p3 = (p3 + 1) % m;
80        if (mycmp(angle(v2, po.p[p4 + 1] -
81            po.p[p4])) == 0)
82            p4 = (p4 + 1) % m;
83    }
84    return Max;
85    }

```

4.5 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 x == b.x && y == b.y;
27     }
28     T dot(const point &b) const
29     {
30         return x * b.x + y * b.y;
31     }
32     T cross(const point &b) const
33     {
34         return x * b.y - y * b.x;
35     }
36     point normal() const
37     { //求法向量
38         return point(-y, x);
39     }
40     T abs2() const
41     { //向量長度的平方
42         return dot(*this);
43     }
44     T rad(const point &b) const
45     { //兩向量的弧度
46         return fabs(atan2(fabs(cross(b)), dot(b)));
47     }
48     T getA() const
49     { //對x軸的弧度
50         T A = atan2(y, x); //超過180度會變負
51         if (A <= -PI / 2)
52             A += PI * 2;
53         return A;
54     }
55 };

56 vector<point<T>> p; //逆時針順序
57 T area() const
58 { //面積
59     T ans = 0;
60     for (int i = p.size() - 1, j = 0; j
61         < (int)p.size(); i = j++)
62         ans += p[i].cross(p[j]);
63     return ans / 2;
64 }
65 point<T> center_of_mass() const
66 { //重心
67     T cx = 0, cy = 0, w = 0;
68     for (int i = p.size() - 1, j = 0; j
69         < (int)p.size(); i = j++)
70     {
71         T a = p[i].cross(p[j]);
72         cx += (p[i].x + p[j].x) * a;
73         cy += (p[i].y + p[j].y) * a;
74         w += a;
75     }
76     return point<T>(cx / 3 / w, cy / 3 /
77         w);
78 }
79 char ahas(const point<T> &t) const
80 { //點是否在簡單多邊形內，是的話回傳1、
81     //在邊上回傳-1、否則回傳0
82     bool c = 0;
83     for (int i = 0, j = p.size() - 1; i
84         < p.size(); j = i++)
85         if (line<T>(p[i], p[j]).
86             point_on_segment(t))
87             return -1;
88         else if ((p[i].y > t.y) != (p[j]
89             .y > t.y) &&
90             t.x < (p[j].x - p[i].x) /
91                 * (t.y - p[i].y) /
92                 (p[j].y - p[i].y)
93                 + p[i].x)
94             c = !c;
95     return c;
96 }
97 char point_in_convex(const point<T> &x)
98 const
99 {
100     int l = 1, r = (int)p.size() - 2;
101     while (l <= r)
102     { //點是否在凸多邊形內，是的話回傳1
103         //在邊上回傳-1、否則回傳0
104         int mid = (l + r) / 2;
105         T a1 = (p[mid] - p[0]).cross(x -
106             p[0]);
107         T a2 = (p[mid + 1] - p[0]).cross
108             (x - p[0]);
109         if (a1 >= 0 && a2 <= 0)
110         {
111             T res = (p[mid + 1] - p[mid]
112                 ).cross(x - p[mid]);
113             return res > 0 ? 1 : (res >=
114                 0 ? -1 : 0);
115         }
116         else if (a1 < 0)
117             r = mid - 1;
118         else
119             l = mid + 1;
120     }
121 }

122 return 0;
123 }
124 vector<T> getA() const
125 { //凸包邊對x軸的夾角
126     vector<T> res; //一定是遞增的
127     for (size_t i = 0; i < p.size(); ++i
128         )
129         res.push_back((p[(i + 1) % p.
130             size()] - p[i]).getA());
131     return res;
132 }
133 bool line_intersect(const vector<T> &A,
134     const line<T> &l) const
135 { //O(logN)
136     int f1 = upper_bound(A.begin(), A.
137         end(), (l.p1 - l.p2).getA()) - A
138         .begin();
139     int f2 = upper_bound(A.begin(), A.
140         end(), (l.p2 - l.p1).getA()) - A
141         .begin();
142     return l.cross_seg(line<T>(p[f1], p[
143         f2]));
144 }
145 polygon cut(const line<T> &l) const
146 { //凸包對直線切割，得到直線l左側的凸包
147     polygon ans;
148     for (int n = p.size(), i = n - 1, j
149         = 0; j < n; i = j++)
150     {
151         if (l.ori(p[i]) >= 0)
152         {
153             ans.p.push_back(p[i]);
154             if (l.ori(p[j]) < 0)
155                 ans.p.push_back(l.
156                     line_intersection(
157                         line<T>(p[i], p[j])),
158                     );
159         }
160         else if (l.ori(p[j]) > 0)
161             ans.p.push_back(l.
162                 line_intersection(line<T>
163                     >(p[i], p[j])));
164     }
165     return ans;
166 }
167 static bool Andrew_Monotone_Chain_angle(
168     const point<T> &a, const point<T> &b
169     )
170 { //凸包排序函數 // 起始點不同
171     return (a.y < b.y) || (a.y == b.y &&
172         a.x < b.x); //Y最小開始
173 }
174 void Andrew_Monotone_Chain(vector<point<
175     T>> &s)
176 { //凸包 Convexhull 2D
177     sort(s.begin(), s.end(),
178         Andrew_Monotone_Chain_angle);
179     p.resize(s.size() + 1);
180     int m = 0;
181     // cross >= 0 順時針，cross <= 0 逆
182     // 時針旋轉
183     for (size_t i = 0; i < s.size(); ++i
184         )
185     {
186         while (m >= 2 && (p[m - 1] - p[m
187             - 2]).cross(s[i] - p[m -
188                 2]) <= 0)
189             --m;
190         p[m++] = s[i];
191     }
192     for (int i = s.size() - 2, t = m +
193         1; i >= 0; --i)
194     {
195         while (m >= t && (p[m - 1] - p[m
196             - 2]).cross(s[i] - p[m -
197                 2]) <= 0)
198             --m;
199         p[m++] = s[i];
200     }
201     if (s.size() > 1) // 重複頭一次需扣
202         掉
203         --m;
204     p.resize(m);
205     // p.pb(s[0]); // 需要頭在 pb 回去!!
206 }
207 T diam()
208 { //直徑
209     int n = p.size(), t = 1;
210     T ans = 0;
211     p.push_back(p[0]);
212     for (int i = 0; i < n; i++)
213     {
214         point<T> now = p[i + 1] - p[i];
215         while (now.cross(p[t + 1] - p[i]
216             ]) > now.cross(p[t] - p[i]))
217             t = (t + 1) % n;
218         ans = max(ans, (p[i] - p[t]).
219             abs2());
220     }
221     return p.pop_back(), ans;
222 }
223 T min_cover_rectangle()
224 { // 先做凸包 //最小覆蓋矩形
225     int n = p.size(), t = 1, r = 1, l;
226     if (n < 3)
227         return 0; //也可以做最小周長矩形
228     T ans = 1e99;
229     p.push_back(p[0]);
230     for (int i = 0; i < n; i++)
231     {
232         point<T> now = p[i + 1] - p[i];
233         while (now.cross(p[t + 1] - p[i]
234             ]) > now.cross(p[t] - p[i]))
235             t = (t + 1) % n;
236         while (now.dot(p[r + 1] - p[i])
237             > now.dot(p[r] - p[i]))
238             r = (r + 1) % n;
239         if (!l)
240             l = r;
241         while (now.dot(p[l + 1] - p[i])
242             <= now.dot(p[l] - p[i]))
243             l = (l + 1) % n;
244         T d = now.abs2();
245         T tmp = now.cross(p[t] - p[i]) *
246             (now.dot(p[r] - p[i]) - now
247                 .dot(p[l] - p[i])) / d;
248         ans = min(ans, tmp);
249     }
250     return p.pop_back(), ans;
251 }

```

4.6 Polygon

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}

```

```

149 }
150 T dis2(polygon &p1)
151 { //凸包最近距離平方
152   vector<point<T>> &P = p, &Q = p1.p;
153   int n = P.size(), m = Q.size(), l = 0, r = 0;
154   for (int i = 0; i < n; ++i)
155     if (P[i].y < P[l].y)
156       l = i;
157   for (int i = 0; i < m; ++i)
158     if (Q[i].y < Q[r].y)
159       r = i;
160   P.push_back(P[0]), Q.push_back(Q[0]);
161   T ans = 1e99;
162   for (int i = 0; i < n; ++i)
163   {
164     while ((P[l] - P[l + 1]).cross(Q[r + 1] - Q[r]) < 0)
165       r = (r + 1) % m;
166     ans = min(ans, line<T>(P[l], P[l + 1]).seg_dis2(line<T>(Q[r], Q[r + 1])));
167     l = (l + 1) % n;
168   }
169   return P.pop_back(), Q.pop_back(), ans;
170 }
171 static char sign(const point<T> &t)
172 {
173   return (t.y == 0 ? t.x : t.y) < 0;
174 }
175 static bool angle_cmp(const line<T> &A, const line<T> &B)
176 {
177   point<T> a = A.p2 - A.p1, b = B.p2 - B.p1;
178   return sign(a) < sign(b) || (sign(a) == sign(b) && a.cross(b) > 0);
179 }
180 int halfplane_intersection(vector<line<T>> &s)
181 { //半平面交
182   sort(s.begin(), s.end(), angle_cmp);
183   //線段左側為該線段半平面
184   int L, R, n = s.size();
185   vector<point<T>> px(n);
186   vector<line<T>> q(n);
187   q[L = R = 0] = s[0];
188   for (int i = 1; i < n; ++i)
189   {
190     while (L < R && s[i].ori(px[R - 1]) <= 0)
191       --R;
192     while (L < R && s[i].ori(px[L]) <= 0)
193       ++L;
194     q[++R] = s[i];
195     if (q[R].parallel(q[R - 1]))
196     {
197       --R;
198       if (q[R].ori(s[i].p1) > 0)
199         q[R] = s[i];
200     }
201     if (L < R)

```

```

201     px[R - 1] = q[R - 1].
202     line_intersection(q[R]);
203   }
204   while (L < R && q[L].ori(px[R - 1]) <= 0)
205     --R;
206   p.clear();
207   if (R - L <= 1)
208     return 0;
209   px[R] = q[R].line_intersection(q[L]);
210   for (int i = L; i <= R; ++i)
211     p.push_back(px[i]);
212   return R - L + 1;
213 }

```

4.7 Triangle

```

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

```

5 Graph

5.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/
2 #define inf 99999 //define by you maximum
3 edges weight
4 vector<vector<int>> edges;
5 vector<int> dist;
6 vector<int> ancestor;
7 void BellmanFord(int start, int node) {
8   dist[start] = 0;
9   for (int it = 0; it < node - 1; it++) {
10     for (int i = 0; i < node; i++) {
11       for (int j = 0; j < node; j++) {
12         if (edges[i][j] != -1) {
13           if (dist[i] + edges[i][j] < dist[j]) {
14             dist[j] = dist[i] + edges[i][j];
15             ancestor[j] = i;
16           }
17         }
18       }
19     }
20   }
21   for (int i = 0; i < node; i++) // negative cycle detection
22     for (int j = 0; j < node; j++)
23       if (dist[i] + edges[i][j] < dist[j]) {
24         cout << "Negative cycle!" << endl;
25         return;
26       }
27 }
28 int main() {
29   int node;
30   cin >> node;
31   edges.resize(node, vector<int>(node, inf));
32   dist.resize(node, inf);
33   ancestor.resize(node, -1);
34   int a, b, d;
35   while (cin >> a >> b >> d) {
36     /*input: source destination weight*/
37     if (a == -1 && b == -1 && d == -1)
38       break;
39     edges[a][b] = d;
40   }
41   int start;
42   cin >> start;
43   BellmanFord(start, node);
44   return 0;
45 }

```

5.2 BFS-queue

```

1 /*BFS - queue version*/
2 void BFS(vector<int> &result, vector<pair<int, int>> edges, int node, int start)
3 {
4   vector<int> pass(node, 0);
5   queue<int> q;
6   queue<int> p;
7   q.push(start);
8   int count = 1;
9   vector<pair<int, int>> newedges;
10  while (!q.empty())
11  {
12    pass[q.front()] = 1;
13    for (int i = 0; i < edges.size(); i++)
14    {
15      if (edges[i].first == q.front() && pass[edges[i].second] == 0)
16      {
17        p.push(edges[i].second);
18        result[edges[i].second] = count;
19      }
20      else if (edges[i].second == q.front() && pass[edges[i].first] == 0)
21      {
22        p.push(edges[i].first);
23        result[edges[i].first] = count;
24      }
25      else
26        newedges.push_back(edges[i]);
27    }
28    edges = newedges;
29    newedges.clear();
30    q.pop();
31    if (q.empty() == true)
32    {
33      q = p;
34      queue<int> tmp;
35      p = tmp;
36      count++;
37    }
38  }
39 }
40 int main()
41 {
42   int node;
43   cin >> node;
44   vector<pair<int, int>> edges;
45   int a, b;
46   while (cin >> a >> b)
47   {
48     /*a = b = -1 means input edges ended*/
49     if (a == -1 && b == -1)
50       break;
51     edges.push_back(pair<int, int>(a, b));
52   }
53   vector<int> result(node, -1);
54   BFS(result, edges, node, 0);
55 }

```

```

56     return 0;
57 }

```

5.3 DFS-rec

```

1  /*DFS - Recursive version*/
2  map<pair<int,int>,int> edges;
3  vector<int> pass;
4  vector<int> route;
5  void DFS(int start){
6      pass[start] = 1;
7      map<pair<int,int>,int>::iterator iter;
8      for(iter = edges.begin(); iter != edges.
9          end(); iter++){
10         if((*iter).first.first == start &&
11            (*iter).second == 0 && pass[(*)
12            iter).first.second] == 0){
13             route.push_back((*iter).first.
14                 second);
15             DFS((*iter).first.second);
16         }
17         else if((*iter).first.second ==
18             start && (*iter).second == 0 &&
19             pass[(*)iter).first.first] == 0){
20             route.push_back((*iter).first.
21                 first);
22             DFS((*iter).first.first);
23         }
24     }
25 }
26
27 int main(){
28     int node;
29     cin>>node;
30     pass.resize(node,0);
31     int a,b;
32     while(cin>>a>>b){
33         if(a == -1 && b == -1)
34             break;
35         edges.insert(pair<pair<int,int>,int>
36             >(pair<int,int>(a,b),0));
37     }
38     int start;
39     cin>>start;
40     route.push_back(start);
41     DFS(start);
42     return 0;
43 }

```

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(MAXN);
6  vector<int> isDone(MAXN, false), dist,
7      ancestor;
8  void dijkstra(int s)
9  {

```

```

9      priority_queue<pii, vector<pii>, greater
10         <pii>> pq;
11      pq.push(pii(0, s));
12      ancestor[s] = -1;
13      while (!pq.empty()){
14          int u = pq.top().second;
15          pq.pop();
16          isDone[u] = true;
17          for (auto &pr : weight[u])
18          {
19              int v = pr.first, w = pr.second;
20              if (!isDone[v] && dist[u] + w <
21                  dist[v])
22              {
23                  dist[v] = dist[u] + w;
24                  pq.push(pii(dist[v], v));
25                  ancestor[v] = u;
26              }
27          }
28      }
29      // weight[a - 1].push_back(pii(b - 1, w));
30      // weight[b - 1].push_back(pii(a - 1, w));
31      // dist.resize(n, inf);
32      // ancestor.resize(n, -1);
33      // dist[0] = 0;
34      // dijkstra(0);
35 }

```

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    O(V³)
3  // 有向圖 · 無負環    O(V³)
4  // 有向圖 · 有負環  不適用
5
6  // 無向圖 · 正邊    O(V³)
7  // 無向圖 · 無負環  不適用
8  // 無向圖 · 有負環  不適用
9  /*Find min weight cycle*/
10 #define inf 99999
11 void floyd_warshall(vector<vector<int>> &
12     distance, vector<vector<int>> &ancestor,
13     int n)
14 {
15     for (int k = 0; k < n; k++)
16     {
17         for (int i = 0; i < n; i++)
18         {
19             for (int j = 0; j < n; j++)
20             {
21                 if (distance[i][k] +
22                     distance[k][j] <
23                     distance[i][j])
24                 {
25                     distance[i][j] =
26                         distance[i][k] +
27                         distance[k][j];
28                     ancestor[i][j] =
29                         ancestor[k][j];
30                 }
31             }
32         }
33     }
34 }
35
36 vector<vector<int>> distance(n, vector<int>(
37     n, inf));
38 vector<vector<int>> ancestor(n, vector<int>(
39     n, -1));
40 distance[a][b] = w;
41 ancestor[a][b] = w;
42 floyd_warshall(distance, ancestor, n);
43
44 /*Negative cycle detection*/
45 for (int i = 0; i < n; i++)
46 {
47     if (distance[i][i] < 0)
48     {
49         cout << "Negative cycle!" << endl;
50         break;
51     }
52 }
53 }
54 }

```

5.7 Hamilton_cycle

```

1  /*find hamilton cycle*/
2  void hamilton(vector<vector<int>> gp, int k,
3      int cur, vector<int>& solution, vector<
4      bool> pass, bool& flag){
5      if(k == gp.size()-1){
6          if(gp[cur][1] == 1){
7              cout << 1 << " ";
8              while (cur != 1){
9                  cout << cur << " ";
10             }
11         }
12     }
13 }

```

```

8         cur = solution[cur];
9     }
10     cout << cur << endl;
11     flag = true;
12     return;
13 }
14 }
15 for (int i = 0; i < gp[cur].size() && !
16     flag; i++){
17     if(gp[cur][i] == 1 && !pass[i]){
18         pass[i] = true;
19         solution[i] = cur;
20         hamilton(gp, k + 1, i, solution,
21             pass, flag);
22         pass[i] = false;
23     }
24 }
25 int main(){
26     int n;
27     while(cin>>n){
28         int a,b;
29         bool end = false;
30         vector<vector<int>> gp(n+1,vector<
31             int>(n+1,0));
32         while(cin>>a>>b){
33             if(a == 0 && b == 0)
34                 break;
35             gp[a][b] = 1;
36             gp[b][a] = 1;
37         }
38         vector<int> solution(n + 1, -1);
39         vector<bool> pass(n + 1, false);
40         solution[1] = 0;
41         pass[1] = true;
42         bool flag = false;
43         hamilton(gp, 1,1 ,solution,pass,flag
44             );
45         if(!flag)
46             cout << "N" << endl;
47     }
48     return 0;
49 }
50 /*
51 4
52 1 2
53 2 3
54 2 4
55 3 4
56 3 1
57 0 0
58 output: 1 3 4 2 1
59 */

```

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      ){

```



```

7     return a.weight > b.weight;
8 }
9 };
10 int find(int x, vector<int>& union_set){
11     if(x != union_set[x])
12         union_set[x] = find(union_set[x],
13                             union_set);
14     return union_set[x];
15 }
16 void merge(int a, int b, vector<int>&
17            union_set){
18     int pa = find(a, union_set);
19     int pb = find(b, union_set);
20     if(pa != pb)
21         union_set[pa] = pb;
22 }
23 void kruskal(priority_queue<edges> pq, int n)
24 {
25     vector<int> union_set(n, 0);
26     for (int i = 0; i < n; i++)
27         union_set[i] = i;
28     int edge = 0;
29     int cost = 0; //evaluate cost of mst
30     while(!pq.empty() && edge < n - 1){
31         edges cur = pq.top();
32         int from = find(cur.from, union_set)
33         ;
34         int to = find(cur.to, union_set);
35         if(from != to){
36             merge(from, to, union_set);
37             edge += 1;
38             cost += cur.weight;
39         }
40         pq.pop();
41     }
42     if(edge < n-1)
43         cout << "No mst" << endl;
44     else
45         cout << cost << endl;
46 }
47 int main(){
48     int n;
49     cin >> n;
50     int a, b, d;
51     priority_queue<edges> pq;
52     while(cin>>a>>b>>d){
53         if(a == -1 && b == -1 && d == -1)
54             break;
55         edges tmp;
56         tmp.from = a;
57         tmp.to = b;
58         tmp.weight = d;
59         pq.push(tmp);
60     }
61     kruskal(pq, n);
62     return 0;
63 }

```

5.9 Minimum Weight Cycle

```

1 // 最小環
2 // 圖上無負環 !!!!

```

```

3 #define INF 99999
4 vector<vector<int>> w, d, p;
5 vector<int> cycle;
6 int c = 0;
7 void trace(int i, int j)
8 {
9     cycle[c++] = i;
10    if (i != j)
11        trace(p[i][j], j);
12 }
13 void init(int n)
14 {
15     for (int i = 0; i < n; ++i)
16         d[i][i] = 0;
17 }
18 void minimum_cycle(int n)
19 {
20     int weight = 1e9;
21     for (int k = 0; k < n; ++k)
22     {
23         for (int i = 0; i < k; ++i)
24             for (int j = 0; j < k; ++j)
25                 if (i != j)
26                     if (w[k][i] + d[i][j] +
27                         w[j][k] < weight)
28                     {
29                         weight = w[k][i] + d[i][j] + w[j][k];
30                         c = 0;
31                         trace(i, j);
32                         cycle[c++] = k;
33                     }
34     }
35     for (int i = 0; i < n; ++i)
36     {
37         for (int j = 0; j < n; ++j)
38             if (d[i][k] + d[k][j] < d[i][j])
39             {
40                 d[i][j] = d[i][k] + d[k][j];
41                 p[i][j] = p[i][k];
42             }
43     }
44 }
45 }
46 if (weight == 1e9)
47     cout << "No exist";
48 else
49 {
50     bug(weight);
51     bug(c);
52     bugarr(cycle);
53 }
54 }
55 void simple_minimum_cycle(int n) // No use
56     vector p
57 {
58     int weight = INF;
59     for (int k = 0; k < n; ++k)
60     {
61         for (int i = 0; i < k; ++i)
62             for (int j = 0; j < k; ++j)
63                 if (i != j)

```

```

64         weight = min(mp[k][i] +
65                     d[i][j] + mp[j][k],
66                     weight);
67     }
68     for (int i = 0; i < n; ++i)
69         for (int j = 0; j < n; ++j)
70             d[i][j] = min(d[i][k] + d[k][j], d[i][j]);
71 }
72 if (weight == INF)
73     cout << "Back to jail\n";
74 else
75     cout << weight << endl;
76 }
77 w.resize(n, vector<int>(n, INF));
78 d.resize(n, vector<int>(n, INF));
79 p.resize(n, vector<int>(n));
80 cycle.resize(n);
81 //Edge input
82 w[a][b] = w;
83 d[a][b] = w;
84 p[a][b] = b;
85 init(n);
86 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
14          start)
15 {
16     vector<bool> pass(n, false);
17     int edge = 0;
18     int cost = 0; //evaluate cost of mst
19     priority_queue<edges> pq;
20     for (int i = 0; i < n; i++)
21     {
22         if (gp[start][i] != inf)
23         {
24             edges tmp;
25             tmp.from = start;
26             tmp.to = i;
27             tmp.weight = gp[start][i];
28             pq.push(tmp);
29         }
30     }
31     pass[start] = true;
32     while (!pq.empty() && edge < n - 1)
33     {
34         edges cur = pq.top();
35         pq.pop();
36         if (!pass[cur.to])

```

```

37     {
38         for (int i = 0; i < n; i++)
39         {
40             if (gp[cur.to][i] != inf)
41             {
42                 edges tmp;
43                 tmp.from = cur.to;
44                 tmp.to = i;
45                 tmp.weight = gp[cur.to][i];
46                 pq.push(tmp);
47             }
48         }
49         pass[cur.to] = true;
50         edge += 1;
51         cost += cur.weight;
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,
64         inf));
65     while (cin >> a >> b >> d)
66     {
67         if (a == -1 && b == -1 && d == -1)
68             break;
69         if (gp[a][b] > d)
70             gp[a][b] = d;
71     }
72     Prim(gp, n, 0);
73     return 0;
74 }

```

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 }
19 void Union(int x, int y)
20 {

```

```

21 int m = Find(x);
22 int n = Find(y);
23 if (m != n)
24 {
25     father[n] = m;
26     people[m] += people[n];
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
 - $a \equiv b \pmod{p} \Leftrightarrow am \equiv bm \pmod{p}$

6.5 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] - 48) * base;
10            base = base * 16;
11        }
12        else if (num[i] >= 'A' && num[i] <= 'F')
13        {
14            temp += (num[i] - 55) * base;
15            base = base * 16;
16        }
17    }
18    return temp;
19 }
20 void DecToHex(int p) //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 }

```

6.6 Log

```

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

```

6.7 Mod

```

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

```

6.8 Mod 性質

加法： $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$

減法： $(a - b) \bmod p = (a \bmod p - b \bmod p + p) \bmod p$

乘法： $(a * b) \bmod p = (a \bmod p * b \bmod p) \bmod p$

次方： $(a^b) \bmod p = ((a \bmod p)^b) \bmod p$

加法結合律： $((a + b) \bmod p + c) \bmod p = (a + (b + c)) \bmod p$

乘法結合律： $((a * b) \bmod p * c) \bmod p = (a * (b * c)) \bmod p$

加法交換律： $(a + b) \bmod p = (b + a) \bmod p$

乘法交換律： $(a * b) \bmod p = (b * a) \bmod p$

結合律： $((a + b) \bmod p * c) = ((a * c) \bmod p + (b * c) \bmod p) \bmod p$

如果 $a \equiv b \pmod{m}$ ，我們會說 a, b 在模 m 下同餘。

以下為性質：

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

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

- 保持基本運算：

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

- 放大縮小模數：

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

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

整數 a 在模 c 下要有模反元素的充分必要條件為 a, c 互質。

模逆元如果存在會有無限個，任意兩相鄰模逆元相差 c 。

費馬小定理

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

歐拉定理

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

Wilson's theorem

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

6.9 PI

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

6.10 Prime table

```
1 const int maxn = sqrt(INT_MAX);
2 vector<int> p;
```

```
3 bitset<maxn> is_notp;
4 void PrimeTable()
5 {
6     is_notp.reset();
7     is_notp[0] = is_notp[1] = 1;
8     for (int i = 2; i <= maxn; ++i)
9     {
10         if (!is_notp[i])
11             p.push_back(i);
12         for (int j = 0; j < (int)p.size(); ++j)
13         {
14             if (i * p[j] > maxn)
15                 break;
16             is_notp[i * p[j]] = 1;
17             if (i % p[j] == 0)
18                 break;
19         }
20     }
21 }
```

6.11 Prime 判斷

```
1 typedef long long ll;
2 ll modmul(ll a, ll b, ll mod)
3 {
4     ll ret = 0;
5     for (; b >= 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 = 1ll;
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     }
```

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

6.12 Round(小數)

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

6.13 二分逼近法

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

```
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else
11            R = M;
12    }
13    printf("%.31f\n", R);
14 }
```

6.14 公式

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

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

6.15 四則運算

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

```

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

```

6.18 數字加法組合

```

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

```

6.16 因數表

```

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

```

6.17 數字乘法組合

```

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

```

6.19 羅馬數字

```

1 int romanToInt(string s)
2 {
3     unordered_map<char, int> T;
4     T['I'] = 1;
5     T['V'] = 5;
6     T['X'] = 10;
7     T['L'] = 50;
8     T['C'] = 100;
9     T['D'] = 500;
10    T['M'] = 1000;
11
12    int sum = T[s.back()];
13    for (int i = s.length() - 2; i >= 0; --i)
14    {
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19    }
20    return sum;
21 }

```

6.20 質因數分解

```

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

```

6.21 質數數量

```

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

```

```

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

```

7 Other

7.1 binary search 三類變化

```

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

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

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

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

```

7.2 Heap sort

```

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

```

7.3 Josephus

```

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

```

7.4 Merge sort

```

1 long long merge(vector<int> &arr, int left,
  int mid, int right)
2 {
3     int *tmp = new int[right - left + 1];
4     long long sum = 0;
5     int l = left, r = mid + 1, m = 0;
6     while (l <= mid && r <= right)
7     {
8         if (arr[l] <= arr[r])
9             tmp[m++] = arr[l++];
10        else
11        {
12            tmp[m++] = arr[r++];
13            sum += mid - l + 1;
14        }
15    }
16    while (l <= mid)
17        tmp[m++] = arr[l++];
18    while (r <= right)
19        tmp[m++] = arr[r++];
20    for (int i = left; i <= right; ++i)
21        arr[i] = tmp[i - left];
22    delete[] tmp;
23    return sum;
24 }
25 long long mergesort(vector<int> &arr, int
  left, int right)
26 {
27     long long sum = 0;
28     // left = 0, right = P.size() - 1
29     if (left < right)
30     {
31         int mid = (left + right) / 2;
32         sum = mergesort(arr, left, mid);
33         sum += mergesort(arr, mid + 1, right
34         );
35         sum += merge(arr, left, mid, right);
36     }
37    return sum; // 回傳為 swap 次數

```

7.5 Quick sort

```

1 int Partition(vector<int> &arr, int front,
  int end)
2 {
3     int pivot = arr[end];
4     int i = front - 1;
5     for (int j = front; j < end; j++)
6     {
7         if (arr[j] < pivot)
8         {
9             i++;
10            swap(arr[i], arr[j]);
11        }
12    }
13    i++;
14    swap(arr[i], arr[end]);
15    return i;
16 }

```

```

17 void QuickSort(vector<int> &arr, int front,
  int end)
18 {
19     // front = 0, end = arr.size() - 1
20     if (front < end)
21     {
22         int pivot = Partition(arr, front,
23         end);
24         QuickSort(arr, front, pivot - 1);
25         QuickSort(arr, pivot + 1, end);
26     }
27 }

```

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 多區間算最大

```

1 bool name(pii a, pii b)
2 { return b.first > a.first; }
3 vector<pii> data;
4 data.pb(pii(a, c)); // 區間 a 到 c
5 sort(data.begin(), data.end(), name); //
6     pair first 從小到大
7 int l = data[0].x, r = data[0].y, res = 0;
8 for (int i = 1; i < data.size(); i++)
9 {
10     if (data[i].x <= r)
11     {
12         if (r < data[i].y)
13             r = data[i].y;
14     }
15     else
16     {
17         res += r - l;
18         l = data[i].x;
19         r = data[i].y;
20     }
21 }
22 res += r - l; // 最大段落不重疊

```

7.8 數獨解法

```

1 int getSquareIndex(int row, int column, int
2     n)
3 {
4     return row / n * n + column / n;
5 }
6 bool backtracking(vector<vector<int>> &board
7     , vector<vector<bool>> &rows, vector<
8     vector<bool>> &cols,
9     vector<vector<bool>> &boxes
10     , int index, int n)
11 {
12     int n2 = n * n;
13     int rowNum = index / n2, colNum = index
14         % n2;
15     if (index >= n2 * n2)
16         return true;
17
18     if (board[rowNum][colNum] != 0)
19         return backtracking(board, rows,
20             cols, boxes, index + 1, n);
21
22     for (int i = 1; i <= n2; i++)
23     {
24         if (!rows[rowNum][i] && !cols[colNum]
25             [i] && !boxes[getSquareIndex(
26                 rowNum, colNum, n)][i])
27         {
28             rows[rowNum][i] = true;
29             cols[colNum][i] = true;
30             boxes[getSquareIndex(rowNum,
31                 colNum, n)][i] = true;
32             board[rowNum][colNum] = i;
33             if (backtracking(board, rows,
34                 cols, boxes, index + 1, n))

```

```

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

```

8.3 Sliding window

```

1 string minWindow(string s, string t)
2 {
3     unordered_map<char, int> letterCnt;

```

```

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

```

8.4 Split

```

1 vector<string> mysplit(string s, string d)
2 {
3     int ps = 0, pe, dl = d.length();
4     string token;
5     vector<string> res;
6     while ((pe = s.find(d, ps)) != string::
7         npos)
8     {
9         token = s.substr(ps, pe - ps);
10         ps = pe + dl;
11         res.push_back(token);
12     }
13     res.push_back(s.substr(ps));
14     return res;

```

9 data structure

9.1 Bigint

```

1 //台大 //非必要請用python
2 struct Bigint
3 {
4     static const int LEN = 60; //
5     static const int BIGMOD = 10000; //10為
6     static const int maxLEN
7     正常位數
8     int s;
9     int v1, v[LEN];
10     // vector<int> v;
11     Bigint() : s(1) { v1 = 0; }

```

```

10 Bigint(long long a)
11 {
12     s = 1;
13     vl = 0;
14     if (a < 0)
15     {
16         s = -1;
17         a = -a;
18     }
19     while (a)
20     {
21         push_back(a % BIGMOD);
22         a /= BIGMOD;
23     }
24 }
25 Bigint(string str)
26 {
27     s = 1;
28     vl = 0;
29     int stPos = 0, num = 0;
30     if (!str.empty() && str[0] == '-')
31     {
32         stPos = 1;
33         s = -1;
34     }
35     for (int i = str.length() - 1, q = 1; i >= stPos; i--)
36     {
37         num += (str[i] - '0') * q;
38         if ((q *= 10) >= BIGMOD)
39         {
40             push_back(num);
41             num = 0;
42             q = 1;
43         }
44     }
45     if (num)
46         push_back(num);
47     n();
48 }
49 int len() const
50 {
51     return vl; //return SZ(v);
52 }
53 bool empty() const { return len() == 0; }
54 void push_back(int x)
55 {
56     v[vl++] = x; //v.PB(x);
57 }
58 void pop_back()
59 {
60     vl--; //v.pop_back();
61 }
62 int back() const
63 {
64     return v[vl - 1]; //return v.back();
65 }
66 void n()
67 {
68     while (!empty() && !back())
69         pop_back();
70 }
71 void resize(int nl)
72 {
73     vl = nl;
74     fill(v, v + vl, 0); //fill(ALL(v), 0);
75 }
76 void print() const
77 {
78     if (empty())
79     {
80         putchar('0');
81         return;
82     }
83     if (s == -1)
84         putchar('-');
85     printf("%d", back());
86     for (int i = len() - 2; i >= 0; i--)
87         printf("%.4d", v[i]);
88 }
89 friend std::ostream &operator<<(std::ostream &out, const Bigint &a)
90 {
91     if (a.empty())
92     {
93         out << "0";
94         return out;
95     }
96     if (a.s == -1)
97         out << "-";
98     out << a.back();
99     for (int i = a.len() - 2; i >= 0; i--)
100     {
101         char str[10];
102         snprintf(str, 5, "%.4d", a.v[i]);
103         out << str;
104     }
105     return out;
106 }
107 int cp3(const Bigint &b) const
108 {
109     if (s != b.s)
110         return s - b.s;
111     if (s == -1)
112         return -(*this).cp3(-b);
113     if (len() != b.len())
114         return len() - b.len(); //int
115     for (int i = len() - 1; i >= 0; i--)
116         if (v[i] != b.v[i])
117             return v[i] - b.v[i];
118     return 0;
119 }
120 bool operator<(const Bigint &b) const
121 {
122     return cp3(b) < 0;
123 }
124 bool operator<=(const Bigint &b) const
125 {
126     return cp3(b) <= 0;
127 }
128 bool operator==(const Bigint &b) const
129 {
130     return cp3(b) == 0;
131 }
132 bool operator!=(const Bigint &b) const
133 {
134     return cp3(b) != 0;
135 }
136 bool operator>(const Bigint &b) const
137 {
138     return cp3(b) > 0;
139 }
140 bool operator>=(const Bigint &b) const
141 {
142     return cp3(b) >= 0;
143 }
144 Bigint operator-() const
145 {
146     Bigint r = (*this);
147     r.s = -r.s;
148     return r;
149 }
150 Bigint operator+(const Bigint &b) const
151 {
152     if (s == -1)
153         return -(*this) + (-b);
154     if (b.s == -1)
155         return (*this) - (-b);
156     Bigint r;
157     int nl = max(len(), b.len());
158     r.resize(nl + 1);
159     for (int i = 0; i < nl; i++)
160     {
161         if (i < len())
162             r.v[i] += v[i];
163         if (i < b.len())
164             r.v[i] += b.v[i];
165         if (r.v[i] >= BIGMOD)
166         {
167             r.v[i + 1] += r.v[i] / BIGMOD;
168             r.v[i] %= BIGMOD;
169         }
170     }
171     r.n();
172     return r;
173 }
174 Bigint operator-(const Bigint &b) const
175 {
176     if (s == -1)
177         return -(*this) - (-b);
178     if (b.s == -1)
179         return (*this) + (-b);
180     if ((*this) < b)
181         return -b - (*this);
182     Bigint r;
183     r.resize(len());
184     for (int i = 0; i < len(); i++)
185     {
186         r.v[i] += v[i];
187         if (i < b.len())
188             r.v[i] -= b.v[i];
189         if (r.v[i] < 0)
190         {
191             r.v[i] += BIGMOD;
192             r.v[i + 1]--;
193         }
194     }
195     r.n();
196     return r;
197 }
198 Bigint operator*(const Bigint &b)
199 {
200     Bigint r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j++)
206         {
207             r.v[i + j] += v[i] * b.v[j];
208             if (r.v[i + j] >= BIGMOD)
209             {
210                 r.v[i + j + 1] += r.v[i + j] / BIGMOD;
211                 r.v[i + j] %= BIGMOD;
212             }
213         }
214     }
215     r.n();
216     return r;
217 }
218 Bigint operator/(const Bigint &b)
219 {
220     Bigint r;
221     r.resize(max(1, len() - b.len() + 1));
222     int oriS = s;
223     Bigint b2 = b; // b2 = abs(b)
224     s = b2.s; r.s = 1;
225     for (int i = r.len() - 1; i >= 0; i--)
226     {
227         int d = 0, u = BIGMOD - 1;
228         while (d < u)
229         {
230             int m = (d + u + 1) >> 1;
231             r.v[i] = m;
232             if ((r * b2) > (*this))
233                 u = m - 1;
234             else
235                 d = m;
236         }
237         r.v[i] = d;
238     }
239     s = oriS;
240     r.s = s * b.s;
241     r.n();
242     return r;
243 }
244 Bigint operator%(const Bigint &b)
245 {
246     return (*this) - (*this) / b * b;
247 }
248 };

```

9.2 DisjointSet

```

1 struct DisjointSet {
2     int p[maxn], sz[maxn], n, cc;
3     vector<pair<int*, int>> his;
4     vector<int> sh;
5     void init(int _n) {
6         n = _n; cc = n;
7         for (int i = 0; i < n; ++i) sz[i] = 1, p[i] = i;

```

```

8     sh.clear(); his.clear();
9 }
10 void assign(int *k, int v) {
11     his.emplace_back(k, *k);
12     *k = v;
13 }
14 void save() {
15     sh.push_back((int)his.size());
16 }
17 void undo() {
18     int last = sh.back(); sh.pop_back();
19     while (his.size() != last) {
20         int *k, v;
21         tie(k, v) = his.back(); his.
22             pop_back();
23         *k = v;
24     }
25 }
26 int find(int x) {
27     if (x == p[x]) return x;
28     return find(p[x]);
29 }
30 void merge(int x, int y) {
31     x = find(x); y = find(y);
32     if (x == y) return;
33     if (sz[x] > sz[y]) swap(x, y);
34     assign(&sz[y], sz[x] + sz[y]);
35     assign(&p[x], y);
36     assign(&cc, cc - 1);
37 }

```

9.3 Matirx

```

1 template <typename T>
2 struct Matrix
3 {
4     using rt = std::vector<T>;
5     using mt = std::vector<rt>;
6     using matrix = Matrix<T>;
7     int r, c; // [r][c]
8     mt m;
9     Matrix(int r, int c) : r(r), c(c), m(r,
10         rt(c)) {}
11     Matrix(mt a) { m = a, r = a.size(), c =
12         a[0].size(); }
13     rt &operator[](int i) { return m[i]; }
14     matrix operator+(const matrix &a)
15     {
16         matrix rev(r, c);
17         for (int i = 0; i < r; ++i)
18             for (int j = 0; j < c; ++j)
19                 rev[i][j] = m[i][j] + a.m[i][j];
20         return rev;
21     }
22     matrix operator-(const matrix &a)
23     {
24         matrix rev(r, c);
25         for (int i = 0; i < r; ++i)
26             for (int j = 0; j < c; ++j)
27                 rev[i][j] = m[i][j] - a.m[i][j];
28     }
29 }

```

```

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

```

```

88     for (auto &j : m[i])
89         j /= lazy[i];
90 }
91 return det;
92 }
93 };

```

9.4 Trie

```

1 // biginteger字典數
2 struct BigInteger{
3     static const int BASE = 100000000;
4     static const int WIDTH = 8;
5     vector<int> s;
6     BigInteger(long long num = 0){
7         *this = num;
8     }
9     BigInteger operator = (long long num){
10         s.clear();
11         do{
12             s.push_back(num % BASE);
13             num /= BASE;
14         }while(num > 0);
15         return *this;
16     }
17     BigInteger operator = (const string& str)
18     {
19         s.clear();
20         int x, len = (str.length() - 1) /
21             WIDTH + 1;
22         for(int i = 0; i < len; i++){
23             int end = str.length() - i*WIDTH;
24             ;
25             int start = max(0, end-WIDTH);
26             sscanf(str.substr(start, end-
27                 start).c_str(), "%d", &x);
28             s.push_back(x);
29         }
30         return *this;
31     }
32     BigInteger operator + (const BigInteger&
33         b) const{
34         BigInteger c;
35         c.s.clear();
36         for(int i = 0, g = 0; i < s.size() && i
37             <= b.s.size()) break;
38             int x = g;
39             if(i < s.size()) x+=s[i];
40             if(i < b.s.size()) x+=b.s[i];
41             c.s.push_back(x % BASE);
42             g = x / BASE;
43         }
44         return c;
45     }
46 };
47 ostream& operator << (ostream &out, const
48     BigInteger& x){
49     out << x.s.back();
50     for(int i = x.s.size()-2; i >= 0; i--){
51         char buf[20];
52         sprintf(buf, "%08d", x.s[i]);
53         for(int j = 0; j < strlen(buf); j++){
54             out << buf[j];
55         }
56     }
57     return out;
58 }
59 istream& operator >> (istream &in,
60     BigInteger& x){
61     string s;
62     if(!(in >> s))
63         return in;
64     x = s;
65     return in;
66 }
67 struct Trie{
68     int c[500005][10];
69     int val[500005];
70     int sz;
71     int getIndex(char c){
72         return c - '0';
73     }
74     void init(){
75         memset(c[0], 0, sizeof(c[0]));
76         memset(val, -1, sizeof(val));
77         sz = 1;
78     }
79     void insert(BigInteger x, int v){
80         int u = 0;
81         int max_len_count = 0;
82         int firstNum = x.s.back();
83         char firstBuf[20];
84         sprintf(firstBuf, "%d", firstNum);
85         for(int j = 0; j < strlen(firstBuf); j++){
86             int index = getIndex(firstBuf[j]);
87             if(!c[u][index]){
88                 memset(c[sz], 0, sizeof(c[
89                     sz]));
90                 val[sz] = v;
91                 c[u][index] = sz++;
92             }
93             u = c[u][index];
94             max_len_count++;
95         }
96         for(int i = x.s.size()-2; i >= 0; i--){
97             char buf[20];
98             sprintf(buf, "%08d", x.s[i]);
99             for(int j = 0; j < strlen(buf); j++){
100                 && max_len_count < 50; 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             }
112             if(max_len_count >= 50){

```

```

106         break;
107     }
108 }
109 }
110 int find(const char* s){
111     int u = 0;
112     int n = strlen(s);
113     for(int i = 0 ; i < n; ++i)
114     {
115         int index = getIndex(s[i]);
116         if(!c[u][index]){
117             return -1;
118         }
119         u = c[u][index];
120     }
121     return val[u];
122 }
123 }

```

```

36     cout << "/" << d;
37 }
38 };

```

9.5 分數

```

1 typedef long long ll;
2 struct fraction
3 {
4     ll n, d;
5     fraction(const ll &n = 0, const ll &d =
6         1) : n(_n), d(_d)
7     {
8         ll t = __gcd(n, d);
9         n /= t, d /= t;
10        if (d < 0)
11            n = -n, d = -d;
12    }
13    fraction operator-() const
14    {
15        return fraction(-n, d);
16    }
17    fraction operator+(const fraction &b)
18        const
19    {
20        return fraction(n * b.d + b.n * d, d * b
21            .d);
22    }
23    fraction operator-(const fraction &b)
24        const
25    {
26        return fraction(n * b.d - b.n * d, d * b
27            .d);
28    }
29    fraction operator*(const fraction &b)
30        const
31    {
32        return fraction(n * b.n, d * b.d);
33    }
34    fraction operator/(const fraction &b)
35        const
36    {
37        return fraction(n * b.d, d * b.n);
38    }
39    void print()
40    {
41        cout << n;
42        if (d != 1)

```

TO DO WRITING NOT THINKING

Contents

1 Basic	1	2.8 LIS	2	5.6 Floyd-warshall	8	6.19 羅馬數字	12
1.1 Basic codeblock setting . . .	1	2.9 LPS	2	5.7 Hamilton_cycle	8	6.20 質因數分解	12
1.2 Basic vim setting	1	2.10 Max_subarray	2	5.8 Kruskal	8	6.21 質數數量	12
1.3 Code Template	1	2.11 Money problem	2	5.9 Minimum Weight Cycle . . .	9		
1.4 Python	1			5.10 Prim	9	7 Other	13
1.5 Range data	1	3 Flow & matching	3	5.11 Union_find	9	7.1 binary search 三類變化	13
1.6 Some Function	1	3.1 Dinic	3			7.2 Heap sort	13
1.7 Time	1	3.2 Edmonds_karp	3	6 Mathematics	10	7.3 Josephus	13
2 DP	1	3.3 hungarian	3	6.1 Catalan	10	7.4 Merge sort	13
2.1 3 維 DP 思路	1	3.4 Maximum_matching	3	6.2 Combination	10	7.5 Quick sort	13
2.2 Knapsack Bounded	1	3.5 MFlow Model	4	6.3 Extended Euclidean	10	7.6 Weighted Job Scheduling . .	13
2.3 Knapsack sample	1			6.4 Fermat	10	7.7 多區間算最大	14
2.4 Knapsack Unbounded	2	4 Geometry	4	6.5 Hex to Dec	10	7.8 數獨解法	14
2.5 LCIS	2	4.1 Circle Intersect	4	6.6 Log	10		
2.6 LCS	2	4.2 Closest Pair	4	6.7 Mod	10	8 String	14
2.7 LIS O(Nlog(N))	2	4.3 Line	5	6.8 Mod 性質	11	8.1 KMP	14
		4.4 max_cover_rectangle	5	6.9 PI	11	8.2 Min Edit Distance	14
		4.5 Point	5	6.10 Prime table	11	8.3 Sliding window	14
		4.6 Polygon	6	6.11 Prime 判斷	11	8.4 Split	14
		4.7 Triangle	7	6.12 Round(小數)	11		
		5 Graph	7	6.13 二分逼近法	11	9 data structure	14
		5.1 Bellman-Ford	7	6.14 公式	11	9.1 Bigint	14
		5.2 BFS-queue	7	6.15 四則運算	11	9.2 DisjointSet	15
		5.3 DFS-rec	8	6.16 因數表	12	9.3 Matirx	16
		5.4 Dijkstra	8	6.17 數字乘法組合	12	9.4 Trie	16
		5.5 Euler circuit	8	6.18 數字加法組合	12	9.5 分數	17