Contents

1 語法

1.1 c++

```cpp
// c++ code
std::ios::sync_with_stdio(false); // 加速

#include <bits/stdc++.h>
lower_bound(a, a + n, k);      //最左邊 ≥ k 的位置
upper_bound(a, a + n, k);      //最左邊 > k 的位置
upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
[lower_bound, upper_bound) //等於 k 的範圍
equal_range(a, a+n, k);

// 從小到大
priority_queue<int, vector<int>, greater<int>>pq

insert(it,x)//向 vector 的任意迭代器 it 處插入一個元素 x
erase(it)//刪除迭代器為 it 處的元素，erase(first,last)
//刪除一個區間[first,last)內的所有元素，時間複雜度均為 O(N)

set
insert(x) //將 x 插入 set 中   O(log(n))
count(x)  //回傳 x 是否存在於 set 中()   O(log(n))
erase(x)  //刪除在 set 中的 x   O(log(n))
clear() //刪除 set 中所有元素  O(n)
empty() //回傳是否為空   O(1)
size()  //回傳共有幾個元素   O(1)

map
insert(x) //將 x 這個 pair 插入 map 中   O(log(n))
count(x)  //回傳 x 這個 key 是否在 map 中  O(log(n))
erase(x)  //刪除在 map 中 key 為 x 的  O(log(n))

double cnt = 3.5555;
cout << fixed << setprecision(3) << cnt ;

#include <bits/stdc++.h>
using namespace std;

int main(){
  set<int>s;
  for(int i = 0; i < 10; i++){
    s.insert(i);
  }
```

```cpp
  cout << "lower bound: " << *s.lower_bound(5) <<
       '\n';// 5
  cout << "upper bound: " << *s.upper_bound(5) <<
       '\n';// 6

  if(s.lower_bound(20) == s.end()){
    cout << "all elements are less than 20\n";
  }
}
```

1.2 pythontmp

```python
sorted((4,1,9,6),reverse=True)
fruits = ['apple', 'watermelon', 'pear', 'banana']
a = sorted(fruits, key = lambda x : len(x))
print(a)
# 輸出：['pear', 'apple', 'banana', 'watermelon']
divmod(a,b)
把除數和餘數運算結果結合起來，
返回一個包含商和餘數的元組(a // b, a % b)

pow(base, exp[, mod])
>>> pow(38, -1, mod=97)
23
>>> 23 * 38 % 97 == 1
True

eof 寫法
try:
  while True:
    s = input()
except EOFError:
  pass

eval(expression, globals=None, locals=None)


list(map(int, input().split()))
 L.append(r)
my_list = ['This', 'is', 'a', 'string', 'in',
          'Python']
my_string = " ".join(my_list)
#This is a string in Python
test = [[0 for j in range(m)] for i in range(n)]
```

2 Graph

2.1 Bellman-Ford

```cpp
#include<iostream>
using namespace std;
const int INF = 1e9;
const int MAXN = 1000;
const int MAXM = 1000;
struct Edge {
    int u;
    int v;
    int w;
};

int n, m;
Edge edges[MAXM];
int dis[MAXN];

// s 是起點
bool bellman(int s) {
    for (int i = 0; i < n; i++) {
        dis[i] = INF;
    }
    dis[s] = 0;
    bool relax;
```

```
23    // 做 n 輪
24    for (int i = 0; i < n; i++) {
25        relax = false;
26        for (int j = 0; j < m; j++) {
27            int u = edges[j].u;
28            int v = edges[j].v;
29            int w = edges[j].w;
30            if (dis[u] == INF) {
31                continue;
32            }
33            if (dis[v] > dis[u] + w) {
34                dis[v] = dis[u] + w;
35                relax = true;
36            }
37        }
38        if (!relax) {
39            break;
40        }
41    }
42    return relax;
43 }
44
45
46 int main(){
47
48 }
```

2.2 Dijkstra

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define M 100005
4  #define INF 1e9
5  struct Edge{
6      int v, w;
7      Edge(int a, int b):v(a), w(b){};
8  };
9  struct node{
10     int u, dis;
11     node(){};
12     node(int a, int b):u(a), dis(b){};
13     bool operator<(const node &r)const{
14         return dis > r.dis;
15     }
16 };
17 int dis[M]; //距離
18 vector<Edge> G[M];
19 void init(){
20     fill(dis, dis+M, INF);
21     for(int i = 0; i < M; i++){
22         G[i].clear();
23     }
24 }
25 void dijkstra(int start){
26     dis[start] = 0;
27     priority_queue<node> pq;
28     pq.push(node(start, 0));
29     while(!pq.empty()){
30         node now = pq.top();
31         pq.pop();
32         if(now.dis > dis[now.u]) continue;
33         for(Edge i : G[now.u]){
34             if(dis[i.v] > now.dis + i.w){
35                 dis[i.v] = now.dis + i.w;
36                 pq.push(node(i.v, dis[i.v]));
37                 // printf("push(%d, %d)\n", i.v,
                        dis[i.v]);
38             }
39         }
40     }
41 }
42
43 int main(){
44   int point, side;
45     cin >> point >> side;
```

```
46    init();
47    for(int i = 0; i < side; i++){
48        int s, t, w;
49        cin >> s >> t >> w;
50        G[s].push_back(Edge(t, w));
51        G[t].push_back(Edge(s, w));
52    }
53    dijkstra(1);
54    for(int i = 2; i <= point; i++){
55        cout << dis[i] << '\n';
56    }
57
58 }
```

2.3 Floyd-Warshall

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define M 1005
4  #define INF 1e9
5
6  int dis[M][M];
7  // int G[M][M];
8  void init(int n){
9      for(int i = 0; i <= n; i++){
10         for(int j = 0; j <= n; j++){
11             dis[i][j] = INF;
12             if(i == j) dis[i][j] = 0;
13         }
14     }
15 }
16 void Floyd(int n){
17     for(int k = 1; k <= n; k++){
18         for(int i = 1; i <= n; i++){
19             for(int j = 1; j <= i; j++){
20                 dis[i][j]= dis[j][i] =
                        min(dis[i][k]+dis[k][j],
                        dis[i][j]);
21             }
22         }
23     }
24 }
25 void printarr(int r, int c){
26     for(int i = 1; i <= r; i++){
27         for(int j = 1; j <= c; j++){
28             if(dis[i][j] == INF) cout << "INF ";
29             else cout << dis[i][j] << ' ';
30         }
31         cout << '\n';
32     }
33 }
34 int main(){
35   int point, side;
36     cin >> point >> side;
37     init(point);
38     for(int i = 0; i < side; i++){
39         int s, t, w;
40         cin >> s >> t >> w;
41         dis[s][t] = w;
42         dis[t][s] = w;
43     }
44     Floyd(point);
45     int Cas;
46     cin >> Cas;
47     while(Cas--){
48         int i, j;
49         cin >> i >> j;
50         cout << dis[i][j] << '\n';
51     }
52     // printarr(point, point);
53
54 }
```

2.4 SPFA

```cpp
const int INF = 1e9;
const int MAXN = 1000;
struct Edge {
    int v;
    int w;
};
int n, m;
vector<Edge> G[MAXN];    //向量記圖
int dis[MAXN];
void SPFA(int s) {
    // 記錄目前的點是否在 queue 中
    bool inq[n];
    for (int i = 0; i < n; i++) {
        dis[i] = INF;
        inq[i] = false;
    }
    dis[s] = 0;
    inq[s] = true;
    queue<int> q;
    q.push(s);
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        inq[u] = false;
        for (Edge e : G[u]) {
            if (dis[e.v] > dis[u] + e.w) {
                dis[e.v] = dis[u] + e.w;
                if (!inq[e.v]) {
                    inq[e.v] = true;
                    q.push(e.v);
                }
            }
        }
    }
}

/*
Bellmam Ford / SPFA 偵測負環

如果有一個點被放到 queue 裡面超過V次,那麼有負環
最大負環為包含所有點的環,共有V條邊,被更新V次
,在極端的例子,被長度為1.2.3..V的路徑都
被更新一次最短距離。

比較
Floyd: ,需要計算許多點對的距離。
Dijkstra:沒有負邊且起點固定。
Bellmam Ford / SPFA:其他狀況。
*/
```

2.5 smallTree

```cpp
#include<bits/stdc++.h>
using namespace std;
#define M 100005
int tree[M] = {};    //parents
int r[M] = {};

struct Edge{
    int s, t, w;
    bool operator<(const Edge& r)const{
        return w < r.w;
    }
};

vector<Edge> G;

void init(int n){
    for(int i = 0; i <= n; i++){
        tree[i] = i;
        r[i] = 1;
    }
```

```cpp
}
int Find(int n){
    if(tree[n] == n) return n;    //find root
    return tree[n] = Find(tree[n]);
}

void Union(int a, int b){
    a = Find(a);
    b = Find(b);
    if (a == b) return;
    if (r[a] <= r[b]){
        tree[a] = b;    //a接b
        r[b]+=r[a];
    }
    else{
        tree[b] = a;    //b接a
        r[a] += r[b];
    }
}

int kruskal(){
    int cost = 0, flag = 0, Space = 0;
    for (auto it : G){
        it.s = Find(it.s);
        it.t = Find(it.t);
        if (it.s == it.t){
            if(Space) cout << ' ';
            Space = 1;
            flag = 1;
            cout << it.w;
            continue;
        }
        cost += it.w;
        Union(it.s, it.t);
    }
    return flag;
}
int main(){
    int point, side, Max = 0;
    while(cin >> point >> side){
        G.clear();
        if(point+side == 0) break;
        init(point);
        for(int i = 0; i < side; i++){
            Edge tmp;
            cin >> tmp.s >> tmp.t >> tmp.w;
            G.push_back(tmp);
        }
        sort(G.begin(), G.end());
        if(!kruskal()){
            cout << "forest";
        }
        cout << '\n';
    }
}
```

3 Other

3.1 KM

```cpp
// uva12083
#include<bits/stdc++.h>
using namespace std;

const int M = 500+5;
struct people{
    int high;
    char sex;
    string music, sport;
};

vector<int> G[M];
people Class[M];
int used[M] = {0};
```

Left column:

```
15  int Last[M] = {0};
16
17  bool Check(people a, people b){
18      if(abs(a.high-b.high) > 40) return true;
19      if(a.sex == b.sex) return true;
20      if(a.music != b.music) return true;
21      if(a.sport == b.sport) return true;
22      return false;
23  }
24
25  bool KM(int x){
26      for(int i = 0; i < G[x].size(); i++){
27          int v = G[x][i];
28          if(used[v]) continue;
29          used[v] = 1;
30          if(Last[v] == -1 || KM(Last[v])){
                //v找到還沒配對的人或前一個v配對的人找到別人
31              Last[v] = x;
32              return true;
33          }
34      }
35      return false;
36  }
37
38  int Ans(int n){
39      int Max = 0;
40      memset(Last, -1, sizeof(Last));
41      for(int i = 0; i < n; i++){
42          memset(used, 0, sizeof(used));
43          if(KM(i)){
44              Max++;
45          }
46      }
47      return Max;
48  }
49
50  int main(){
51      int Cas;
52      cin >> Cas;
53      while(Cas--){
54          int n;
55          cin >> n;
56          for(int i = 0; i < n; i++){
57              G[i].clear();
58              cin >> Class[i].high >> Class[i].sex >>
                    Class[i].music >> Class[i].sport;
59          }
60          for(int i = 0; i < n; i++){
61              if(Class[i].sex == 'M') continue;
62              for(int j = 0; j < n; j++){
63                  if(i == j) continue;
64                  if(!Check(Class[i], Class[j])){
65                      G[i].push_back(j);
66                  }
67              }
68          }
69          int MaxPeople = n-Ans(n);
70          cout << MaxPeople << '\n';
71      }
72
73  }
```

3.2 LCS

```
1   int n1 = s1.size(), n2 = s2.size();
2       int dp[N][N] = {};
3       for (int i = 1; i <= n1; ++i)
4       {
5           for (int j = 1; j <= n2; ++j)
6           {
7               if (s1[i - 1] == s2[j - 1])
8                   dp[i][j] = dp[i - 1][j - 1] + 1;
9               else
10                  dp[i][j] = max(dp[i - 1][j], dp[i][j
                        - 1]);
```

Right column:

```
11          }
12
13      }
14  }
15
16
17  #include<bits/stdc++.h>
18  using namespace std;
19
20  int dp[1005][1005] = {0};
21
22  int main(){
23    string a, b;
24      while(getline(cin, a) && getline(cin, b)){
25          memset(dp, 0, sizeof(dp));
26          int asize = a.size(), bsize = b.size();
27          for(int i = 1; i <= asize; i++){
28              for(int j = 1; j <= bsize; j++){
29                  if(a[i-1] == b[j-1]){
30                      dp[i][j] = dp[i-1][j-1] + 1;
31                  }
32                  else dp[i][j] = max(dp[i-1][j],
                        dp[i][j-1]);
33              }
34          }
35          cout << dp[asize][bsize] << '\n';
36      }
37
38  }
39
40
41  int n1 = s1.size(), n2 = s2.size();
42  int dp[2][N] = {};
43  for (int i = 1; i <= n1; i++)
44  {
45      int cur = i % 2;
46      int old = 1 - cur;
47      for (int j = 1; j <= n2; ++j)
48      {
49          if (s1[i - 1] == s2[j - 1])
50              dp[cur][j] = dp[old][j - 1] + 1;
51          else
52              dp[cur][j] = max(dp[old][j], dp[cur][j -
                    1]);
53      }
54  }
55  }
56
```

3.3 LIS

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   // 前後兩次LIS
4   int main(){
5     int n;
6       while(cin >> n){
7           int arr[10005] = {0};
8           int dp[10005] = {0};
9           int dp2[10005] = {0};
10          int Max = -1;
11          for(int i = 0; i < n; i++){
12              cin >> arr[i];
13          }
14          for(int i = 0; i < n; i++){
15              dp[i] = 1;
16              for(int j = 0; j < i; j++){
17                  if(arr[i] > arr[j]){
18                      dp[i] = max(dp[i], dp[j]+1);
19                  }
20              }
21          }
22          for(int i = n-1; i >= 0; i--){
23              dp2[i] = 1;
24              for(int j = n-1; j > i; j--){
```

```
25            if(arr[i] > arr[j]){
26                dp2[i] = max(dp2[i], dp2[j]+1);
27            }
28        }
29    }
30    int lds = 0, lis = 0;
31    for(int i = 0; i < n; i++){
32        Max = max(Max, min(dp[i],dp2[i]));
33    }
34    cout << 2*Max-1 << '\n';
35    }
36
37 }
38
39 void LDS(vector<int> &s){
40    if(s.size() == 0) return;
41    vector<int> v;
42    v.emplace_back(s[0]);
43    revseq[0] = 1;
44    for(int i = 1; i < s.size(); ++i){
45        int n = s[i];
46        if(n > v.back())
47            v.push_back(n);
48        else
49            *lower_bound(v.begin(), v.end(), n) = n;
50        revseq[i] = v.size();
51    }
52    return;
53 }
```

3.4 merge

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define M 100010
5 // int cnt = 0;
6 void printarr(int arr[],int l,int r){
7    for(int i=l;i<=r;i++){
8        printf(" %d",arr[i]);
9    }
10    puts("");
11 }
12
13 int merge(int arr[], int l, int r, int mid){
14    int L = l, R = mid+1;
15    int tmplen = r-l+1, tmpi = 0;
16    int tmp[M]={0};
17   int cnt = 0;
18   while(L <= mid && R <= r){
19        if(arr[L]<=arr[R]){
20            tmp[tmpi]=arr[L];
21            L++;
22        }
23        else{
24            tmp[tmpi]=arr[R];
25     cnt += mid-L+1;
26            R++;
27        }
28        tmpi++;
29    }
30    if(L>mid){
31        while(R<=r){
32            tmp[tmpi]=arr[R];
33            R++;
34            tmpi++;
35        }
36    }
37    else{
38        while(L<=mid){
39            tmp[tmpi]=arr[L];
40            L++;
41            tmpi++;
42        }
43    }
```

```
44    //L>mid&&R>r才可以全部跑過
45    L=l;
46    for (tmpi=0; tmpi<tmplen; tmpi++) {
47        arr[L] = tmp[tmpi];
48        L++;
49    }
50
51   // printf("%d %d %d:",l,mid,r);
52    // printarr(arr,l,r);
53    return cnt;
54 }
55
56 int mergeSort(int arr[],int l,int r){
57    if(r <= l) return 0;
58    int mid=(l+r)/2;
59    int cnt = 0;
60    cnt += mergeSort(arr, l, mid);
61    cnt += mergeSort(arr, mid+1, r);
62    cnt += merge(arr, l, r, mid);
63    return cnt;
64 }
65
66 int main(){
67    int n;
68    while(cin >> n){
69        if(n == 0) break;
70        int arr[M] = {0};
71        for(int i = 0; i < n; i++){
72            cin >> arr[i];
73        }
74        if(mergeSort(arr, 0, n-1)%2) cout << "Marcelo\n";
75        else cout << "Carlos\n";
76    }
77
78 }
```

3.5 Prime

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define M 10000
4 #define sq int(sqrt(double(M+5)))
5 bool prime[sq];
6 int main(){
7    memset(prime, true, sizeof(prime));
8    prime[0] = prime[1] = false;
9    for(int i = 2; i <sq; i++){
10        if(prime[i]){
11            for(int j = i*i; j < sq; j+=i){
12                prime[j] = false;
13            }
14        }
15    }
16 }
```

3.6 UVA12321

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 struct node{
4    int l, r;
5    node(){};
6    node(int l, int r):l(l), r(r){};
7    bool operator<(cnost node &a)const{
8        return l < a.l;
9    }
10 }
11
12 node gas[100005];
13 int main(){
14    int L, G;
15    while(cin >> L >> G){
16        if(L == 0 && G == 0) break;
```

```
17          for(int i = 0; i < G; i++){
18              int a, b;
19              cin >> a >> b;
20              gas[i].l = a-b;
21              gas[i].r = a+b;
22          }
23          sort(gas, gas+G);
24          int ans = G, lcover = 0, rcover = 0,i = 0;
25          while(L > lcover){
26              rcover = lcover;
27              for(; i < G && gas[i].l <= lcover; i++){
28                  if(gas[i].r > rcover) rcover =
                            gas[i].r;
29              }
30              if(lcover == rcover) break;
31              lcover = rcover;
32              ans--;
33          }
34          if(lcover < L) cout << "-1\n";
35          else cout << ans << '\n';
36      }
37  }
38  // 天然氣
```

3.7 Fire

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define M 1005
5
6  int arr[M][M] = {0};
7  int movei[4]={1,0,-1,0};
8  int movej[4]={0,1,0,-1};
9
10 struct point{
11   int I, J, n;
12   point(){};
13   point(int I, int J, int n):I(I), J(J), n(n){};
14 };
15
16 int main(){
17   int Cas;
18   cin >> Cas;
19   while(Cas--){
20     memset(arr, 0, sizeof(arr));
21     queue<point> walk;
22     queue<point> fire;
23     int r, c;
24     cin >> r >> c;
25     for(int i = 0; i < r; i++){
26       for(int j = 0; j < c; j++){
27         char tmp;
28         cin >> tmp;
29         if(tmp == '#') arr[i][j] = -1;
30         if(tmp == 'F'){
31           arr[i][j] = 1;
32           fire.push(point(i, j, 0));
33         }
34         if(tmp == 'J'){
35           arr[i][j] = 2;
36           walk.push(point(i, j, 0));
37         }
38       }
39     }
40     int ans = 0;
41     while(!walk.empty()){
42       point now = walk.front();
43       walk.pop();
44       if(now.I == r-1 || now.I == 0 || now.J == c-1
              || now.J == 0){
45         ans = now.n+1;
46         break;
47       }
48       while(fire.front().n == now.n){
```

```
49           point tmp = fire.front();
50           fire.pop();
51           for(int i = 0; i < 4; i++){
52             int tmpi = tmp.I+movei[i];
53             int tmpj = tmp.J+movej[i];
54             if(tmpi < r && tmpi >= 0 && tmpj < c &&
                  tmpj >= 0){
55               if(arr[tmpi][tmpj] == 0){
56                 arr[tmpi][tmpj] = 1;
57                 fire.push(point(tmpi, tmpj, tmp.n+1));
58               }
59             }
60           }
61         }
62         for(int i = 0; i < 4; i++){
63           int tmpi = now.I+movei[i];
64           int tmpj = now.J+movej[i];
65           if(tmpi < r && tmpi >= 0 && tmpj < c && tmpj
                >= 0){
66             if(arr[tmpi][tmpj] == 0){
67               walk.push(point(tmpi, tmpj, now.n+1));
68             }
69           }
70         }
71       }
72       if(ans) cout << ans << '\n';
73       else cout << "IMPOSSIBLE\n";
74   }
75 }
```

3.8 ALLSUM

```
1  /*最大連續區間和*/
2  int ans = A[1], dp[N];
3  for (int i = 2; i <= n; ++i)
4  {
5      dp[i] = max(dp[i - 1], 0) + A[i];
6      ans = max(ans, dp[i]);
7  }
```

4 ENDLN

4.1 Minimum Edit Distance

```
1  // 利用 dfs 輸出替換字串的步驟
2  void backtracking(int i, int j){
3      if(i == 0 || j == 0){
4          while(i > 0){
5              cout << cnt++ << " Delete " << i << endl;
6              i--;
7          }
8          while(j > 0){
9              cout << cnt++ << " Insert " << i + 1 <<
                    "," << strB[j-1] << endl;
10             j--;
11         }
12         return;
13     }
14     if(strA[i-1] == strB[j-1]){
15         backtracking(i-1, j-1);
16     }
17     else{
18         if(dis[i][j] == dis[i-1][j-1] + 1){
19             cout << cnt++ << " Replace " << i << ","
                    << strB[j-1] << endl;
20             backtracking(i-1, j-1);
21         }
22         else if(dis[i][j] == dis[i-1][j] + 1){
23             cout << cnt++ << " Delete " << i << endl;
24             backtracking(i-1, j);
25         }
26         else if(dis[i][j] == dis[i][j-1] + 1){
```

```cpp
27          cout << cnt++ << " Insert " << i + 1 <<
                ","  << strB[j-1] << endl;
28          backtracking(i, j-1);
29      }
30    }
31 }
32 void MED(){
33     // 由於 B 是 0，所以 A 轉換成 B
           時每個字元都要被刪除
34     for(int i = 0; i <= strA.size(); ++i) dis[i][0] =
           i;
35     // 由於 A 是 0，所以 A 轉換成 B
           時每個字元都需要插入
36     for(int j = 0; j <= strB.size(); ++j) dis[0][j] =
           j;
37     for(int i = 1; i <= strA.size(); ++i){
38        for(int j = 1; j <= strB.size(); ++j){
39            // 字元相同代表不需修改，修改距離直接延續
40            if(strA[i-1] == strB[j-1]) dis[i][j] =
                  dis[i-1][j-1];
41            else{
42                // 取 replace , delete , insert
                      最小，選其 +1 為最少編輯距離
43                dis[i][j] = min(dis[i-1][j-1],
                      min(dis[i-1][j], dis[i][j-1])) +
                      1;
44            }
45        }
46     }
47 }
```

4.2 Bipatirate

```cpp
1 /* 二分圖 */
2 const int maxn = 300 + 5;
3 int n, color[maxn];
4 vector<vector<int>> v(maxn);
5 bool dfs(int s){
6     for(auto it : v[s]){
7         if(color[it] == -1){
8             color[it] = 3 - color[s];
9             if(!dfs(it)){
10                return false;
11            }
12        }
13        if(color[s] == color[it]){
14            return false;
15        }
16    }
17    return true;
18 }
19 void isBipatirate(){
20    bool flag = true;
21    for(int i = 1; i <= n; ++i){
22        if(color[i] == -1){
23            color[i] = 1;
24            flag &= dfs(i);
25        }
26    }
27    if(flag){
28        cout << "YES" << endl;
29    }
30    else{
31        cout << "NO" << endl;
32    }
33 }
34 int main(){
35    while(cin >> n && n){
36        for(int i = 1; i <= n; ++i) v[i].clear();
37        memset(color, -1, sizeof(color));
38        int a, b;
39        while(cin >> a >> b && (a || b)){
40            v[a].emplace_back(b);
41            v[b].emplace_back(a);
42        }
```

```cpp
43          isBipatirate();
44      }
45 }
```

4.3 LCA

```cpp
1 /*最低共同祖先*/
2 // 此 node 下有機顆 node
3 int dfs(int node, int dep){
4     depth[node] = dep + 1;
5     if(G[node].empty()){
6         siz[node] = 1;
7         return 1;
8     }
9     int total = 1;
10    for(auto i : G[node])
11        total += dfs(i.v, dep + 1);
12    siz[node] = total;
13    return siz[node];
14 }
15 // 找出每個節點的 2^i 倍祖先
16 // 2^20 = 1e6 > 200000
17 void find_parent(){
18    for(int i = 1; i < 20; i++)
19        for (int j = 0; j < N; j++)
20            parent[j][i] =
                  parent[parent[j][i-1]][i-1];
21 }
22 // 求兩點的 LCA（利用倍增法）
23 int LCA(int a, int b){
24    if (depth[b] < depth[a]) swap(a, b);
25    if (depth[a] != depth[b]){
26        int dif = depth[b] - depth[a];
27        for (int i = 0; i < 20; i++){
28            if (dif & 1) b = parent[b][i];
29            dif >>= 1;
30        }
31    }
32    if (a == b) return a;
33    for (int i = 19; i >= 0; i--){
34        if (parent[a][i] != parent[b][i]){
35            a = parent[a][i];
36            b = parent[b][i];
37        }
38    }
39    return parent[a][0];
40 }
```

4.4 Trie

```cpp
1 /* Trie 字典樹 */
2 struct Tire{
3     int path;
4     map<string, int> G[maxn];
5     void init(){
6         path = 1;
7         G[0].clear();
8     }
9     void insert(string str){
10        int u = 0;
11        string word = "";
12        for(int i = 0; i < str.size(); ++i){
13            if(str[i] == '\\'){
14                if(!G[u].count(word)){
15                    G[path].clear();
16                    G[u][word] = path++;
17                }
18                u = G[u][word];
19                word = "";
20            }
21            else word += str[i];
22        }
23    }
```

```
24    void put(int u, int space){
25        for(auto i = G[u].begin(); i != G[u].end();
            ++i){
26            for(int j = 0; j < space; ++j){
27                cout << " ";
28            }
29            cout << i->first << endl;
30            put(i->second, space + 1);
31        }
32    }
33 }tree;
```

4.5 GCD LCM

```
1 /*最大公因數*/
2 int gcd(int a, int b){
3     return (b == 0 ? a : gcd(b, a % b));
4 }
5 /*最小公倍數*/
6 int lcm(int a, int b){
7     return a * b / gcd(a, b);
8 }
9 /* 輾轉相除法 - 求兩數是否互質
10 如果兩數互質 最終結果其中一方為0時 另一方必為1
11 若兩數有公因數 最終結果其中一方為0時 另一方必不為1 */
12 while ( ( num1 %= num2 ) != 0 && ( num2 %= num1 ) !=
        0 );
```