Contents

1 語法

1.1 c++

```
// c++ code
#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)//刪除迭代器爲it處的元素，erase(first,last)
//刪除一個區間[first,last)內的所有元素，時間複雜度均爲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))


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

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

```
#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;
    // 做 n 輪
    for (int i = 0; i < n; i++) {
        relax = false;
        for (int j = 0; j < m; j++) {
            int u = edges[j].u;
            int v = edges[j].v;
            int w = edges[j].w;
            if (dis[u] == INF) {
                continue;
            }
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                relax = true;
            }
        }
        if (!relax) {
            break;
        }
    }
    return relax;
}


int main(){

}
```

2.2 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
#define M 100005
#define INF 1e9
struct Edge{
    int v, w;
    Edge(int a, int b):v(a), w(b){};
};
struct node{
    int u, dis;
    node(){};
    node(int a, int b):u(a), dis(b){};
    bool operator<(const node &r)const{
        return dis > r.dis;
    }
};
int dis[M]; //距離
vector<Edge> G[M];
void init(){
    fill(dis, dis+M, INF);
    for(int i = 0; i < M; i++){
        G[i].clear();
    }
}
void dijkstra(int start){
    dis[start] = 0;
    priority_queue<node> pq;
    pq.push(node(start, 0));
    while(!pq.empty()){
        node now = pq.top();
        pq.pop();
        if(now.dis > dis[now.u]) continue;
        for(Edge i : G[now.u]){
            if(dis[i.v] > now.dis + i.w){
                dis[i.v] = now.dis + i.w;
                pq.push(node(i.v, dis[i.v]));
                // printf("push(%d, %d)\n", i.v, dis[i.v]);
            }
        }
    }
}

int main(){
  int point, side;
    cin >> point >> side;
    init();
    for(int i = 0; i < side; i++){
        int s, t, w;
        cin >> s >> t >> w;
        G[s].push_back(Edge(t, w));
        G[t].push_back(Edge(s, w));
    }
    dijkstra(1);
    for(int i = 2; i <= point; i++){
        cout << dis[i] << '\n';
    }

}
```

2.3 Floyd-Warshall

```
#include<bits/stdc++.h>
using namespace std;
#define M 1005
#define INF 1e9

int dis[M][M];
// int G[M][M];
void init(int n){
    for(int i = 0; i <= n; i++){
        for(int j = 0; j <= n; j++){
            dis[i][j] = INF;
            if(i == j) dis[i][j] = 0;
        }
    }
}
void Floyd(int n){
    for(int k = 1; k <= n; k++){
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= i; j++){
                dis[i][j]= dis[j][i] = min(dis[i][k]+dis[k][j], dis[i][j]);
            }
        }
    }
}
void printarr(int r, int c){
    for(int i = 1; i <= r; i++){
        for(int j = 1; j <= c; j++){
            if(dis[i][j] == INF) cout << "INF ";
            else cout << dis[i][j] << ' ';
        }
        cout << '\n';
    }
}
int main(){
  int point, side;
    cin >> point >> side;
    init(point);
    for(int i = 0; i < side; i++){
        int s, t, w;
        cin >> s >> t >> w;
        dis[s][t] = w;
        dis[t][s] = w;
    }
    Floyd(point);
    int Cas;
    cin >> Cas;
    while(Cas--){
        int i, j;
        cin >> i >> j;
        cout << dis[i][j] << '\n';
    }
    // printarr(point, point);

}
```

2.4 SPFA

```
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

```
#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;
    }
}
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

```
// 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};
int Last[M] = {0};

bool Check(people a, people b){
    if(abs(a.high-b.high) > 40) return true;
    if(a.sex == b.sex) return true;
    if(a.music != b.music) return true;
    if(a.sport == b.sport) return true;
    return false;
}

bool KM(int x){
    for(int i = 0; i < G[x].size(); i++){
        int v = G[x][i];
        if(used[v]) continue;
        used[v] = 1;
        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
            Last[v] = x;
            return true;
        }
    }
    return false;
}

int Ans(int n){
    int Max = 0;
    memset(Last, -1, sizeof(Last));
    for(int i = 0; i < n; i++){
        memset(used, 0, sizeof(used));
        if(KM(i)){
            Max++;
        }
    }
    return Max;
}

int main(){
    int Cas;
    cin >> Cas;
    while(Cas--){
        int n;
        cin >> n;
        for(int i = 0; i < n; i++){
            G[i].clear();
            cin >> Class[i].high >> Class[i].sex >> Class[i].music >> Class[i].sport;
        }
        for(int i = 0; i < n; i++){
            if(Class[i].sex == 'M') continue;
            for(int j = 0; j < n; j++){
                if(i == j) continue;
                if(!Check(Class[i], Class[j])){
                    G[i].push_back(j);
                }
            }
        }
        int MaxPeople = n-Ans(n);
        cout << MaxPeople << '\n';
    }

}
```

3.2 LCS

```
int n1 = s1.size(), n2 = s2.size();
    int dp[N][N] = {};
    for (int i = 1; i <= n1; ++i)
    {
        for (int j = 1; j <= n2; ++j)
        {
            if (s1[i - 1] == s2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }

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

int dp[1005][1005] = {0};

int main(){
  string a, b;
    while(getline(cin, a) && getline(cin, b)){
        memset(dp, 0, sizeof(dp));
        int asize = a.size(), bsize = b.size();
        for(int i = 1; i <= asize; i++){
            for(int j = 1; j <= bsize; j++){
                if(a[i-1] == b[j-1]){
                    dp[i][j] = dp[i-1][j-1] + 1;
                }
                else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
            }
        }
        cout << dp[asize][bsize] << '\n';
    }

}

int n1 = s1.size(), n2 = s2.size();
int dp[2][N] = {};
for (int i = 1; i <= n1; i++)
{
    int cur = i % 2;
    int old = 1 - cur;
    for (int j = 1; j <= n2; ++j)
    {
        if (s1[i - 1] == s2[j - 1])
            dp[cur][j] = dp[old][j - 1] + 1;
        else
            dp[cur][j] = max(dp[old][j], dp[cur][j - 1]);

    }
}
```

3.3 LIS

```
#include<bits/stdc++.h>
using namespace std;
// 前後兩次LIS
int main(){
  int n;
    while(cin >> n){
        int arr[10005] = {0};
        int dp[10005] = {0};
        int dp2[10005] = {0};
        int Max = -1;
        for(int i = 0; i < n; i++){
            cin >> arr[i];
        }
        for(int i = 0; i < n; i++){
            dp[i] = 1;
            for(int j = 0; j < i; j++){
                if(arr[i] > arr[j]){
                    dp[i] = max(dp[i], dp[j]+1);
                }
            }
        }
        for(int i = n-1; i >= 0; i--){
            dp2[i] = 1;
            for(int j = n-1; j > i; j--){
                if(arr[i] > arr[j]){
                    dp2[i] = max(dp2[i], dp2[j]+1);
                }
            }
        }
        // for(int i = 0; i < n; i++){
        //     cout << arr[i] << ":\n"[i == n-1];
        // }
        // for(int i = 0; i < n; i++){
        //     cout << dp[i] << ":\n"[i == n-1];
        // }
        // for(int i = 0; i < n; i++){
        //     cout << dp2[i] << ":\n"[i == n-1];
        // }
        int lds = 0, lis = 0;
        for(int i = 0; i < n; i++){
            Max = max(Max, min(dp[i],dp2[i]));
        }
        cout << 2*Max-1 << '\n';
    }

}
```

3.4 merge

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

#define M 100010
// int cnt = 0;
void printarr(int arr[],int l,int r){
    for(int i=l;i<=r;i++){
        printf(" %d",arr[i]);
    }
    puts("");
}

int merge(int arr[], int l, int r, int mid){
    int L = l, R = mid+1;
    int tmplen = r-l+1, tmpi = 0;
    int tmp[M]={0};
  int cnt = 0;
    while(L <= mid && R <= r){
        if(arr[L]<=arr[R]){
            tmp[tmpi]=arr[L];
            L++;
        }
        else{
            tmp[tmpi]=arr[R];
      cnt += mid-L+1;
            R++;
        }
        tmpi++;
    }
    if(L>mid){
        while(R<=r){
            tmp[tmpi]=arr[R];
            R++;
            tmpi++;
        }
    }
    else{
        while(L<=mid){
            tmp[tmpi]=arr[L];
            L++;
            tmpi++;
        }
    }
    //L>mid&&R>r才可以全部跑過
    L=l;
    for (tmpi=0; tmpi<tmplen; tmpi++) {
        arr[L] = tmp[tmpi];
        L++;
    }

  // printf("%d %d %d:",l,mid,r);
    // printarr(arr,l,r);
  return cnt;
}

int mergeSort(int arr[],int l,int r){
  if(r <= l) return 0;
  int mid=(l+r)/2;
  int cnt = 0;
  cnt += mergeSort(arr, l, mid);
  cnt += mergeSort(arr, mid+1, r);
  cnt += merge(arr, l, r, mid);
    return cnt;
}

int main(){
  int n;
  while(cin >> n){
    if(n == 0) break;
    int arr[M] = {0};
    for(int i = 0; i < n; i++){
      cin >> arr[i];
    }
    if(mergeSort(arr, 0, n-1)%2) cout << "Marcelo\n";
    else cout << "Carlos\n";
  }

}
```

3.5 Prime

```
#include<bits/stdc++.h>
using namespace std;
#define M 10000
#define sq int(sqrt(double(M+5)));
bool prime[sq];
int main(){
    memset(prime, true, sizeof(prime));
    prime[0] = prime[1] = false;
    for(int i = 2; i <sq; i++){
        if(prime[i]){
            for(int j = i*i; j < sq; j+=i){
                prime[j] = false;
            }
        }
    }
}
```

3.6 UVA12321

```
#include<bits/stdc++.h>
using namespace std;
struct node{
    int l, r;
    node(){};
    node(int l, int r):l(l), r(r){};
    bool operator<(cnost node &a)const{
        return l < a.l;
    }
}

node gas[100005];
int main(){
    int L, G;
    while(cin >> L >> G){
        if(L == 0 && G == 0) break;
        for(int i = 0; i < G; i++){
            int a, b;
            cin >> a >> b;
            gas[i].l = a-b;
            gas[i].r = a+b;
        }
        sort(gas, gas+G);
        int ans = G, lcover = 0, rcover = 0,i = 0;
        while(L > lcover){
            rcover = lcover;
            for(; i < G && gas[i].l <= lcover; i++){
                if(gas[i].r > rcover) rcover = gas[i].r;
            }
            if(lcover == rcover) break;
            lcover = rcover;
            ans--;
        }
        if(lcover < L) cout << "-1\n";
        else cout << ans << '\n';
    }
}
// 天然氣
```

3.7 Fire

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

#define M 1005

int arr[M][M] = {0};
int movei[4]={1,0,-1,0};
int movej[4]={0,1,0,-1};

struct point{
  int I, J, n;
  point(){};
  point(int I, int J, int n):I(I), J(J), n(n){};
};

int main(){
  int Cas;
  cin >> Cas;
  while(Cas--){
    memset(arr, 0, sizeof(arr));
    queue<point> walk;
    queue<point> fire;
    int r, c;
    cin >> r >> c;
    for(int i = 0; i < r; i++){
      for(int j = 0; j < c; j++){
        char tmp;
        cin >> tmp;
        if(tmp == '#') arr[i][j] = -1;
        if(tmp == 'F'){
          arr[i][j] = 1;
          fire.push(point(i, j, 0));
        }
        if(tmp == 'J'){
          arr[i][j] = 2;
          walk.push(point(i, j, 0));
        }
      }
    }
    int ans = 0;
    while(!walk.empty()){
      point now = walk.front();
      walk.pop();
      if(now.I == r-1 || now.I == 0 || now.J == c-1 || now.J == 0){
        ans = now.n+1;
        break;
      }
      while(fire.front().n == now.n){
        point tmp = fire.front();
        fire.pop();
        for(int i = 0; i < 4; i++){
          int tmpi = tmp.I+movei[i];
          int tmpj = tmp.J+movej[i];
          if(tmpi < r && tmpi >= 0 && tmpj < c && tmpj >= 0){
            if(arr[tmpi][tmpj] == 0){
              arr[tmpi][tmpj] = 1;
              fire.push(point(tmpi, tmpj, tmp.n+1));
            }
          }
        }
      }
      for(int i = 0; i < 4; i++){
        int tmpi = now.I+movei[i];
        int tmpj = now.J+movej[i];
        if(tmpi < r && tmpi >= 0 && tmpj < c && tmpj >= 0){
          if(arr[tmpi][tmpj] == 0){
            walk.push(point(tmpi, tmpj, now.n+1));
          }
        }
      }
    }
    if(ans) cout << ans << '\n';
    else cout << "IMPOSSIBLE\n";
  }
}
```