

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 pb push_back
7 #define endl '\n'
8 #define bug(x) cout << "value of x is " << x
9 << endl;
10 #define bugarr(x) \
11     for (auto i : x) \
12         cout << i << ' '; \
13         cout << endl;
14 #define x first
15 #define y second
16 int main()
17 {
18     ios::sync_with_stdio(0);
19     cin.tie(0);
20     return 0;
21 }
```

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
```

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); // 無條件捨去
3 | floor(double f); // 無條件進入
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 |
15 | /*queue*/
16 | queue<datatype> q;
17 | 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(); /*移掉最前面的值*/
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(); /*移掉最上面的值*/
```

```
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;
```

2 DP

2.1 3 維 DP 思路

```
1 | 解題思路: dp[i][j][k]
2 | i 跟 j 代表 range i ~ j 的 value
3 | k 在我的理解裡是視題目的要求而定的
4 | 像是 Remove Boxes 當中 k 代表的是在 i 之前還
   | 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[
           | i]);
9 |         for (int k = 1; num > 0; k *= 2)
10 |        {
11 |            if (k > num)
12 |                k = num;
13 |            num -= k;
14 |            for (int j = w; j >= weight[i] *
           | k; --j)
15 |                c[j] = max(c[j], c[j -
           | weight[i] * k] + cost[i]
           | * k);
```

```
16 |         }
17 |     }
18 |     cout << "Max Prince" << c[w];
19 | }
```

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(),
   | vector<int>(bagWeight + 1, 0));
7 |     for (int j = weight[0]; j <= bagWeight;
   | j++)
8 |         dp[0][j] = value[0];
9 |     // weight數組的大小就是物品個數
10 |    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<
5     int>(M + 1, 0));
6     for (int i = 1; i <= N; ++i)
7     {
8         for (int j = 1; j <= M; ++j)
9         {
10            if (Ans[i - 1] == num[j - 1])
11                LCS[i][j] = LCS[i - 1][j -
12                1] + 1;
13            else
14                LCS[i][j] = max(LCS[i - 1][j],
15                LCS[i][j - 1]);
16        }
17    }
18    cout << LCS[N][M] << '\n';
19    //列印 LCS
20    int n = N, m = M;
21    vector<string> k;
22    while (n && m)
23    {
24        if (LCS[n][m] != max(LCS[n - 1][m],
25        LCS[n][m - 1]))
26        {
27            k.push_back(Ans[n - 1]);
28            n--;
29        }
30    }
31 }

```

```

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

```

1 vector<int> ans;
2 void printLIS(vector<int> &arr, vector<int>
  &pos, int index)
3 {
4     if (pos[index] != -1)
5         printLIS(arr, pos, pos[index]);
6     // printf("%d", arr[index]);
7     ans.push_back(arr[index]);
8 }
9 void LIS(vector<int> &arr)
10 {
11     vector<int> dp(arr.size(), 1);
12     vector<int> pos(arr.size(), -1);
13     int res = INT_MIN, index = 0;
14     for (int i = 0; i < arr.size(); ++i)
15     {
16         for (int j = i + 1; j < arr.size();
17         ++j)
18         {
19             if (arr[j] > arr[i])
20             {
21                 if (dp[i] + 1 > dp[j])
22                     dp[j] = dp[i] + 1;
23                 pos[j] = i;
24             }
25         }
26         if (dp[i] > res)
27         {
28             res = dp[i];
29             index = i;
30         }
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            length
11            x++;
12        x--;
13        if (2 * x + 1 > maxlen)
14        {
15            maxlen = 2 * x + 1;
16            l = i - x;
17            r = i + x;
18        }
19        x = 0;
20        while ((s[i - x] == s[i + 1 + x]) &&
21        (i - x >= 0) && (i + 1 + x < n)) //even length
22            x++;
23        if (2 * x > maxlen)
24        {
25            maxlen = 2 * x;
26            l = i - x + 1;
27            r = i + x;
28        }
29    }
30    cout << maxlen << '\n'; // 最後長度
31    cout << l + 1 << ' ' << r + 1 << '\n';
32    //頭到尾
33 }

```

2.9 Max_subarray

```

1 /*Kadane's algorithm*/
2 int maxSubArray(vector<int> & nums) {
3     int local_max = nums[0], global_max =
4     nums[0];
5     for (int i = 1; i < nums.size(); i++) {
6         local_max = max(nums[i], local_max +
7         nums[i]);
8         global_max = max(local_max,
9         global_max);
10    }
11    return global_max;
12 }

```

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 }

```

```

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    }
50    for (int i = 1; i <= 63; ++i)
51        if (c[m] & (1 << i))
52            cout << "用" << i << "個錢幣可湊
53            得價位" << m;
54 }

```

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 to, rev, cap, flow;
6     };
7     vector<Edge> e;
8     vector<int> G[MAXN];
9     int S, T, n;
10    int d[MAXN], iter[MAXN];
11    void addEdge(int from, int to, int cap) {
12        e.push_back({to, G[to].size(), cap, 0});
13        G[from].push_back(e.size() - 1);
14        e.push_back({from, G[from].size() - 1, 0, 0});
15        G[to].push_back(e.size() - 1);
16    }
17    bool bfs() {
18        queue<int> q;
19        d[S] = 0;
20        q.push(S);
21        while (!q.empty()) {
22            int u = q.front(); q.pop();
23            for (int i = 0; i < G[u].size(); i++) {
24                Edge &e = e[G[u][i]];
25                if (e.cap > e.flow && d[e.to] > d[u] + 1) {
26                    d[e.to] = d[u] + 1;
27                    q.push(e.to);
28                }
29            }
30        }
31        return d[T] < INF;
32    }
33    int dfs(int u, int f) {
34        if (u == T) return f;
35        for (int i = iter[u]; i < G[u].size(); i++) {
36            Edge &e = e[G[u][i]];
37            if (e.cap > e.flow && d[e.to] == d[u] + 1) {
38                int f2 = dfs(e.to, min(f, e.cap - e.flow));
39                e.flow += f2;
40                e[G[e.to][e.rev]].flow -= f2;
41                return f2;
42            }
43            iter[u]++;
44        }
45        return 0;
46    }
47    int maxflow() {
48        int flow = 0;
49        while (bfs()) {
50            int f;
51            while (f = dfs(S, INF)) flow += f;
52        }
53        return flow;
54    }
55 }

```

```

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 }
17 // min cut start
18 bool side[MAXN];
19 void cut(int u) {
20     side[u] = 1;
21     for ( int i : G[u] ) {
22         if ( !side[ edges[i].v ] &&
23             edges[i].rest )
24             cut(edges[i].v);
25 }
26 }
27 // min cut end
28 int add_node(){
29     return n++;
30 }
31 void add_edge(int u, int v, long long
32     cap){
33     edges.push_back( {u, v, cap, cap} );
34     edges.push_back( {v, u, 0, 0LL} );
35     m = edges.size();
36     G[u].push_back(m-2);
37     G[v].push_back(m-1);
38 }
39 bool bfs(){
40     fill(d,d+n,-1);
41     queue<int> que;
42     que.push(s); d[s]=0;
43     while (!que.empty()){
44         int u = que.front(); que.pop();
45         for (int ei : G[u]){
46             Edge &e = edges[ei];
47             if (d[e.v] < 0 && e.rest >
48                 0){
49                 d[e.v] = d[u] + 1;
50                 que.push(e.v);
51             }
52         }
53     }
54     return d[t] >= 0;
55 }
56 long long dfs(int u, long long a){
57     if ( u == t || a == 0 ) return a;
58     long long flow = 0, f;
59     for ( int &i=cur[u]; i < (int)G[u].
60         size(); i++ ) {
61         Edge &e = edges[ G[u][i] ];
62         if ( d[u] + 1 != d[e.v] )
63             continue;
64         f = dfs(e.v, min(a, e.rest) );
65         if ( f > 0 ) {
66             e.rest -= f;
67             edges[ G[u][i]^1 ].rest += f;
68             flow += f;
69             a -= f;
70             if ( a == 0 ) break;
71         }
72     }
73     return flow;
74 }
75 }
76 }
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
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         for(int cur = t; cur != s; cur = pre[cur]
31             ){
32             residual[pre[cur]][cur] +=
33                 bottleneck[t];
34             residual[cur][pre[cur]] -=
35                 bottleneck[t];
36         }
37         ans += bottleneck[t];
38     }
39     return ans;
40 }
41 int main(){
42     int testcase = 1;
43     int n;
44     while(cin>>n){
45         if(n == 0)

```

```

38     break;
39     vector<vector<int>> capacity(n+1, vector
40         <int>(n+1, 0));
41     int s, t, c;
42     cin >> s >> t >> c;
43     int a, b, bandwidth;
44     for(int i = 0 ; i < c ; ++i){
45         cin >> a >> b >> bandwidth;
46         capacity[a][b] += bandwidth;
47         capacity[b][a] += bandwidth;
48     }
49     cout << "Network " << testcase++ << endl
50         << endl;
51     cout << "The bandwidth is " <<
52         getMaxFlow(capacity, s, t, n) << "."
53         << endl;
54     }
55     return 0;
56 }

```

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(m, -1);
31         for (int i = 0; i < n; i++){
32             vector<bool> pass(m, false);
33             if(dfs(res,i,x,y,pass))
34                 ans += 1;
35         }
36         cout << ans << endl;
37     }
38     return 0;
39 }
40 /*
41 input:
42 4 3 5 //n matching m, l links
43 0 0
44 0 2
45 1 0
46 2 1
47 3 1
48 answer is 3
49 */

```

3.5 MFlow Model

```

1 typedef long long ll;
2 struct MF
3 {
4     static const int N = 5000 + 5;
5     static const int M = 60000 + 5;
6     static const ll oo = 10000000000000LL;
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             ++head;
53         }
54         return vi[t] == tim;
55     }
56     ll dfs(int x, ll a)
57     {
58         if (x == t || a == 0)
59             return a;
60         ll flw = 0, f;
61         int &i = cur[x];
62         for (i = first[x]; i != -1; i = next[i])
63         {
64             if (dis[x] + 1 == dis[v[i]] && (
65                 f = dfs(v[i], min(a, cap[i]
66                     - flow[i]))) > 0)
67             {
68                 flow[i] += f;
69                 flow[i ^ 1] -= f;
70                 a -= f;
71                 flw += f;
72                 if (a == 0)
73                     break;

```

```

71     }
72 }
73 return flw;
74 }
75 ll MaxFlow(int s, int t)
76 {
77     this->s = s;
78     this->t = t;
79     ll flw = 0;
80     while (bfs())
81     {
82         for (int i = 1; i <= n; ++i)
83             cur[i] = 0;
84         flw += dfs(s, oo);
85     }
86     return flw;
87 }
88 };
89 // MF Net;
90 // Net.n = n;
91 // Net.Clear();
92 // a 到 b (注意從1開始!!!!)
93 // Net.Add(a, b, w, 0);
94 // Net.MaxFlow(s, d)
95 // 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    { //點對直線的鏡射 · 要先呼叫pton轉成一般式
59        point<T> R;

```

```

1 T d = a * a + b * b;
2 R.x = (b * b * p.x - a * a * p.x - 2
3     * a * b * p.y - 2 * a * c) / d;
4 R.y = (a * a * p.y - b * b * p.y - 2
5     * a * b * p.x - 2 * b * c) / d;
6 return R;
7 }
8 bool equal(const line &l) const
9 { //直線相等
10    return ori(l.p1) == 0 && ori(l.p2)
11        == 0;
12 }
13 bool parallel(const line &l) const
14 {
15     return (p1 - p2).cross(l.p1 - l.p2)
16         == 0;
17 }
18 bool cross_seg(const line &l) const
19 {
20     return (p2 - p1).cross(l.p1 - p1) *
21         (p2 - p1).cross(l.p2 - p1) <= 0;
22     //直線是否交線段
23 }
24 int line_intersect(const line &l) const
25 { //直線相交情況 · -1無限多點、1交於一
26     點、0不相交
27     return parallel(l) ? (ori(l.p1) == 0
28         ? -1 : 0) : 1;
29 }
30 int seg_intersect(const line &l) const
31 {
32     T c1 = ori(l.p1), c2 = ori(l.p2);
33     T c3 = l.ori(p1), c4 = l.ori(p2);
34     if (c1 == 0 && c2 == 0)
35     { //共線
36         bool b1 = btw(l.p1) >= 0, b2 =
37             btw(l.p2) >= 0;
38         T a3 = l.btw(p1), a4 = l.btw(p2);
39         if (b1 && b2 && a3 == 0 && a4 >=
40             0)
41             return 2;
42         if (b1 && b2 && a3 >= 0 && a4 ==
43             0)
44             return 3;
45         if (b1 && b2 && a3 >= 0 && a4 >=
46             0)
47             return 0;
48         return -1; //無限交點
49     }
50     else if (c1 * c2 <= 0 && c3 * c4 <=
51         0)
52         return 1;
53     return 0; //不相交
54 }
55 point<T> line_intersection(const line &l1
56     ) const
57 { /*直線交點*/
58     point<T> a = p2 - p1, b = l.p2 - l.
59         p1, s = l.p1 - p1;
60     //if(a.cross(b)==0)return INF;
61     return p1 + a * (s.cross(b) / a.
62         cross(b));
63 }

```

```

100 point<T> seg_intersection(const line &l)
101     const
102 { //線段交點
103     int res = seg_intersect(l);
104     if (res <= 0)
105         assert(0);
106     if (res == 2)
107         return p1;
108     if (res == 3)
109         return p2;
110     return line_intersection(l);
111 };

```

```

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

```

4.4 Polygon

4.3 Point

```

1 template <typename T>
2 struct point
3 {
4     T x, y;
5     point() {}
6     point(const T &x, const T &y) : x(x), y(y) {}
7     point operator+(const point &b) const
8     {
9         return point(x + b.x, y + b.y);
10    }
11    point operator-(const point &b) const
12    {
13        return point(x - b.x, y - b.y);
14    }
15    point operator*(const T &b) const
16    {
17        return point(x * b, y * b);
18    }
19    point operator/(const T &b) const
20    {
21        return point(x / b, y / b);
22    }
23    bool operator==(const point &b) const
24    {
25        return x == b.x && y == b.y;
26    }
27    T dot(const point &b) const
28    {
29        return x * b.x + y * b.y;
30    }
31    T cross(const point &b) const
32    {
33        return x * b.y - y * b.x;
34    }
35    point normal() const
36    { //求法向量
37        return point(-y, x);
38    }
39    T abs2() const
40    { //向量長度的平方
41        return dot(*this);
42    }
43    T rad(const point &b) const
44    { //兩向量的弧度

```

```

1 template <typename T>
2 struct polygon
3 {
4     polygon() {}
5     vector<point<T>> p; //逆時針順序
6     T area() const
7     { //面積
8         T ans = 0;
9         for (int i = p.size() - 1, j = 0; j
10             < (int)p.size(); i = j++)
11             ans += p[i].cross(p[j]);
12         return ans / 2;
13    }
14    point<T> center_of_mass() const
15    { //重心
16        T cx = 0, cy = 0, w = 0;
17        for (int i = p.size() - 1, j = 0; j
18            < (int)p.size(); i = j++)
19        {
20            T a = p[i].cross(p[j]);
21            cx += (p[i].x + p[j].x) * a;
22            cy += (p[i].y + p[j].y) * a;
23            w += a;
24        }
25        return point<T>(cx / 3 / w, cy / 3 /
26            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        {
35            if (line<T>(p[i], p[j]).
36                point_on_segment(t))
37                return -1;
38            else if ((p[i].y > t.y) != (p[j]
39                .y > t.y) &&
40                t.x < (p[j].x - p[i].x)
41                    * (t.y - p[i].y) /
42                    (p[j].y - p[i].y)
43                    + p[i].x)
44                c = !c;
45        }
46        return c;
47    }
48    char point_in_convex(const point<T> &x)
49    const

```

```

37 {
38     int l = 1, r = (int)p.size() - 2;
39     while (l <= r)
40     { //點是否在凸多邊形內，是的話回傳1
41         、在邊上回傳-1、否則回傳0
42         int mid = (l + r) / 2;
43         T a1 = (p[mid] - p[0]).cross(x -
44             p[0]);
45         T a2 = (p[mid + 1] - p[0]).cross
46             (x - p[0]);
47         if (a1 >= 0 && a2 <= 0)
48         {
49             T res = (p[mid + 1] - p[mid]
50                 ).cross(x - p[mid]);
51             return res > 0 ? 1 : (res >=
52                 0 ? -1 : 0);
53         }
54         else if (a1 < 0)
55             r = mid - 1;
56         else
57             l = mid + 1;
58     }
59     return 0;
60 }
61 vector<T> getA() const
62 { //凸包邊對x軸的夾角
63     vector<T> res; //一定是遞增的
64     for (size_t i = 0; i < p.size(); ++i
65         )
66         res.push_back((p[(i + 1) % p.
67             size()] - p[i]).getA());
68     return res;
69 }
70 bool line_intersect(const vector<T> &A,
71     const line<T> &l) const
72 { //O(logN)
73     int f1 = upper_bound(A.begin(), A.
74         end(), (l.p1 - l.p2).getA()) - A
75         .begin();
76     int f2 = upper_bound(A.begin(), A.
77         end(), (l.p2 - l.p1).getA()) - A
78         .begin();
79     return l.cross_seg(line<T>(p[f1], p[
80         f2]));
81 }
82 polygon cut(const line<T> &l) const
83 { //凸包對直線切割，得到直線l左側的凸包
84     polygon ans;
85     for (int n = p.size(), i = n - 1, j
86         = 0; j < n; i = j++)
87     {
88         if (l.ori(p[i]) >= 0)
89         {
90             ans.p.push_back(p[i]);
91             if (l.ori(p[j]) < 0)
92                 ans.p.push_back(l.
93                     line_intersection(
94                         line<T>(p[i], p[j])),
95                     );
96         }
97         else if (l.ori(p[j]) > 0)
98             ans.p.push_back(l.
99                 line_intersection(line<T>
100                     (>(p[i], p[j]))));
101     }
102 }

```

```

83     return ans;
84 }
85 static bool graham_cmp(const point<T> &a
86     , const point<T> &b)
87 { //凸包排序函數 // 起始點不同
88     // return (a.x < b.x) || (a.x == b.x
89         && a.y < b.y); //最左下角開始
90     return (a.y < b.y) || (a.y == b.y &&
91         a.x < b.x); //Y最小開始
92 }
93 void graham(vector<point<T>> &s)
94 { //凸包 Convexhull 2D
95     sort(s.begin(), s.end(), graham_cmp)
96     ;
97     p.resize(s.size() + 1);
98     int m = 0;
99     // cross >= 0 順時針，cross <= 0 逆
100     時針旋轉
101     for (size_t i = 0; i < s.size(); ++i
102         )
103     {
104         while (m >= 2 && (p[m - 1] - p[m
105             - 2]).cross(s[i] - p[m -
106                 2]) <= 0)
107             --m;
108         p[m++] = s[i];
109     }
110     for (int i = s.size() - 2, t = m +
111         1; i >= 0; --i)
112     {
113         while (m >= t && (p[m - 1] - p[m
114             - 2]).cross(s[i] - p[m -
115                 2]) <= 0)
116             --m;
117         p[m++] = s[i];
118     }
119     if (s.size() > 1) // 重複頭一次需扣
120         掉
121         --m;
122     p.resize(m);
123 }
124 T diam()
125 { //直徑
126     int n = p.size(), t = 1;
127     T ans = 0;
128     p.push_back(p[0]);
129     for (int i = 0; i < n; i++)
130     {
131         point<T> now = p[i + 1] - p[i];
132         while (now.cross(p[t + 1] - p[i]
133             ]) > now.cross(p[t] - p[i]))
134             t = (t + 1) % n;
135         ans = max(ans, (p[i] - p[t]).
136             abs2());
137     }
138     return p.pop_back(), ans;
139 }
140 T min_cover_rectangle()
141 { //最小覆蓋矩形
142     int n = p.size(), t = 1, r = 1, l;
143     if (n < 3)
144         return 0; //也可以做最小周長矩形
145     T ans = 1e99;
146     p.push_back(p[0]);

```



```

133 for (int i = 0; i < n; i++) 182
134 {
135     point<T> now = p[i + 1] - p[i]; 183
136     while (now.cross(p[t + 1] - p[i] 184
137             ] > now.cross(p[t] - p[i])) 185
138         t = (t + 1) % n; 186
139     while (now.dot(p[r + 1] - p[i]) 187
140             > now.dot(p[r] - p[i])) 188
141         r = (r + 1) % n; 189
142     if (!i) 190
143         l = r; 191
144     while (now.dot(p[l + 1] - p[i]) 192
145             <= now.dot(p[l] - p[i])) 193
146         l = (l + 1) % n; 194
147     T d = now.abs2(); 195
148     T tmp = now.cross(p[t] - p[i]) * 196
149             (now.dot(p[r] - p[i]) - now 197
150             .dot(p[l] - p[i])) / d; 198
151     ans = min(ans, tmp); 199
152 } 200
153 return p.pop_back(), ans; 201
154 }
155 T dis2(polygon &p1) 202
156 { //凸包最近距離平方 203
157     vector<point<T>> &P = p, &Q = p1.p; 204
158     int n = P.size(), m = Q.size(), l = 205
159     0, r = 0; 206
160     for (int i = 0; i < n; ++i) 207
161         if (P[i].y < P[l].y) 208
162             l = i; 209
163     for (int i = 0; i < m; ++i) 210
164         if (Q[i].y < Q[r].y) 211
165             r = i; 212
166     P.push_back(P[0]), Q.push_back(Q[0]) 213
167     ; 214
168     T ans = 1e99; 215
169     for (int i = 0; i < n; ++i) 216
170     { 217
171         while ((P[l] - P[l + 1]).cross(Q 218
172             [r + 1] - Q[r]) < 0) 219
173             r = (r + 1) % m; 220
174         ans = min(ans, line<T>(P[l], P[l 221
175             + 1]).seg_dis2(line<T>(Q[r 222
176             ], Q[r + 1]))); 223
177         l = (l + 1) % n; 224
178     } 225
179     return P.pop_back(), Q.pop_back(), 226
180     ans; 227
181 }
182 static char sign(const point<T> &t) 228
183 { 229
184     return (t.y == 0 ? t.x : t.y) < 0; 230
185 } 231
186 static bool angle_cmp(const line<T> &A, 232
187     const line<T> &B) 233
188 { 234
189     point<T> a = A.p2 - A.p1, b = B.p2 - 235
190     B.p1; 236
191     return sign(a) < sign(b) || (sign(a) 237
192         == sign(b) && a.cross(b) > 0); 238
193 } 239
194 int halfplane_intersection(vector<line<T 240
195     >> &s) 241
196 { //半平面交 242

```

```

sort(s.begin(), s.end(), angle_cmp); 20
//線段左側為該線段半平面
int L, R, n = s.size(); 21
vector<point<T>> px(n); 22
vector<line<T>> q(n); 23
q[L = R = 0] = s[0]; 24
for (int i = 1; i < n; ++i) 25
{ 26
    while (L < R && s[i].ori(px[R - 27
        1]) <= 0) 28
        --R; 29
    while (L < R && s[i].ori(px[L]) 30
        <= 0) 31
        ++L; 32
    q[++R] = s[i]; 33
    if (q[R].parallel(q[R - 1])) 34
    { 35
        --R; 36
        if (q[R].ori(s[i].p1) > 0) 37
            q[R] = s[i]; 38
    } 39
    if (L < R) 40
        px[R - 1] = q[R - 1]. 41
        line_intersection(q[R]); 42
} 43
while (L < R && q[L].ori(px[R - 1]) 44
    <= 0) 45
    --R; 46
p.clear(); 47
if (R - L <= 1) 48
    return 0; 49
px[R] = q[R].line_intersection(q[L]) 50
; 51
for (int i = L; i <= R; ++i) 52
    p.push_back(px[i]); 53
return R - L + 1; 54
}

```

4.5 Triangle

```

1 template <typename T> 2
2 struct triangle 3
3 { 4
4     point<T> a, b, c; 5
5     triangle() {} 6
6     triangle(const point<T> &a, const point< 7
7         T> &b, const point<T> &c) : a(a), b(b), 8
8         c(c) {} 9
9     T area() const 10
10 { 11
11     T t = (b - a).cross(c - a) / 2; 12
12     return t > 0 ? t : -t; 13
13 } 14
14 point<T> barycenter() const 15
15 { //重心 16
16     return (a + b + c) / 3; 17
17 } 18
18 point<T> circumcenter() const 19
19 { //外心 20
20     static line<T> u, v; 21
21     u.p1 = (a + b) / 2; 22

```

```

u.p2 = point<T>(u.p1.x - a.y + b.y, 23
    u.p1.y + a.x - b.x); 24
v.p1 = (a + c) / 2; 25
v.p2 = point<T>(v.p1.x - a.y + c.y, 26
    v.p1.y + a.x - c.x); 27
return u.line_intersection(v); 28
} 29
point<T> incenter() const 30
{ //內心 31
    T A = sqrt((b - c).abs2()), B = sqrt 32
    ((a - c).abs2()), C = sqrt((a - 33
    b).abs2()); 34
    return point<T>(A * a.x + B * b.x + 35
        C * c.x, A * a.y + B * b.y + C * 36
        c.y) / (A + B + C); 37
} 38
point<T> perpencenter() const 39
{ //垂心 40
    return barycenter() * 3 - 41
        circumcenter() * 2; 42
} 43
} 44
}; 45

```

5 Graph

5.1 Bellman-Ford

```

1 /*SPA - Bellman-Ford*/ 2
2 #define inf 99999 //define by you maximum 3
3 edges weight 4
4 vector<vector<int>> edges; 5
4 vector<int> dist; 6
5 vector<int> ancestor; 7
6 void BellmanFord(int start, int node) { 8
7     dist[start] = 0; 9
8     for (int it = 0; it < node - 1; it++) { 10
9         for (int i = 0; i < node; i++) { 11
12             for (int j = 0; j < node; j++) { 13
13                 if (edges[i][j] != -1) { 14
14                     if (dist[i] + edges[i][j] 15
15                         < dist[j]) { 16
16                         dist[j] = dist[i] + 17
17                             edges[i][j]; 18
18                         ancestor[j] = i; 19
19                     } 20
20                 } 21
21             } 22
22         } 23
23     } 24
24     for (int i = 0; i < node; i++) // 25
25         negative cycle detection 26
26         for (int j = 0; j < node; j++) 27
27             if (dist[i] + edges[i][j] < dist[ 28
29                 j]) { 30
30                 cout << "Negative cycle!" << 31
31                     endl; 32
32                 return; 33
33             } 34

```

5.2 BFS-queue

```

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

```

```

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          */
50         if (a == -1 && b == -1)
51             break;
52         edges.push_back(pair<int, int>(a, b)
53         );
54     }
55     vector<int> result(node, -1);
56     BFS(result, edges, node, 0);
57
58     return 0;
59 }

```

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)

```

```

26         break;
27         edges.insert(pair<pair<int,int>,int>
28             >(pair<int,int>(a,b),0));
29     }
30     int start;
31     cin>>start;
32     route.push_back(start);
33     DFS(start);
34     return 0;
35 }

```

5.4 Dijkstra

```

1  /*SPA - Dijkstra*/
2  const int MAXN = 1e5 + 3;
3  const int inf = INT_MAX;
4  typedef pair<int, int> pii;
5  vector<vector<pii>> weight;
6  vector<int> isDone(MAXN, false), dist,
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
20         for (auto &pr : weight[u])
21         {
22             int v = pr.first, w = pr.second;
23
24             if (!isDone[v] && dist[u] + w <
25                 dist[v])
26             {
27                 dist[v] = dist[u] + w;
28                 pq.push(pii(dist[v], v));
29                 ancestor[v] = u;
30             }
31         }
32     }
33
34     // weight[a - 1].push_back(pii(b - 1, w));
35     // weight[b - 1].push_back(pii(a - 1, w));
36     // dist.resize(n, inf);
37     // ancestor.resize(n, -1);
38     // dist[0] = 0;
39     // 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  }
22
23 int main(){
24     int n;
25     cin >> n;
26     int a, b, d;
27     vector<vector<int>> distance(n, vector<
28         int>(n,99999));
29     vector<vector<int>> ancestor(n, vector<
30         int>(n,-1));
31     while(cin>>a>>b>>d){
32         if(a == -1 && b == -1 && d == -1)
33             break;
34         distance[a][b] = d;
35         ancestor[a][b] = a;
36     }
37     for (int i = 0; i < n; i++)
38         distance[i][i] = 0;
39     floyd_warshall(distance, ancestor, n);
40     /*Negative cycle detection*/
41     for (int i = 0; i < n; i++){
42         if(distance[i][i] < 0){
43             cout << "Negative cycle!" <<
44                 endl;
45             break;
46         }
47     }
48     return 0;
49 }

```

5.7 Hamilton_cycle

```

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

```

```

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 };
11 int find(int x,vector<int>& union_set){
12     if(x != union_set[x])
13         union_set[x] = find(union_set[x],
14                             union_set);
15     return union_set[x];
16 }
17 void merge(int a,int b,vector<int>&
18            union_set){
19     int pa = find(a, union_set);
20     int pb = find(b, union_set);
21     if(pa != pb)
22         union_set[pa] = pb;
23 }
24 void kruskal(priority_queue<edges> pq,int n)
25 {
26     vector<int> union_set(n, 0);
27     for (int i = 0; i < n; i++)
28         union_set[i] = i;
29     int edge = 0;
30     int cost = 0; //evaluate cost of mst
31     while(!pq.empty() && edge < n - 1){
32         edges cur = pq.top();
33         int from = find(cur.from, union_set);
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;

```

```

53     tmp.to = b;
54     tmp.weight = d;
55     pq.push(tmp);
56 }
57 kruskal(pq, n);
58 return 0;
59 }

```

5.9 Prim

```

1 /*mst - Prim*/
2 #define inf 99999
3 struct edges{
4     int from;
5     int to;
6     int weight;
7     friend bool operator < (edges a, edges b)
8     ){
9         return a.weight > b.weight;
10    }
11 };
12 void Prim(vector<vector<int>> gp,int n,int
13           start){
14     vector<bool> pass(n,false);
15     int edge = 0;
16     int cost = 0; //evaluate cost of mst
17     priority_queue<edges> pq;
18     for (int i = 0; i < n; i++){
19         if(gp[start][i] != inf){
20             edges tmp;
21             tmp.from = start;
22             tmp.to = i;
23             tmp.weight = gp[start][i];
24             pq.push(tmp);
25         }
26     }
27     pass[start] = true;
28     while(!pq.empty() && edge < n-1){
29         edges cur = pq.top();
30         pq.pop();
31         if(!pass[cur.to]){
32             for (int i = 0; i < n; i++){
33                 if(gp[cur.to][i] != inf){
34                     edges tmp;
35                     tmp.from = cur.to;
36                     tmp.to = i;
37                     tmp.weight = gp[cur.to][i];
38                     pq.push(tmp);
39                 }
40             }
41             pass[cur.to] = true;
42             edge += 1;
43             cost += cur.weight;
44         }
45     }
46     if(edge < n-1)
47         cout << "No mst" << endl;
48     else
49         cout << cost << endl;
50 }
51 int main(){
52     int n;

```

```

51     cin >> n;
52     int a, b, d;
53     vector<vector<int>> gp(n,vector<int>(n,
54         inf));
55     while(cin>>a>>b>>d){
56         if(a == -1 && b == -1 && d == -1)
57             break;
58         if(gp[a][b] > d)
59             gp[a][b] = d;
60     }
61     Prim(gp,n,0);
62     return 0;

```

5.10 Union_find

```

1 int find(int x, vector<int> &union_set)
2 {
3     if (union_set[x] != x)
4         union_set[x] = find(union_set[x],
5                             union_set); //compress path
6     return union_set[x];
7 }
8 void merge(int x, int y, vector<int> &
9            union_set, vector<int> &rank)
10 {
11     int rx, ry;
12     rx = find(x, union_set);
13     ry = find(y, union_set);
14     if (rx == ry)
15         return;
16     /*merge by rank -> always merge small
17     tree to big tree*/
18     if (rank[rx] > rank[ry])
19         union_set[ry] = rx;
20     else
21     {
22         union_set[rx] = ry;
23         if (rank[rx] == rank[ry])
24             ++rank[ry];
25     }
26 }
27 int main()
28 {
29     int node;
30     cin >> node; //Input Node number
31     vector<int> union_set(node, 0);
32     vector<int> rank(node, 0);
33     for (int i = 0; i < node; i++)
34         union_set[i] = i;
35     int edge;
36     cin >> edge; //Input Edge number
37     for (int i = 0; i < edge; i++)
38     {
39         int a, b;
40         cin >> a >> b;
41         merge(a, b, union_set, rank);
42     }
43     /*build party*/
44     vector<vector<int>> party(node, vector<
45         int>(0));
46     for (int i = 0; i < node; i++)

```

```

43     party[find(i, union_set)].push_back(
44         i);
45 }

```

6 Mathematics

6.1 Catalan

Catalan number

- 0~19項の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    };
16 }
17 int main()
18 {
19     int a, b;
20     cin >> a >> b;
21     pair<long long, long long> xy = extgcd(a
22     , b); //(x0,y0)

```



```

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
47         cout << ans << endl;
48     }
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) return 1;
5     int x = pow_mid(a, n / 2, m);
6     long long ans = (long long)x * x % m;
7     if (n % 2 == 1) ans = ans * a % m;
8     return (int)ans;
9 }
10
11 int inv(int a, int n, int p) // n = p-2
12 {
13     long long res = 1;
14     for (; n; n >>= 1, (a *= a) %= p)
15         if (n & 1) (res *= a) %= p;
16     return res;
17 }

```

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
3 const double PI = atan2(0.0, -1.0);

```

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

```

```

35 for (int i = 0; i < t; ++i)
36 {
37     a = fmul(a, a, n);
38     if (a == 1)
39         return false;
40     if (a == n - 1)
41         return true;
42 }
43 return false;
44 }
45 bool is_prime(long long n)
46 {
47     if (n < 2)
48         return false;
49     if (n % 2 == 0)
50         return n == 2;
51     long long u = n - 1;
52     int t = 0;
53     for (; u & 1; u >= 1, ++t)
54         ;
55     for (long long i : chk)
56     {
57         if (!check(i, u, n, t))
58             return false;
59     }
60     return true;
61 }
62 // if (is_prime(int num)) // true == prime
63 // 反之亦然

```

6.12 Round(小數)

```

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

```

6.13 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else

```

```

11        R = M;
12    }
13    printf("%.3lf\n", R);
14 }

```

6.14 公式

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

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

6.15 四則運算

```

1 string s = ""; //開頭是負號要補0
2 long long int DFS(int le, int ri) // (0,
3     string final index)
4 {
5     int c = 0;
6     for (int i = ri; i >= le; i--)
7     {
8         if (s[i] == '(')
9             c++;
10        if (s[i] == '(')
11            c--;
12        if (s[i] == '+' && c == 0)
13            return DFS(le, i - 1) + DFS(i + 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
32        號
33    if (s[le] == '-' && s[ri] == '-')
34        return DFS(le + 1, ri - 1); //去除左
35        右兩邊空格
36    if (s[le] == '-')
37        return DFS(le + 1, ri); //去除左邊空
38        格
39    if (s[ri] == '-')

```

```

36    return DFS(le, ri - 1); //去除右邊空
37    格
38    long long int num = 0;
39    for (int i = le; i <= ri; i++)
40        num = num * 10 + s[i] - '0';
41    return num;

```

6.16 因數表

```

1 vector<vector<int>> arr(1000000);
2 const int limit = 10e7;
3 for (int i = 1; i <= limit; i++)
4 {
5     for (int j = i; j <= limit; j += i)
6         arr[j].pb(i); // i 為因數
7 }

```

6.17 數字乘法組合

```

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

```

6.18 數字加法組合

```

1 void recur(int i, int n, int m, vector<int>
2           &out, vector<vector<int>> &ans)
3 {

```

```

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         ++
25         cout << ans[i][j] << " ";
26     cout << ans[i][ans[i].size() - 1] <<
27     endl;
28 }

```

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 void primeFactorization(int n) // 配合質數表
2 {
3     for (int i = 0; i < (int)p.size(); ++i)
4     {
5         if (p[i] * p[i] > n)

```

```

6         break;
7         if (n % p[i])
8             continue;
9         cout << p[i] << ' ';
10        while (n % p[i] == 0)
11            n /= p[i];
12    }
13    if (n != 1)
14        cout << n << ' ';
15    cout << '\n';
16 }

```

7 Other

7.1 binary search 三類變化

```

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

```

```

43        right = mid;
44    }
45    return right;
46 }

```

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 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 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++)

```

```

10    {
11        if (LeftSub[idxLeft] <= RightSub[
12            idxRight])
13        {
14            arr[i] = LeftSub[idxLeft];
15            idxLeft++;
16        }
17        else
18        {
19            arr[i] = RightSub[idxRight];
20            idxRight++;
21        }
22    }
23 }
24 void MergeSort(vector<int> &arr, int front,
25     int end)
26 {
27     // front = 0, end = arr.size() - 1
28     if (front < end)
29     {
30         int mid = (front + end) / 2;
31         MergeSort(arr, front, mid);
32         MergeSort(arr, mid + 1, end);
33         Merge(arr, front, mid, end);
34     }
35 }

```

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 - 1]);
30     }
31     int result = table[n - 1];
32     delete[] table;
33     return result;
34 }
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     if (board[rowNum][colNum] != 0)
18         return backtracking(board, rows,
19             cols, boxes, index + 1, n);

```

```

16     for (int i = 1; i <= n2; i++)
17     {
18         if (!rows[rowNum][i] && !cols[colNum
19             ][i] && !boxes[getSquareIndex(
20                 rowNum, colNum, n)][i])
21         {
22             rows[rowNum][i] = true;
23             cols[colNum][i] = true;
24             boxes[getSquareIndex(rowNum,
25                 colNum, n)][i] = true;
26             board[rowNum][colNum] = i;
27             if (backtracking(board, rows,
28                 cols, boxes, index + 1, n))
29                 return true;
30             board[rowNum][colNum] = 0;
31             rows[rowNum][i] = false;
32             cols[colNum][i] = false;
33             boxes[getSquareIndex(rowNum,
34                 colNum, n)][i] = false;
35         }
36     }
37     return false;
38 }
39 /*用法 main*/
40 int n = sqrt(數獨邊長大小) /*e.g. 9*9 n=3*/
41 vector<vector<int>> board(n * n + 1, vector<
42     int>(n * n + 1, 0));
43 vector<vector<bool>> isRow(n * n + 1, vector<
44     bool>(n * n + 1, false));
45 vector<vector<bool>> isColumn(n * n + 1,
46     vector<bool>(n * n + 1, false));
47 vector<vector<bool>> isSquare(n * n + 1,
48     vector<bool>(n * n + 1, false));
49 for (int i = 0; i < n * n; ++i)
50 {
51     for (int j = 0; j < n * n; ++j)
52     {
53         int number;
54         cin >> number;
55         board[i][j] = number;
56         if (number == 0)
57             continue;
58         isRow[i][number] = true;
59         isColumn[j][number] = true;
60         isSquare[getSquareIndex(i, j, n)][
61             number] = true;
62     }
63 }
64 if (backtracking(board, isRow, isColumn,
65     isSquare, 0, n))
66     /*有解答*/
67 else
68     /*解答*/

```

8 String

8.1 KMP

```

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

```

```

17         dp[i - 1][j - 1]);
18     }
19     return dp[m][n];
20 }

```

8.3 Sliding window

```

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

```

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     char *strs = new char[str.length() + 1];
8     char *d = new char[delim.length() + 1];
9     strcpy(strs, str.c_str());
10    strcpy(d, delim.c_str());
11    char *p = strtok(strs, d);
12    while (p)
13    {
14        string s = p;
15        res.push_back(s);
16        p = strtok(NULL, d);
17    }
18    return res;
19 }

```

9 data structure

9.1 Bigint

```

1 //台大 //非必要請用python
2 struct Bigint
3 {
4     static const int LEN = 60; //
5     static const int BIGMOD = 10000; //10為
6     正常位數
7     int s;
8     int vl, v[LEN];
9     // vector<int> v;
10    Bigint() : s(1) { vl = 0; }
11    Bigint(long long a)
12    {
13        s = 1;
14        vl = 0;
15        if (a < 0)
16        {
17            s = -1;
18            a = -a;
19        }
20        while (a)
21        {
22            push_back(a % BIGMOD);
23            a /= BIGMOD;
24        }
25        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 =
36                1; i >= stPos; i--)
37            {
38                num += (str[i] - '0') * q;
39                if ((q *= 10) >= BIGMOD)
40                {
41                    push_back(num);
42                    num = 0;
43                    q = 1;
44                }
45            }
46            if (num)
47                push_back(num);
48            n();
49        }
50        int len() const
51        {
52            return vl; //return SZ(v);
53        }
54        bool empty() const { return len() == 0; }
55        void push_back(int x)
56        {

```

```

56        v[vl++] = x; //v.PB(x);
57    }
58    void pop_back()
59    {
60        vl--; //v.pop_back();
61    }
62    int back() const
63    {
64        return v[vl - 1]; //return v.back();
65    }
66    void n()
67    {
68        while (!empty() && !back())
69            pop_back();
70    }
71    void resize(int nl)
72    {
73        vl = nl; //v.resize(nl);
74        fill(v, v + vl, 0); //fill(ALL(v),
75            0);
76    }
77    void print() const
78    {
79        if (empty())
80        {
81            putchar('0');
82            return;
83        }
84        if (s == -1)
85            putchar('-');
86        printf("%d", back());
87        for (int i = len() - 2; i >= 0; i--)
88            printf("%.4d", v[i]);
89    }
90    friend std::ostream &operator<<(std::
91        ostream &out, const Bigint &a)
92    {
93        if (a.empty())
94        {
95            out << "0";
96            return out;
97        }
98        if (a.s == -1)
99            out << "-";
100        out << a.back();
101        for (int i = a.len() - 2; i >= 0; i--)
102        {
103            char str[10];
104            snprintf(str, 5, "%.4d", a.v[i]);
105            out << str;
106        }
107        return out;
108    }
109    int cp3(const Bigint &b) const
110    {
111        if (s != b.s)
112            return s - b.s;
113        if (s == -1)
114            return -(*this).cp3(-b);
115        if (len() != b.len())
116            return len() - b.len(); //int
117        for (int i = len() - 1; i >= 0; i--)
118            if (v[i] != b.v[i])
119                return v[i] - b.v[i];
120    }
121    bool operator<(const Bigint &b) const
122    {
123        return cp3(b) < 0;
124    }
125    bool operator<=(const Bigint &b) const
126    {
127        return cp3(b) <= 0;
128    }
129    bool operator==(const Bigint &b) const
130    {
131        return cp3(b) == 0;
132    }
133    bool operator!=(const Bigint &b) const
134    {
135        return cp3(b) != 0;
136    }
137    bool operator>(const Bigint &b) const
138    {
139        return cp3(b) > 0;
140    }
141    bool operator>=(const Bigint &b) const
142    {
143        return cp3(b) >= 0;
144    }
145    Bigint operator-() const
146    {
147        Bigint r = (*this);
148        r.s = -r.s;
149        return r;
150    }
151    Bigint operator+(const Bigint &b) const
152    {
153        if (s == -1)
154            return -(*this) + (-b);
155        if (b.s == -1)
156            return (*this) - (-b);
157        Bigint r;
158        int nl = max(len(), b.len());
159        r.resize(nl + 1);
160        for (int i = 0; i < nl; i++)
161        {
162            if (i < len())
163                r.v[i] += v[i];
164            if (i < b.len())
165                r.v[i] += b.v[i];
166            if (r.v[i] >= BIGMOD)
167            {
168                r.v[i + 1] += r.v[i] /
169                    BIGMOD;
170                r.v[i] %= BIGMOD;
171            }
172        }
173        r.n();
174        return r;
175    }
176    Bigint operator-(const Bigint &b) const
177    {
178        if (s == -1)
179            return -(*this) - (-b);
180        if (b.s == -1)
181            return (*this) + (-b);
182        if ((*this) < b)
183            return -(b - (*this));
184        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
211                        + j] / BIGMOD;
212                    r.v[i + j] %= BIGMOD;
213                }
214            }
215        }
216        r.n();
217        return r;
218    }
219    Bigint operator/(const Bigint &b)
220    {
221        Bigint r;
222        r.resize(max(1, len() - b.len() + 1)
223            );
224        int oriS = s;
225        Bigint b2 = b; // b2 = abs(b)
226        s = b2.s = r.s = 1;
227        for (int i = r.len() - 1; i >= 0; i
228            --){
229            int d = 0, u = BIGMOD - 1;
230            while (d < u)
231            {
232                int m = (d + u + 1) >> 1;
233                r.v[i] = m;
234                if ((r * b2) > (*this))
235                    u = m - 1;
236                else
237                    d = m;
238            }
239            r.v[i] = d;
240        }
241        s = oriS;
242        r.s = s * b.s;
243        r.n();
244        return r;
245    }
246    Bigint operator%(const Bigint &b)

```


9.2 Matirx

```

245 {
246     return (*this) - (*this) / b * b;
247 }
248 };

template <typename T>
struct Matrix
{
    using rt = std::vector<T>;
    using mt = std::vector<rt>;
    using matrix = Matrix<T>;
    int r, c; // [r][c]
    mt m;
    Matrix(int r, int c) : r(r), c(c), m(r,
        rt(c)) {}
    Matrix(mt a) { m = a, r = a.size(), c =
        a[0].size(); }
    rt &operator[](int i) { return m[i]; }
    matrix operator+(const matrix &a)
    {
        matrix rev(r, c);
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                rev[i][j] = m[i][j] + a.m[i][j];
        return rev;
    }
    matrix operator-(const matrix &a)
    {
        matrix rev(r, c);
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < c; ++j)
                rev[i][j] = m[i][j] - a.m[i][j];
        return rev;
    }
    matrix operator*(const matrix &a)
    {
        matrix rev(r, a.c);
        matrix tmp(a.c, a.r);
        for (int i = 0; i < a.r; ++i)
            for (int j = 0; j < a.c; ++j)
                tmp[j][i] = a.m[i][j];
        for (int i = 0; i < r; ++i)
            for (int j = 0; j < a.c; ++j)
                for (int k = 0; k < c; ++k)
                    rev.m[i][j] += m[i][k] *
                        tmp[j][k];
        return rev;
    }
    bool inverse() //逆矩陣判斷
    {
        Matrix t(r, r + c);
        for (int y = 0; y < r; y++)
        {
            t.m[y][c + y] = 1;
            for (int x = 0; x < c; ++x)
                t.m[y][x] = m[y][x];
        }
        if (!t.gas())
            return false;
    }
};

```

```

52     for (int y = 0; y < r; y++)
53         for (int x = 0; x < c; ++x)
54             m[y][x] = t.m[y][c + x] / t.
                    m[y][y];
55     return true;
56 }
57 T gas() //行列式
58 {
59     vector<T> lazy(r, 1);
60     bool sign = false;
61     for (int i = 0; i < r; ++i)
62     {
63         if (m[i][i] == 0)
64         {
65             int j = i + 1;
66             while (j < r && !m[j][i])
67                 j++;
68             if (j == r)
69                 continue;
70             m[i].swap(m[j]);
71             sign = !sign;
72         }
73         for (int j = 0; j < r; ++j)
74         {
75             if (i == j)
76                 continue;
77             lazy[j] = lazy[j] * m[i][i];
78             T mx = m[j][i];
79             for (int k = 0; k < c; ++k)
80                 m[j][k] = m[j][k] * m[i][i] - m[i][k] * mx;
81         }
82     }
83     T det = sign ? -1 : 1;
84     for (int i = 0; i < r; ++i)
85     {
86         det = det * m[i][i];
87         det = det / lazy[i];
88         for (auto &j : m[i])
89             j /= lazy[i];
90     }
91     return det;
92 }
93 };

```

9.3 Trie

```

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 }

```

```

16 }
17 BigInteger operator = (const string& str
18 ){
19     s.clear();
20     int x, len = (str.length() - 1) /
        WIDTH + 1;
21     for(int i = 0; i < len; i++){
22         int end = str.length() - i*WIDTH;
23         int start = max(0, end-WIDTH);
24         sscanf(str.substr(start, end-
        start).c_str(), "%d", &x);
25         s.push_back(x);
26     }
27     return *this;
28 }
29 BigInteger operator + (const BigInteger&
30 b) const{
31     BigInteger c;
32     c.s.clear();
33     for(int i = 0, g = 0; i < s.size(); i++){
34         if(g == 0 && i >= s.size() && i
        >= b.s.size()) break;
35         int x = g;
36         if(i < s.size()) x+=s[i];
37         if(i < b.s.size()) x+=b.s[i];
38         c.s.push_back(x % BASE);
39         g = x / BASE;
40     }
41     return c;
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 istream& operator >> (istream &in,
57 BigInteger& x){
58     string s;
59     if(!(in >> s))
60         return in;
61     x = s;
62     return in;
63 }
64 struct Trie{
65     int c[5000005][10];
66     int val[5000005];
67     int sz;
68     int getIndex(char c){
69         return c - '0';
70     }
71     void init(){
72         memset(c[0], 0, sizeof(c[0]));
73         memset(val, -1, sizeof(val));
74     }
75 }

```

```

74     sz = 1;
75 }
76 void insert(BigInteger x, int v){
77     int u = 0;
78     int max_len_count = 0;
79     int firstNum = x.s.back();
80     char firstBuf[20];
81     sprintf(firstBuf, "%d", firstNum);
82     for(int j = 0; j < strlen(firstBuf);
83         j++){
84         int index = getIndex(firstBuf[j]);
85         if(!c[u][index]){
86             memset(c[sz], 0, sizeof(c[
87                 sz]));
88             val[sz] = v;
89             c[u][index] = sz++;
90         }
91         u = c[u][index];
92         max_len_count++;
93     }
94     for(int i = x.s.size()-2; i >= 0; i
95         --){
96         char buf[20];
97         sprintf(buf, "%08d", x.s[i]);
98         for(int j = 0; j < strlen(buf)
99             && max_len_count < 50; j++){
100             int index = getIndex(buf[j]);
101             if(!c[u][index]){
102                 memset(c[sz], 0, sizeof
103                     (c[sz]));
104                 val[sz] = v;
105                 c[u][index] = sz++;
106             }
107             u = c[u][index];
108             max_len_count++;
109         }
110         if(max_len_count >= 50){
111             break;
112         }
113     }
114 }
115 int find(const char* s){
116     int u = 0;
117     int n = strlen(s);
118     for(int i = 0; i < n; ++i)
119     {
120         int index = getIndex(s[i]);
121         if(!c[u][index]){
122             return -1;
123         }
124         u = c[u][index];
125     }
126     return val[u];
127 }

```

9.4 分數

```

1 typedef long long ll;
2 struct fraction
3 {

```

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

TO DO WRITING NOT THINKING

Contents

1 Basic	1	3 Flow & matching	2	6 Mathematics	8	7 Other	11
1.1 Code Template	1	3.1 Dinic	2	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	11
1.3 IO_fast	1	3.3 hungarian	3	6.3 Extended Euclidean	8	7.3 Merge sort	11
1.4 Python	1	3.4 Maximum_matching	3	6.4 Fermat	9	7.4 Quick	11
1.5 Range data	1	3.5 MFlow Model	3	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
2 DP	1	4.2 Line	4	6.8 Mod 性質	9	8.1 KMP	12
2.1 3 維 DP 思路	1	4.3 Point	5	6.9 PI	9	8.2 Min Edit Distance	12
2.2 Knapsack Bounded	1	4.4 Polygon	5	6.10 Prime table	9	8.3 Sliding window	12
2.3 Knapsack sample	1	4.5 Triangle	6	6.11 Prime 判斷	10	8.4 Split	12
2.4 Knapsack Unbounded	1	5 Graph	6	6.12 Round(小數)	10	9 data structure	13
2.5 LCIS	2	5.1 Bellman-Ford	6	6.13 二分逼近法	10	9.1 Bigint	13
		5.2 BFS-queue	6	6.14 公式	10	9.2 Matirx	14
		5.3 DFS-rec	7	6.15 四則運算	10	9.3 Trie	14
		5.4 Dijkstra	7	6.16 因數表	10	9.4 分數	14
		2.6 LCS	2				
		2.7 LIS	2	5.5 Euler circuit	7	6.17 數字乘法組合	10
		2.8 LPS	2	5.6 Floyd-warshall	7	6.18 數字加法組合	10
		2.9 Max_subarray	2	5.7 Hamilton_cycle	7	6.19 羅馬數字	11
		2.10 Money problem	2	5.8 Kruskal	8	6.20 質因數分解	11
				5.9 Prim	8		
				5.10 Union_find	8		