

1 Basic

1.1 Code Template

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef unsigned long long ull;
5 typedef pair<int, int> pii;
6 #define x first
7 #define y second
8 #define pb push_back
9 #define len length()
10 #define all(p) p.begin(), p.end()
11 #define endl '\n'
12 #define bug(x) cout << "value of " << #x <<
13   " is " << x << endl;
14 #define bugarr(x) \
15   for (auto i : x) \
16     cout << i << ' '; \
17   cout << endl;
18 int main()
19 {
20   ios::sync_with_stdio(0);
21   cin.tie(0);
22   return 0;
23 }
```

1.2 Codeblock setting

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

1.3 IO_fast

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

1.4 Python

```
1 //輸入
2 import sys
3 line = sys.stdin.readline() // 會讀到換行
4 input().strip()
5
6 array = [0] * (N) //N個0
7 range(0, N) // 0 ~ N-1
8 D, R, N = map(int, line[:-1].split()) // 分
   三個 int 變數
9
10 pow(a, b, c) // a ^ b % c
11
12 print(*objects, sep = ' ', end = '\n')
13 // objects -- 可以一次輸出多個對象
14 // sep -- 分開多個objects
15 // end -- 默認值是\n
16
17 // EOF break
18 try:
19     while True:
20         //input something
21 except EOFError:
22     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
   9223372036854775807)
6 unsigned long long (0 to
   18446744073709551615)
```

1.6 Some Function

```
1 round(double f); // 四捨五入
2 ceil(double f); // 進入
3 floor(double f); // 捨去
4 __builtin_popcount(int n); // 32bit有多少 1
5 to_string(int s); // int to string
6
7 /** 全排列要先 sort !!! **/
8 next_permutation(num.begin(), num.end());
9 prev_permutation(num.begin(), num.end());
10 //用binary search找大於或等於val的最小值的位
   置
11 vector<int>::iterator it = lower_bound(v.
   begin(), v.end(), val);
12 //用binary search找大於val的最小值的位置
13 vector<int>::iterator it = upper_bound(v.
   begin(), v.end(), val);
14 /*queue*/
15 queue<datatype> q;
16 front(); //取出最前面的值(沒有移除掉)*/
```

```
18 back(); //取出最後面的值(沒有移除掉)*/
19 pop(); //移除最前面的值*/
20 push(); //新增值到最後面*/
21 empty(); //回傳bool,檢查是不是空的queue*/
22 size(); //queue 的大小*/
23
24 /*stack*/
25 stack<datatype> s;
26 top(); //取出最上面的值(沒有移除掉)*/
27 pop(); //移除最上面的值*/
28 push(); //新增值到最上面*/
29 empty(); //bool 檢查是不是空*/
30 size(); //stack 的大小*/
31
32 /*unordered_set*/
33 unordered_set<datatype> s;
34 unordered_set<datatype> s(arr, arr + n);
35 /*initial with array*/
36 insert(); //插入值*/
37 erase(); //刪除值*/
38 empty(); //bool 檢查是不是空*/
39 count(); //判斷元素存在回傳1 無則回傳0*/
```

1.7 Time

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

1.8 Vim setting

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

2 DP

2.1 3 維 DP 思路

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

2.2 Knapsack Bounded

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

2.3 Knapsack sample

```
1 int Knapsack(vector<int> weight, vector<int>
   value, int bag_Weight)
2 {
3     // vector<int> weight = {1, 3, 4};
4     // vector<int> value = {15, 20, 30};
5     // int bagWeight = 4;
6     vector<vector<int>> dp(weight.size(),
7         vector<int>(bagWeight + 1, 0));
8     for (int j = weight[0]; j <= bagWeight;
9         j++)
10         dp[0][j] = value[0];
11     // weight數組的大小就是物品個數
12     for (int i = 1; i < weight.size(); i++)
13     { // 遍歷物品
14         for (int j = 0; j <= bagWeight; j++)
15         { // 遍歷背包容量
16             if (j < weight[i]) dp[i][j] = dp
17                 [i - 1][j];
18             else dp[i][j] = max(dp[i - 1][j],
19                 dp[i - 1][j - weight[i]]
20                 + value[i]);
21         }
22     }
23     cout << dp[weight.size() - 1][bagWeight]
24         << endl;
25 }
```

2.4 Knapsack Unbounded

```

1 const int N = 100, W = 100000;
2 int cost[N], weight[N];
3 int c[W + 1];
4 void knapsack(int n, int w)
5 {
6     memset(c, 0, sizeof(c));
7     for (int i = 0; i < n; ++i)
8         for (int j = weight[i]; j <= w; ++j)
9             c[j] = max(c[j], c[j - weight[i]] + cost[i]);
10
11     cout << "最高的價值為" << c[w];
12 }

```

2.5 LCIS

```

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

```

2.6 LCS

```

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

```

```

9             if (Ans[i - 1] == num[j - 1])
10                 LCS[i][j] = LCS[i - 1][j -
11                   1] + 1;
12             else
13                 LCS[i][j] = max(LCS[i - 1][j]
14                   , LCS[i][j - 1]);
15         }
16     }
17     cout << LCS[N][M] << '\n';
18     //列印 LCS
19     int n = N, m = M;
20     vector<string> k;
21     while (n && m)
22     {
23         if (LCS[n][m] != max(LCS[n - 1][m],
24           LCS[n][m - 1]))
25         {
26             k.push_back(Ans[n - 1]);
27             n--;
28             m--;
29         }
30         else if (LCS[n][m] == LCS[n - 1][m])
31             n--;
32         else if (LCS[n][m] == LCS[n][m - 1])
33             m--;
34     }
35     reverse(k.begin(), k.end());
36     for (auto i : k)
37         cout << i << " ";
38     cout << endl;
39     return LCS[N][M];
40 }

```

2.7 LIS

```

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

```

```

27     if (dp[i] > res)
28     {
29         res = dp[i];
30         index = i;
31     }
32 }
33 cout << res << endl; // length
34 printLIS(arr, pos, index);
35 for (int i = 0; i < ans.size(); i++)
36 {
37     cout << ans[i];
38     if (i != ans.size() - 1)
39         cout << ' ';
40 }
41 cout << '\n';
42 }

```

2.8 LPS

```

1 void LPS(string s)
2 {
3     int maxlen = 0, l, r;
4     int n = s.size();
5     for (int i = 0; i < n; i++)
6     {
7         int x = 0;
8         while ((s[i - x] == s[i + x]) && (i
9           - x >= 0) && (i + x < n)) //odd
10             x++;
11         x--;
12         if (2 * x + 1 > maxlen)
13         {
14             maxlen = 2 * x + 1;
15             l = i - x;
16             r = i + x;
17         }
18         x = 0;
19         while ((s[i - x] == s[i + 1 + x]) &&
20           (i - x >= 0) && (i + 1 + x < n)) //even length
21             x++;
22         if (2 * x > maxlen)
23         {
24             maxlen = 2 * x;
25             l = i - x + 1;
26             r = i + x;
27         }
28     }
29     cout << maxlen << '\n'; // 最後長度
30     cout << l + 1 << ' ' << r + 1 << '\n';
31     // 頭到尾
32 }

```

2.9 Max_subarray

```

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

```

```

4     for (int i = 1; i < nums.size(); i++) {
5         local_max = max(nums[i], nums[i] +
6           local_max);
7         global_max = max(local_max,
8           global_max);
9     }
10     return global_max;
11 }

```

2.10 Money problem

```

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

```

```

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

```

3 Flow & matching

3.1 Dinic

```

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

```

3.2 Edmonds_karp

```

1 /*Flow - Edmonds-karp*/
2 /*Based on UVa820*/
3 #define inf 1000000
4 int getMaxFlow(vector<vector<int>> &capacity
5     , int s, int t, int n){
6     int ans = 0;
7     vector<vector<int>> residual(n+1, vector<
8         int>(n+1, 0)); //residual network
9     while(true){
10        vector<int> bottleneck(n+1, 0);
11        bottleneck[s] = inf;
12        queue<int> q;
13        q.push(s);
14        vector<int> pre(n+1, 0);
15        while(!q.empty() && bottleneck[t] == 0){
16            int cur = q.front();
17            q.pop();
18            for(int i = 1; i <= n ; i++){
19                if(bottleneck[i] == 0 && capacity[
20                    cur][i] > residual[cur][i]){
21                    q.push(i);
22                    pre[i] = cur;
23                    bottleneck[i] = min(bottleneck[cur]
24                        , capacity[cur][i] - residual
25                        [cur][i]);
26                }
27            }
28        }
29        if(bottleneck[t] == 0) break;
30        int cur = t;
31        while(cur != s){
32            int p = pre[cur];
33            residual[p][cur] += bottleneck[t];
34            residual[cur][p] -= bottleneck[t];
35            cur = p;
36        }
37        ans += bottleneck[t];
38    }
39    return ans;
40 }

```

3.3 hungarian

```

1 /*bipartite - hungarian*/
2 struct Graph{
3     static const int MAXN = 5003;
4     vector<int> G[MAXN];
5     int n, match[MAXN], vis[MAXN];
6     void init(int _n){
7         n = _n;
8         for (int i=0; i<n; i++) G[i].clear()
9             ;
10    }
11    bool dfs(int u){
12        for (int v:G[u]){
13            if (vis[v]) continue;
14            vis[v]=true;
15            if (match[v]==-1 || dfs(match[v]
16                )){
17                match[v] = u;
18                match[u] = v;
19                return true;
20            }
21        }
22        return false;
23    }
24    int solve(){
25        int res = 0;
26        memset(match,-1,sizeof(match));
27        for (int i=0; i<n; i++){
28            if (match[i]==-1){
29                memset(vis,0,sizeof(vis));
30                if ( dfs(i) ) res++;
31            }
32        }
33        return res;
34    }
35 } graph;

```

3.4 Maximum_matching

```

1 /*bipartite - maximum matching*/
2 bool dfs(vector<vector<bool>> res,int node,
3     vector<int>& x, vector<int>& y, vector<
4     bool> pass){
5     for (int i = 0; i < res[0].size(); i++){
6         if(res[node][i] && !pass[i]){
7             pass[i] = true;
8             if(y[i] == -1 || dfs(res,y[i],x,
9                 y,pass)){
10                 x[node] = i;
11                 y[i] = node;
12                 return true;
13             }
14         }
15     }
16     return false;
17 }
18 int main(){
19     int n,m,l;
20     while(cin>>n>>m>>l){
21         vector<vector<bool>> res(n, vector<
22             bool>(m, false));
23         for (int i = 0; i < l; i++){
24             int a, b;
25             cin >> a >> b;
26             res[a][b] = true;
27         }
28         int ans = 0;
29         vector<int> x(n, -1);
30         vector<int> y(n, -1);
31         for (int i = 0; i < n; i++){
32             vector<bool> pass(n, false);
33             if(dfs(res,i,x,y,pass))
34                 ans += 1;
35         }
36         cout << ans << endl;
37     }
38     return 0;
39 }
40 /*
41 input:
42 4 3 5 //n matching m, l links
43 0 0
44 0 2
45 1 0
46 2 1

```

```

43 3 1
44 answer is 3
45 */

```

3.5 MFlow Model

```

1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 1000000000000LL;
7
8     int n, m, s, t, tot, tim;
9     int first[N], next[M];
10    int u[M], v[M], cur[N], vi[N];
11    ll cap[M], flow[M], dis[N];
12    int que[N + N];
13
14    void Clear()
15    {
16        tot = 0;
17        tim = 0;
18        for (int i = 1; i <= n; ++i)
19            first[i] = -1;
20    }
21    void Add(int from, int to, ll cp, ll flw)
22    {
23        u[tot] = from;
24        v[tot] = to;
25        cap[tot] = cp;
26        flow[tot] = flw;
27        next[tot] = first[u[tot]];
28        first[u[tot]] = tot;
29        ++tot;
30    }
31    bool bfs()
32    {
33        ++tim;
34        dis[s] = 0;
35        vi[s] = tim;
36
37        int head, tail;
38        head = tail = 1;
39        que[head] = s;
40        while (head <= tail)
41        {
42            for (int i = first[que[head]]; i
43                != -1; i = next[i])
44            {
45                if (vi[v[i]] != tim && cap[i]
46                    > flow[i])
47                {
48                    vi[v[i]] = tim;
49                    dis[v[i]] = dis[que[head]] + 1;
50                    que[++tail] = v[i];
51                }
52            }
53            ++head;
54        }
55        return vi[t] == tim;

```

```

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

```

4 Geometry

4.1 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;

```

4.2 Line

```

1 template <typename T>
2 struct line
3 {
4     line() {}
5     point<T> p1, p2;
6     T a, b, c; //ax+by+c=0
7     line(const point<T> &x, const point<T> &
8         y) : p1(x), p2(y) {}
9     void pton()
10    { //轉成一般式
11        a = p1.y - p2.y;
12        b = p2.x - p1.x;
13        c = -a * p1.x - b * p1.y;
14    }
15    T ori(const point<T> &p) const
16    { //點和有向直線的關係 · >0左邊、=0在線上
17        <0右邊
18        return (p2 - p1).cross(p - p1);
19    }
20    T btw(const point<T> &p) const
21    { //點投影落在線段上<=0
22        return (p1 - p).dot(p2 - p);
23    }
24    bool point_on_segment(const point<T> &p)
25        const
26    { //點是否在线段上
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();

```

```

36    }
37    T tmp = v.cross(v1);
38    return tmp * tmp / v.abs2();
39    }
40    T seg_dis2(const line<T> &l) const
41    { //兩線段距離平方
42        return min({dis2(l.p1, 1), dis2(l.p2
43            , 1), l.dis2(p1, 1), l.dis2(p2,
44                1)});
45    }
46    point<T> projection(const point<T> &p)
47        const
48    { //點對直線的投影
49        point<T> n = (p2 - p1).normal();
50        return p - n * (p - p1).dot(n) / n.
51            abs2();
52    }
53    point<T> mirror(const point<T> &p) const
54    {
55        //點對直線的鏡射 · 要先呼叫pton轉成一般式
56        point<T> R;
57        T d = a * a + b * b;
58        R.x = (b * b * p.x - a * a * p.x - 2
59            * a * b * p.y - 2 * a * c) / d;
60        R.y = (a * a * p.y - b * b * p.y - 2
61            * a * b * p.x - 2 * b * c) / d;
62        return R;
63    }
64    bool equal(const line &l) const
65    { //直線相等
66        return ori(l.p1) == 0 && ori(l.p2)
67            == 0;
68    }
69    bool parallel(const line &l) const
70    {
71        return (p1 - p2).cross(l.p1 - l.p2)
72            == 0;
73    }
74    bool cross_seg(const line &l) const
75    {
76        return (p2 - p1).cross(l.p1 - p1) *
77            (p2 - p1).cross(l.p2 - p1) <= 0;
78        //直線是否交線段
79    }
80    int line_intersect(const line &l) const
81    { //直線相交情況 · -1無限多點、1交於一點、0不相交
82        return parallel(l) ? (ori(l.p1) == 0
83            ? -1 : 0) : 1;
84    }
85    int seg_intersect(const line &l) const
86    {
87        T c1 = ori(l.p1), c2 = ori(l.p2);
88        T c3 = l.ori(p1), c4 = l.ori(p2);
89        if (c1 == 0 && c2 == 0)
90        { //共線
91            bool b1 = btw(l.p1) >= 0, b2 =
92                btw(l.p2) >= 0;
93            T a3 = l.btw(p1), a4 = l.btw(p2)
94                ;
95            if (b1 && b2 && a3 == 0 && a4 >=
96                0)
97                return 2;

```

```

84     if (b1 && b2 && a3 >= 0 && a4 == 24
85         0)
86         return 3;
87     if (b1 && b2 && a3 >= 0 && a4 >= 25
88         0)
89         return 0;
90     return -1; //無限交點
91 }
92 else if (c1 * c2 <= 0 && c3 * c4 <= 26
93     0)
94     return 1;
95     return 0; //不相交
96 }
97 point<T> line_intersection(const line &l 27
98     ) const
99 { /*直線交點*/
100     point<T> a = p2 - p1, b = l.p2 - l. 28
101     p1, s = l.p1 - p1;
102     //if(a.cross(b)==0)return INF;
103     return p1 + a * (s.cross(b) / a. 29
104         cross(b));
105 }
106 point<T> seg_intersection(const line &l 30
107     const
108     { //線段交點
109     int res = seg_intersect(l);
110     if (res <= 0)
111         assert(0);
112     if (res == 2)
113         return p1;
114     if (res == 3)
115         return p2;
116     return line_intersection(l);
117 }
118 };

```

4.3 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)),
47         dot(b)));
48 }
49 T getA() const
50 {
51     //對x軸的弧度
52     T A = atan2(y, x); //超過180度會變負
53     if (A <= -PI / 2)
54         A += PI * 2;
55     return A;
56 }
57 };

```

4.4 Polygon

```

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

```

```

25         return point<T>(cx / 3 / w, cy / 3 / 23
26             w);
27     }
28     char ahas(const point<T> &t) const
29     { //點是否在簡單多邊形內，是的話回傳1、
30         在邊上回傳-1、否則回傳0
31         bool c = 0;
32         for (int i = 0, j = p.size() - 1; i
33             < p.size(); j = i++)
34             if (line<T>(p[i], p[j]).
35                 point_on_segment(t))
36                 return -1;
37             else if ((p[i].y > t.y) != (p[j]
38                 ].y > t.y) &&
39                 t.x < (p[j].x - p[i].x)
40                     * (t.y - p[i].y) /
41                     (p[j].y - p[i].y)
42                     + p[i].x)
43                 c = !c;
44         return c;
45     }
46     char point_in_convex(const point<T> &x)
47     const
48     {
49         int l = 1, r = (int)p.size() - 2;
50         while (l <= r)
51         { //點是否在凸多邊形內，是的話回傳1
52             、在邊上回傳-1、否則回傳0
53             int mid = (l + r) / 2;
54             T a1 = (p[mid] - p[0]).cross(x -
55                 p[0]);
56             T a2 = (p[mid + 1] - p[0]).cross
57                 (x - p[0]);
58             if (a1 >= 0 && a2 <= 0)
59             {
60                 T res = (p[mid + 1] - p[mid]
61                     ).cross(x - p[mid]);
62                 return res > 0 ? 1 : (res >=
63                     0 ? -1 : 0);
64             }
65             else if (a1 < 0)
66                 r = mid - 1;
67             else
68                 l = mid + 1;
69         }
70         return 0;
71     }
72     vector<T> getA() const
73     { //凸包邊對x軸的夾角
74         vector<T> res; //一定是遞增的
75         for (size_t i = 0; i < p.size(); ++i
76             )
77             res.push_back((p[(i + 1) % p.
78                 size()] - p[i]).getA());
79         return res;
80     }
81     bool line_intersect(const vector<T> &A,
82         const line<T> &l) const
83     { //O(logN)
84         int f1 = upper_bound(A.begin(), A.
85             end(), (l.p1 - l.p2).getA()) - A
86             .begin();
87         int f2 = upper_bound(A.begin(), A.
88             end(), (l.p2 - l.p1).getA()) - A
89             .begin();

```

```

90         return l.cross_seg(line<T>(p[f1], p[
91             f2]));
92     }
93     polygon cut(const line<T> &l) const
94     { //凸包對直線切割，得到直線l左側的凸包
95         polygon ans;
96         for (int n = p.size(), i = n - 1, j
97             = 0; j < n; i = j++)
98         {
99             if (l.ori(p[i]) >= 0)
100             {
101                 ans.p.push_back(p[i]);
102                 if (l.ori(p[j]) < 0)
103                     ans.p.push_back(l.
104                         line_intersection(
105                             line<T>(p[i], p[j]))
106                             );
107             }
108             else if (l.ori(p[j]) > 0)
109                 ans.p.push_back(l.
110                     line_intersection(line<T>
111                         >(p[i], p[j])));
112         }
113         return ans;
114     }
115     static bool graham_cmp(const point<T> &a
116         , const point<T> &b)
117     { //凸包排序函數 // 起始點不同
118         // return (a.x < b.x) || (a.x == b.x
119             && a.y < b.y); //最左下角開始
120         return (a.y < b.y) || (a.y == b.y &&
121             a.x < b.x); //Y最小開始
122     }
123     void graham(vector<point<T>> &s)
124     { //凸包 Convexhull 2D
125         sort(s.begin(), s.end(), graham_cmp)
126             ;
127         p.resize(s.size() + 1);
128         int m = 0;
129         // cross >= 0 順時針，cross <= 0 逆
130             時針旋轉
131         for (size_t i = 0; i < s.size(); ++i
132             )
133         {
134             while (m >= 2 && (p[m - 1] - p[m
135                 - 2]).cross(s[i] - p[m -
136                 2]) <= 0)
137                 --m;
138             p[m++] = s[i];
139         }
140         for (int i = s.size() - 2, t = m +
141             1; i >= 0; --i)
142         {
143             while (m >= t && (p[m - 1] - p[m
144                 - 2]).cross(s[i] - p[m -
145                 2]) <= 0)
146                 --m;
147             p[m++] = s[i];
148         }
149         if (s.size() > 1) // 重複頭一次需扣
150             掉
151             --m;
152         p.resize(m);
153     }

```



```

112 T diam()
113 { //直徑
114     int n = p.size(), t = 1;
115     T ans = 0;
116     p.push_back(p[0]);
117     for (int i = 0; i < n; i++)
118     {
119         point<T> now = p[i + 1] - p[i];
120         while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
121             t = (t + 1) % n;
122         ans = max(ans, (p[i] - p[t]).abs2());
123     }
124     return p.pop_back(), ans;
125 }
126 T min_cover_rectangle()
127 { //最小覆蓋矩形
128     int n = p.size(), t = 1, r = 1, l;
129     if (n < 3)
130         return 0; //也可以做最小周長矩形
131     T ans = 1e99;
132     p.push_back(p[0]);
133     for (int i = 0; i < n; i++)
134     {
135         point<T> now = p[i + 1] - p[i];
136         while (now.cross(p[t + 1] - p[i]) > now.cross(p[t] - p[i]))
137             t = (t + 1) % n;
138         while (now.dot(p[r + 1] - p[i]) > now.dot(p[r] - p[i]))
139             r = (r + 1) % n;
140         if (!i)
141             l = r;
142         while (now.dot(p[l + 1] - p[i]) <= now.dot(p[l] - p[i]))
143             l = (l + 1) % n;
144         T d = now.abs2();
145         T tmp = now.cross(p[t] - p[i]) * (now.dot(p[r] - p[i]) - now.dot(p[l] - p[i])) / d;
146         ans = min(ans, tmp);
147     }
148     return p.pop_back(), ans;
149 }
150 T dis2(polygon &p1)
151 { //凸包最近距離平方
152     vector<point<T>> &P = p, &Q = p1.p;
153     int n = P.size(), m = Q.size(), l = 0, r = 0;
154     for (int i = 0; i < n; ++i)
155         if (P[i].y < P[l].y)
156             l = i;
157     for (int i = 0; i < m; ++i)
158         if (Q[i].y < Q[r].y)
159             r = i;
160     P.push_back(P[0]), Q.push_back(Q[0]);
161     T ans = 1e99;
162     for (int i = 0; i < n; ++i)
163     {
164         while ((P[l] - P[l + 1]).cross(Q[r + 1] - Q[r]) < 0)
165             r = (r + 1) % m;

```

```

166         ans = min(ans, line<T>(P[l], P[l + 1]).seg_dis2(line<T>(Q[r], Q[r + 1])));
167         l = (l + 1) % n;
168     }
169     return P.pop_back(), Q.pop_back(), ans;
170 }
171 static char sign(const point<T> &t)
172 {
173     return (t.y == 0 ? t.x : t.y) < 0;
174 }
175 static bool angle_cmp(const line<T> &A, const line<T> &B)
176 {
177     point<T> a = A.p2 - A.p1, b = B.p2 - B.p1;
178     return sign(a) < sign(b) || (sign(a) == sign(b) && a.cross(b) > 0);
179 }
180 int halfplane_intersection(vector<line<T>> &s)
181 { //半平面交
182     sort(s.begin(), s.end(), angle_cmp);
183     //線段左側為該線段半平面
184     int L, R, n = s.size();
185     vector<point<T>> px(n);
186     vector<line<T>> q(n);
187     q[L = R = 0] = s[0];
188     for (int i = 1; i < n; ++i)
189     {
190         while (L < R && s[i].ori(px[R - 1]) <= 0)
191             --R;
192         while (L < R && s[i].ori(px[L]) <= 0)
193             ++L;
194         q[++R] = s[i];
195         if (q[R].parallel(q[R - 1]))
196         {
197             --R;
198             if (q[R].ori(s[i].p1) > 0)
199                 q[R] = s[i];
200         }
201         if (L < R)
202             px[R - 1] = q[R - 1].line_intersection(q[R]);
203     }
204     while (L < R && q[L].ori(px[R - 1]) <= 0)
205         --L;
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 }

```

4.5 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 };

```

```

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

```

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            {

```

5 Graph

5.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/
2 #define inf 99999 //define by you maximum edges weight
3 vector<vector<int>> edges;
4 vector<int> dist;
5 vector<int> ancestor;
6 void BellmanFord(int start,int node){
7     dist[start] = 0;
8     for(int it = 0; it < node-1; it++){

```

```

15     if (edges[i].first == q.front()
16         && pass[edges[i].second] ==
17         0)
18     {
19         p.push(edges[i].second);
20         result[edges[i].second] =
21         count;
22     }
23     else if (edges[i].second == q.
24         front() && pass[edges[i].
25         first] == 0)
26     {
27         p.push(edges[i].first);
28         result[edges[i].first] =
29         count;
30     }
31     else
32         newedges.push_back(edges[i]);
33 }
34 edges = newedges;
35 newedges.clear();
36 q.pop();
37 if (q.empty() == true)
38 {
39     q = p;
40     queue<int> tmp;
41     p = tmp;
42     count++;
43 }
44 }
45 }
46 int main()
47 {
48     int node;
49     cin >> node;
50     vector<pair<int, int>> edges;
51     int a, b;
52     while (cin >> a >> b)
53     {
54         /*a = b = -1 means input edges ended
55         */
56         if (a == -1 && b == -1)
57             break;
58         edges.push_back(pair<int, int>(a, b));
59     }
60     vector<int> result(node, -1);
61     BFS(result, edges, node, 0);
62     return 0;
63 }

```

5.3 DFS-rec

```

1  /*DFS - Recursive version*/
2  map<pair<int,int>,int> edges;
3  vector<int> pass;
4  vector<int> route;
5  void DFS(int start){
6      pass[start] = 1;
7      map<pair<int,int>,int>::iterator iter;

```

```

8      for(iter = edges.begin(); iter != edges.
9          end(); iter++){
10         if((*iter).first.first == start &&
11             (*iter).second == 0 && pass[(*
12             iter).first.second] == 0){
13             route.push_back((*iter).first.
14             second);
15             DFS((*iter).first.second);
16         }
17     }
18     else if((*iter).first.second ==
19         start && (*iter).second == 0 &&
20         pass[(*iter).first.first] == 0){
21         route.push_back((*iter).first.
22         first);
23         DFS((*iter).first.first);
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;
6  vector<int> isDone(MAXN, false), dist,
7  ancestor;
8  void dijkstra(int s)
9  {
10     priority_queue<pii, vector<pii>, greater
11     <pii>> pq;
12     pq.push(pii(0, s));
13     ancestor[s] = -1;
14     while (!pq.empty())
15     {
16         int u = pq.top().second;
17         pq.pop();
18         isDone[u] = true;
19         for (auto &pr : weight[u])
20         {
21             int v = pr.first, w = pr.second;

```

```

23         if (!isDone[v] && dist[u] + w <
24             dist[v])
25         {
26             dist[v] = dist[u] + w;
27             pq.push(pii(dist[v], v));
28             ancestor[v] = u;
29         }
30     }
31 }
32 // weight[a - 1].push_back(pii(b - 1, w));
33 // weight[b - 1].push_back(pii(a - 1, w));
34 // dist.resize(n, inf);
35 // ancestor.resize(n, -1);
36 // dist[0] = 0;
37 // dijkstra(0);

```

5.5 Euler circuit

```

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

```

5.6 Floyd-warshall

```

1  /*SPA - Floyd-Warshall*/
2  #define inf 99999
3  void floyd_warshall(vector<vector<int>>&
4  distance, vector<vector<int>>& ancestor,
5  int n){
6      for (int k = 0; k < n; k++){
7          for (int i = 0; i < n; i++){
8              for (int j = 0; j < n; j++){
9                  if(distance[i][k] + distance
10                     [k][j] < distance[i][j])
11                  {
12                      distance[i][j] =
13                      distance[i][k] +
14                      distance[k][j];
15                      ancestor[i][j] =
16                      ancestor[k][j];
17                  }
18              }
19          }
20      }
21 }

```

```

15 int main(){
16     int n;
17     cin >> n;
18     int a, b, d;
19     vector<vector<int>> distance(n, vector<
20     int>(n,99999));
21     vector<vector<int>> ancestor(n, vector<
22     int>(n,-1));
23     while(cin>>a>>b>>d){
24         if(a == -1 && b == -1 && d == -1)
25             break;
26         distance[a][b] = d;
27         ancestor[a][b] = a;
28     }
29     for (int i = 0; i < n; i++)
30         distance[i][i] = 0;
31     floyd_warshall(distance, ancestor, n);
32     /*Negative cycle detection*/
33     for (int i = 0; i < n; i++){
34         if(distance[i][i] < 0){
35             cout << "Negative cycle!" <<
36             endl;
37             break;
38         }
39     }
40     return 0;
41 }

```

5.7 Hamilton_cycle

```

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

```

```

29 vector<vector<int>> gp(n+1,vector<
    int>(n+1,0));
30 while(cin>>a>>b){
31     if(a == 0 && b == 0)
32         break;
33     gp[a][b] = 1;
34     gp[b][a] = 1;
35 }
36 vector<int> solution(n + 1, -1);
37 vector<bool> pass(n + 1, false);
38 solution[1] = 0;
39 pass[1] = true;
40 bool flag = false;
41 hamilton(gp, 1,1 ,solution,pass,flag
    );
42 if(!flag)
43     cout << "N" << endl;
44 }
45 return 0;
46 }
47 /*
48 4
49 1 2
50 2 3
51 2 4
52 3 4
53 3 1
54 0 0
55 output: 1 3 4 2 1
56 */

```

5.8 Kruskal

```

1 /*mst - Kruskal*/
2 struct edges{
3     int from;
4     int to;
5     int weight;
6     friend bool operator < (edges a, edges b
7     ){
8         return a.weight > b.weight;
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

```

```

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

```

5.9 Prim

```

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

```

```

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

```

5.10 Union_find

```

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

```

```

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 {
5     string testCase = "";
6     for (int j = 0; j < input.size(); ++j)
7         if (i & (1 << j))
8             testCase += input[j];
9 }

```

6.3 Extended Euclidean

```

1 // ax + by = gcd(a,b)
2 pair<long long, long long> extgcd(long long
3 a, long long b)
4 {
5     if (b == 0)
6         return {1, 0};
7     long long k = a / b;
8     pair<long long, long long> p = extgcd(b,
9     a - k * b);
10    //cout << p.first << " " << p.second <<
11    endl;
12    //cout << "商數(k)= " << k << endl <<
13    endl;
14    return {p.second, p.first - k * p.second
15    };

```



```

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

```

6.4 Fermat

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

6.5 Hex to Dec

```

1 int HextoDec(string num) //16 to 10
2 {
3     int base = 1;
4     int temp = 0;
5     for (int i = num.length() - 1; i >= 0; i
6         --)
7     {
8         if (num[i] >= '0' && num[i] <= '9')
9         {
10             temp += (num[i] - 48) * base;
11             base = base * 16;
12         }
13         else if (num[i] >= 'A' && num[i] <=
14             'F')
15         {
16             temp += (num[i] - 55) * base;
17             base = base * 16;
18         }
19     }
20     return temp;
21 }
22 void DecToHex(int p) //10 to 16
23 {
24     char *l = new (char);
25     sprintf(l, "%X", p);
26     //int l_intResult = stoi(l);
27     cout << l << "\n";
28     //return l_intResult;
29 }

```

6.6 Log

```

1 double mylog(double a, double base)
2 {
3     //a 的對數底數 b = 自然對數 (a) / 自然對
4     數 (b) 。
5     return log(a) / log(base);
6 }

```

6.7 Mod

```

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

```

6.8 Mod 性質

加法： $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$
 減法： $(a - b) \bmod p = (a \bmod p - b \bmod p + p) \bmod p$
 乘法： $(a * b) \bmod p = (a \bmod p * b \bmod p) \bmod p$
 次方： $(a^b) \bmod p = ((a \bmod p)^b) \bmod p$
 加法結合律： $((a + b) \bmod p + c) \bmod p = (a + (b + c)) \bmod p$
 乘法結合律： $((a * b) \bmod p * c) \bmod p = (a * (b * c)) \bmod p$
 加法交換律： $(a + b) \bmod p = (b + a) \bmod p$
 乘法交換律： $(a * b) \bmod p = (b * a) \bmod p$
 結合律： $((a + b) \bmod p * c) = ((a * c) \bmod p + (b * c) \bmod p) \bmod p$

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

以下為性質：

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

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

- 放大縮小模數：

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

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

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

模逆元如果存在會有無限個，任意兩相鄰模逆元相差 c 。

費馬小定理

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

歐拉定理

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

Wilson's theorem

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

6.9 PI

```

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

```

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; b >>= 1)
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); u >>= 1)
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 }

```

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("%.3lf\n", R);
14 }

```

6.14 公式

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

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

6.15 四則運算

```

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

```

```

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

```

6.16 因數表

```

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

```

6.17 數字乘法組合

```

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

```

6.18 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
  &out, vector<vector<int>> &ans)
2 {
3     if (n == 0)
4     {
5         for (int i : out)
6             if (i > m)
7                 return;
8         ans.push_back(out);
9     }
10    for (int j = i; j <= n; j++)
11    {
12        out.push_back(j);
13        recur(j, n - j, m, out, ans);
14        out.pop_back();
15    }
16 }
17 vector<vector<int>> ans;
18 vector<int> zero;
19 recur(1, num, num, zero, ans);
20 // num 為 input 數字
21 for (int i = 0; i < ans.size(); i++)
22 {
23     for (int j = 0; j < ans[i].size() - 1; j++)
24         cout << ans[i][j] << " ";
25     cout << ans[i][ans[i].size() - 1] <<
26     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 *
17            prime[j] < N; ++j)
18        {
19            np[i * prime[j]] = true;
20            if (i % prime[j] == 0)
21                break;
22        }
23    }
24    return cnt;
25 }
26 const int M = 7;
27 const int PM = 2 * 3 * 5 * 7 * 11 * 13 * 17;
28 int phi[PM + 1][M + 1], sz[M + 1];
29 void init()
30 {
31     getprime();
32     sz[0] = 1;
33     for (int i = 0; i <= PM; ++i)
34         phi[i][0] = i;
35     for (int i = 1; i <= M; ++i)
36     {
37         sz[i] = prime[i] * sz[i - 1];
38         for (int j = 1; j <= PM; ++j)
39             phi[j][i] = phi[j][i - 1] - phi[
40                 j / prime[i]][i - 1];
41     }
42 }

```

```

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

```

```

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

```

7 Other

7.1 binary search 三類變化

```

1 // 查找和目標值完全相等的數
2 int find(vector<int> &nums, int target)
3 {
4     int left = 0, right = nums.size();
5     while (left < right)
6     {
7         int mid = left + (right - left) / 2;
8         if (nums[mid] == target)
9             return mid;
10        else if (nums[mid] < target)
11            left = mid + 1;
12        else
13            right = mid;
14    }
15    return -1;
16 }
17 // 找第一個不小於目標值的數 == 找最後一個小
18 // 於目標值的數
19 // *(lower_bound)*/
20 int find(vector<int> &nums, int target)
21 {
22     int left = 0, right = nums.size();
23     while (left < right)
24     {
25         int mid = left + (right - left) / 2;
26         if (nums[mid] < target)
27             left = mid + 1;
28        else
29            right = mid;
30    }
31    return right;
32 }
33 // 找第一個大於目標值的數 == 找最後一個不大
34 // 於目標值的數
35 // *(upper_bound)*/
36 int find(vector<int> &nums, int target)
37 {
38     int left = 0, right = nums.size();
39     while (left < right)
40     {
41         int mid = left + (right - left) / 2;
42         if (nums[mid] <= target)
43             left = mid + 1;
44        else
45            right = mid;
46    }
47    return right;
48 }

```

7.2 heap sort

```

1 void MaxHeapify(vector<int> &array, int root
2   , int length)
3 {
4     int left = 2 * root,
5       right = 2 * root + 1,
6       largest;
7     if (left <= length && array[left] >
8         array[root])
9         largest = left;
10    else
11        largest = root;
12    if (right <= length && array[right] >
13        array[largest])
14        largest = right;
15    if (largest != root)
16    {
17        swap(array[largest], array[root]);
18        MaxHeapify(array, largest, length);
19    }
20 }
21 void HeapSort(vector<int> &array)
22 {
23     array.insert(array.begin(), 0);
24     for (int i = (int)array.size() / 2; i >=
25         1; i--)
26         MaxHeapify(array, i, (int)array.size()
27             - 1);
28     int size = (int)array.size() - 1;
29     for (int i = (int)array.size() - 1; i >=
30         2; i--)
31     {
32         swap(array[1], array[i]);
33         size--;
34         MaxHeapify(array, 1, size);
35     }
36     array.erase(array.begin());
37 }

```

7.3 Merge sort

```

1 void Merge(vector<int> &arr, int front, int
2   mid, int end)
3 {
4     vector<int> LeftSub(arr.begin() + front,
5       arr.begin() + mid + 1);
6     vector<int> RightSub(arr.begin() + mid +
7       1, arr.begin() + end + 1);
8     LeftSub.insert(LeftSub.end(), INT_MAX);
9     RightSub.insert(RightSub.end(), INT_MAX);
10    ;
11    int idxLeft = 0, idxRight = 0;
12
13    for (int i = front; i <= end; i++)
14    {
15        if (LeftSub[idxLeft] <= RightSub[
16            idxRight])
17        {
18            arr[i] = LeftSub[idxLeft];
19            idxLeft++;
20        }
21        else
22        {
23            arr[i] = RightSub[idxRight];
24            idxRight++;
25        }
26    }

```

```

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

```

7.4 Quick

```

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

```

7.5 Weighted Job Scheduling

```

1 struct Job
2 {
3     int start, finish, profit;
4 };
5 bool jobComparataor(Job s1, Job s2)

```

```

6 {
7     return (s1.finish < s2.finish);
8 }
9 int latestNonConflict(Job arr[], int i)
10 {
11     for (int j = i - 1; j >= 0; j--)
12     {
13         if (arr[j].finish <= arr[i].start)
14             return j;
15     }
16     return -1;
17 }
18 int findMaxProfit(Job arr[], int n)
19 {
20     sort(arr, arr + n, jobComparataor);
21     int *table = new int[n];
22     table[0] = arr[0].profit;
23     for (int i = 1; i < n; i++)
24     {
25         int inclProf = arr[i].profit;
26         int l = latestNonConflict(arr, i);
27         if (l != -1)
28             inclProf += table[l];
29         table[i] = max(inclProf, table[i -
30             1]);
31     }
32     int result = table[n - 1];
33     delete[] table;
34     return result;
35 }

```

7.6 數獨解法

```

1 int getSquareIndex(int row, int column, int
2   n)
3 {
4     return row / n * n + column / n;
5 }
6 bool backtracking(vector<vector<int>> &board
7   , vector<vector<bool>> &rows, vector<
8   vector<bool>> &cols,
9   vector<vector<bool>> &boxes
10   , int index, int n)
11 {
12     int n2 = n * n;
13     int rowNum = index / n2, colNum = index
14       % n2;
15     if (index >= n2 * n2)
16         return true;
17
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         }
30     }

```

```

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

```

8 String

8.1 KMP

```

1 // 用在一個 s 內查找一個詞 w 的出現位置
2 void ComputePrefix(string s, int next[])
3 {
4     int n = s.length();
5     int q, k;
6     next[0] = 0;
7     for (k = 0, q = 1; q < n; q++)
8     {

```

```

9   while (k > 0 && s[k] != s[q])
10       k = next[k];
11   if (s[k] == s[q])
12       k++;
13   next[q] = k;
14   }
15   }
16   void KMPMatcher(string text, string pattern)
17   {
18       int n = text.length();
19       int m = pattern.length();
20       int next[pattern.length()];
21       ComputePrefix(pattern, next);
22
23       for (int i = 0, q = 0; i < n; i++)
24       {
25           while (q > 0 && pattern[q] != text[i])
26               q = next[q];
27           if (pattern[q] == text[i])
28               q++;
29           if (q == m)
30           {
31               cout << "Pattern occurs with shift " << i - m + 1 << endl;
32               ;
33               q = 0;
34           }
35       }
36       // string s = "abcdabcdebc";
37       // string p = "bcd";
38       // KMPMatcher(s, p);
39       // cout << endl;

```

8.3 Sliding window

```

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

```

8.4 Split

```

1   vector<string> mysplit(const string &str,
2       const string &delim)
3   {
4       vector<string> res;
5       if (" " == str)
6           return res;
7
8       char *strs = new char[str.length() + 1];
9       char *d = new char[delim.length() + 1];
10      strcpy(strs, str.c_str());
11      strcpy(d, delim.c_str());
12      char *p = strtok(strs, d);
13      while (p)
14      {
15          string s = p;
16          res.push_back(s);
17          p = strtok(NULL, d);
18      }
19      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       正位數
8       int s;
9       int v1, v[LEN];
10      // vector<int> v;
11      Bigint() : s(1) { v1 = 0; }
12      Bigint(long long a)
13      {
14          s = 1;
15          v1 = 0;
16          if (a < 0)
17          {
18              s = -1;
19              a = -a;
20          }
21          while (a)
22          {
23              push_back(a % BIGMOD);
24              a /= BIGMOD;
25          }
26      }
27      Bigint(string str)
28      {
29          s = 1;
30          v1 = 0;
31          int stPos = 0, num = 0;
32          if (!str.empty() && str[0] == '-')
33          {
34              stPos = 1;
35              s = -1;
36          }
37          for (int i = str.length() - 1, q = 1; i >= stPos; i--)
38          {
39              num += (str[i] - '0') * q;
40              if ((q *= 10) >= BIGMOD)
41              {
42                  push_back(num);
43                  num = 0;
44                  q = 1;
45              }
46          }
47          if (num)
48              push_back(num);
49          n();
50      }
51      int len() const
52      {
53          return v1; //return SZ(v);
54      }
55      bool empty() const { return len() == 0; }
56      void push_back(int x)
57      {
58          v[v1++] = x; //v.PB(x);
59      }
60      void pop_back()
61      {
62          v1--; //v.pop_back();
63      }

```

```

62   int back() const
63   {
64       return v[v1 - 1]; //return v.back();
65   }
66   void n()
67   {
68       while (!empty() && !back())
69           pop_back();
70   }
71   void resize(int n1)
72   {
73       v1 = n1; //v.resize(n1);
74       fill(v, v + v1, 0); //fill(ALL(v), 0);
75   }
76   void print() const
77   {
78       if (empty())
79       {
80           putchar('0');
81           return;
82       }
83       if (s == -1)
84           putchar('-');
85       printf("%d", back());
86       for (int i = len() - 2; i >= 0; i--)
87           printf("%.4d", v[i]);
88   }
89   friend std::ostream &operator<<(std::ostream &out, const Bigint &a)
90   {
91       if (a.empty())
92       {
93           out << "0";
94           return out;
95       }
96       if (a.s == -1)
97           out << "-";
98       out << a.back();
99       for (int i = a.len() - 2; i >= 0; i--)
100      {
101          char str[10];
102          sprintf(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 Bigint &b) const
145 {
146     Bigint r = (*this);
147     r.s = -r.s;
148     return r;
149 }
150 Bigint operator+(const Bigint &b) const
151 {
152     if (s == -1)
153         return -(-(*this) + (-b));
154     if (b.s == -1)
155         return (*this) - (-b);
156     Bigint r;
157     int nl = max(len(), b.len());
158     r.resize(nl + 1);
159     for (int i = 0; i < nl; i++)
160     {
161         if (i < len())
162             r.v[i] += v[i];
163         if (i < b.len())
164             r.v[i] += b.v[i];
165         if (r.v[i] >= BIGMOD)
166         {
167             r.v[i + 1] += r.v[i] /
168                 BIGMOD;
169             r.v[i] %= BIGMOD;
170         }
171     }
172     r.n();
173     return r;
174 }
175 Bigint operator-(const Bigint &b) const
176 {
177     if (s == -1)
178         return -(-(*this) - (-b));
179     if (b.s == -1)
180         return (*this) + (-b);
181     if ((*this) < b)
182         return -(-(*this));
183     Bigint r;
184     r.resize(len());
185     for (int i = 0; i < len(); i++)
186     {
187         r.v[i] += v[i];
188         if (i < b.len())
189             r.v[i] -= b.v[i];

```

```

189         if (r.v[i] < 0)
190         {
191             r.v[i] += BIGMOD;
192             r.v[i + 1]--;
193         }
194     }
195     r.n();
196     return r;
197 }
198 Bigint operator*(const Bigint &b)
199 {
200     Bigint r;
201     r.resize(len() + b.len() + 1);
202     r.s = s * b.s;
203     for (int i = 0; i < len(); i++)
204     {
205         for (int j = 0; j < b.len(); j
206             ++
207         )
208         {
209             r.v[i + j] += v[i] * b.v[j];
210             if (r.v[i + j] >= BIGMOD)
211             {
212                 r.v[i + j + 1] += r.v[i
213                     + j] / BIGMOD;
214                 r.v[i + j] %= BIGMOD;
215             }
216         }
217     }
218     r.n();
219     return r;
220 }
221 Bigint operator/(const Bigint &b)
222 {
223     Bigint r;
224     r.resize(max(1, len() - b.len() + 1)
225 );
226     int oriS = s;
227     Bigint b2 = b; // b2 = abs(b)
228     s = b2.s; r.s = 1;
229     for (int i = r.len() - 1; i >= 0; i
230         --
231     )
232     {
233         int d = 0, u = BIGMOD - 1;
234         while (d < u)
235         {
236             int m = (d + u + 1) >> 1;
237             r.v[i] = m;
238             if ((r * b2) > (*this))
239                 u = m - 1;
240             else
241                 d = m;
242         }
243     }
244     r.v[i] = d;
245 }
246 Bigint operator%(const Bigint &b)
247 {
248     return (*this) - (*this) / b * b;
249 }

```

9.2 DisjointSet

```

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

```

9.3 Matirx

```

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

```

```

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

```

```

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

```

9.4 Trie

```

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

```

```

38            g = x / BASE;
39        }
40        return c;
41    }
42 }
43
44 ostream& operator << (ostream &out, const
45     BigInteger& x){
46     out << x.s.back();
47     for(int i = x.s.size()-2; i >= 0; i--){
48         char buf[20];
49         sprintf(buf, "%08d", x.s[i]);
50         for(int j = 0; j < strlen(buf); j++){
51             out << buf[j];
52         }
53     }
54     return out;
55 }
56
57 istream& operator >> (istream &in,
58     BigInteger& x){
59     string s;
60     if(!(in >> s))
61         return in;
62     x = s;
63     return in;
64 }
65
66 struct Trie{
67     int c[5000005][10];
68     int val[5000005];
69     int sz;
70     int getIndex(char c){
71         return c - '0';
72     }
73     void init(){
74         memset(c[0], 0, sizeof(c[0]));
75         memset(val, -1, sizeof(val));
76         sz = 1;
77     }
78     void insert(BigInteger x, int v){
79         int u = 0;
80         int max_len_count = 0;
81         int firstNum = x.s.back();
82         char firstBuf[20];
83         sprintf(firstBuf, "%d", firstNum);
84         for(int j = 0; j < strlen(firstBuf);
85             j++){
86             int index = getIndex(firstBuf[j]
87                 );
88             if(!c[u][index]){
89                 memset(c[sz], 0, sizeof(c[
90                     sz]));
91                 val[sz] = v;
92                 c[u][index] = sz++;
93             }
94             u = c[u][index];
95             max_len_count++;
96         }
97         for(int i = x.s.size()-2; i >= 0; i

```

```

98         --){
99             char buf[20];
100             sprintf(buf, "%08d", x.s[i]);
101             for(int j = 0; j < strlen(buf)
102                 && max_len_count < 50; j++){
103                 int index = getIndex(buf[j])
104                     ;
105                 if(!c[u][index]){
106                     memset(c[sz], 0, sizeof
107                         (c[sz]));
108                     val[sz] = v;
109                     c[u][index] = sz++;
110                 }
111                 u = c[u][index];
112                 max_len_count++;
113             }
114             if(max_len_count >= 50){
115                 break;
116             }
117         }
118     }
119     int find(const char* s){
120         int u = 0;
121         int n = strlen(s);
122         for(int i = 0; i < n; ++i)
123         {
124             int index = getIndex(s[i]);
125             if(!c[u][index]){
126                 return -1;
127             }
128             u = c[u][index];
129         }
130         return val[u];
131     }
132 }

```

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

```

```

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

```

TO DO WRITING NOT THINKING

Contents

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