

Contents

1 語法

- 1.1 c++ 1
- 1.2 python 1

2 Graph

- 2.1 Bellman-Ford 1
- 2.2 Dijkstra 2
- 2.3 Floyd-Warshall 2
- 2.4 SPFA 2
- 2.5 smallTree 2

3 Other

- 3.1 KM 3
- 3.2 LCS 3
- 3.3 LIS 4
- 3.4 merge 4
- 3.5 Prime 5
- 3.6 UVA12321 5
- 3.7 Fire 5
- 3.8 ALLSUM 5

1 語法

1.1 c++

```

1 // c++ code
2 std::ios::sync_with_stdio(false); // 加速
3
4 #include <bits/stdc++.h>
5 lower_bound(a, a + n, k); //最左邊 ≥ k 的位置
6 upper_bound(a, a + n, k); //最左邊 > k 的位置
7 upper_bound(a, a + n, k) - 1; //最右邊 ≤ k 的位置
8 lower_bound(a, a + n, k) - 1; //最右邊 < k 的位置
9 [lower_bound, upper_bound) //等於 k 的範圍
10 equal_range(a, a+n, k);
11
12 // 從小到大
13 priority_queue<int, vector<int>, greater<int>>pq
14
15 insert(it,x)//向vector的任意迭代器it處插入一個元素x
16 erase(it)//刪除迭代器為it處的元素，erase(first,last)
17 //刪除一個區間[first,last)內的所有元素，時間複雜度均為O(N)
18
19 set
20 insert(x) //將x插入set中 O(log(n))
21 count(x) //回傳x是否存在於set中() O(log(n))
22 erase(x) //刪除在set中的x O(log(n))
23 clear() //刪除set中所有元素 O(n)
24 empty() //回傳是否為空 O(1)
25 size() //回傳共有幾個元素 O(1)
26
27 map
28 insert(x) //將x這個pair插入map中 O(log(n))
29 count(x) //回傳x這個key是否在map中 O(log(n))
30 erase(x) //刪除在map中key為x的 O(log(n))
31
32 double cnt = 3.5555;
33 cout << fixed << setprecision(3) << cnt ;
34
35 #include <bits/stdc++.h>
36 using namespace std;
37
38 int main(){
39     set<int>s;
40     for(int i = 0; i < 10; i++){
41         s.insert(i);
42     }
43     cout << "lower bound: " << *s.lower_bound(5) <<
44         '\n';// 5
45     cout << "upper bound: " << *s.upper_bound(5) <<
46         '\n';// 6

```

```

46 if(s.lower_bound(20) == s.end()){
47     cout << "all elements are less than 20\n";
48 }
49 }

```

1.2 python

```

1 sorted((4,1,9,6),reverse=True)
2 fruits = ['apple', 'watermelon', 'pear', 'banana']
3 a = sorted(fruits, key = lambda x : len(x))
4 print(a)
5 # 輸出: ['pear', 'apple', 'banana', 'watermelon']
6 divmod(a,b)
7 把除數和餘數運算結果結合起來，
8 返回一個包含商和餘數的元組(a // b, a % b)
9
10 pow(base, exp[, mod])
11 >>> pow(38, -1, mod=97)
12 23
13 >>> 23 * 38 % 97 == 1
14 True
15
16 eof 寫法
17 try:
18     while True:
19         s = input()
20 except EOFError:
21     pass
22
23 eval(expression, globals=None, locals=None)
24
25 list(map(int, input().split()))
26 L.append(r)
27 my_list = ['This', 'is', 'a', 'string', 'in',
28            'Python']
29 my_string = " ".join(my_list)
30 #This is a string in Python
31 test = [[0 for j in range(m)] for i in range(n)]

```

2 Graph

2.1 Bellman-Ford

```

1 #include<iostream>
2 using namespace std;
3 const int INF = 1e9;
4 const int MAXN = 1000;
5 const int MAXM = 1000;
6 struct Edge {
7     int u;
8     int v;
9     int w;
10 };
11
12 int n, m;
13 Edge edges[MAXN];
14 int dis[MAXN];
15
16 // s是起點
17 bool bellman(int s) {
18     for (int i = 0; i < n; i++) {
19         dis[i] = INF;
20     }
21     dis[s] = 0;
22     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,
38                     //     dis[i.v]);
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 <= n; j++){
20                 dis[i][j] = min(dis[i][j],
21                     min(dis[i][k]+dis[k][j],
22                         dis[i][j]));
23             }
24         }
25     }
26 }
27 void printarr(int r, int c){
28     for(int i = 1; i <= r; i++){
29         for(int j = 1; j <= c; j++){
30             if(dis[i][j] == INF) cout << "INF ";
31             else cout << dis[i][j] << ' ';
32         }
33         cout << '\n';
34     }
35 }
36 int main(){
37     int point, side;
38     cin >> point >> side;
39     init(point);
40     for(int i = 0; i < side; i++){
41         int s, t, w;
42         cin >> s >> t >> w;
43         dis[s][t] = w;
44         dis[t][s] = w;
45     }
46     Floyd(point);
47     int Cas;
48     cin >> Cas;
49     while(Cas--){
50         int i, j;
51         cin >> i >> j;
52         cout << dis[i][j] << '\n';
53     }
54     // printarr(point, point);
55 }

```

2.4 SPFA

```

1 const int INF = 1e9;
2 const int MAXN = 1000;
3 struct Edge {

```

```

4   int v;
5   int w;
6 };
7 int n, m;
8 vector<Edge> G[MAXN];    //向量記圖
9 int dis[MAXN];
10 void SPFA(int s) {
11     // 記錄目前的點是否在 queue 中
12     bool inq[n];
13     for (int i = 0; i < n; i++) {
14         dis[i] = INF;
15         inq[i] = false;
16     }
17     dis[s] = 0;
18     inq[s] = true;
19     queue<int> q;
20     q.push(s);
21     while (!q.empty()) {
22         int u = q.front();
23         q.pop();
24         inq[u] = false;
25         for (Edge e : G[u]) {
26             if (dis[e.v] > dis[u] + e.w) {
27                 dis[e.v] = dis[u] + e.w;
28                 if (!inq[e.v]) {
29                     inq[e.v] = true;
30                     q.push(e.v);
31                 }
32             }
33         }
34     }
35 }
36
37 /*
38 Bellman Ford / SPFA 偵測負環
39
40 如果有一個點被放到 queue 裡面超過V次,那麼有負環
41 最大負環為包含所有點的環,共有V條邊,被更新V次
42 ,在極端的例子,被長度為1.2..3..V的路徑都
43 被更新一次最短距離。
44
45 比較
46 Floyd: ,需要計算許多點對的距離。
47 Dijkstra:沒有負邊且起點固定。
48 Bellman Ford / SPFA:其他狀況。
49 */

```

2.5 smallTree

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define M 100005
4 int tree[M] = {};    //parents
5 int r[M] = {};
6
7 struct Edge{
8     int s, t, w;
9     bool operator<(const Edge& r)const{
10         return w < r.w;
11     }
12 };
13
14 vector<Edge> G;
15
16 void init(int n){
17     for(int i = 0; i <= n; i++){
18         tree[i] = i;
19         r[i] = 1;
20     }
21 }
22
23 int Find(int n){
24     if(tree[n] == n) return n;    //find root
25     return tree[n] = Find(tree[n]);
26 }

```

```

26
27 void Union(int a, int b){
28     a = Find(a);
29     b = Find(b);
30     if (a == b) return;
31     if (r[a] <= r[b]){
32         tree[a] = b;    //a接b
33         r[b] += r[a];
34     }
35     else{
36         tree[b] = a;    //b接a
37         r[a] += r[b];
38     }
39 }
40
41 int kruskal(){
42     int cost = 0, flag = 0, Space = 0;
43     for (auto it : G){
44         it.s = Find(it.s);
45         it.t = Find(it.t);
46         if (it.s == it.t){
47             if(Space) cout << ' ';
48             Space = 1;
49             flag = 1;
50             cout << it.w;
51             continue;
52         }
53         cost += it.w;
54         Union(it.s, it.t);
55     }
56     return flag;
57 }
58
59 int main(){
60     int point, side, Max = 0;
61     while(cin >> point >> side){
62         G.clear();
63         if(point+side == 0) break;
64         init(point);
65         for(int i = 0; i < side; i++){
66             Edge tmp;
67             cin >> tmp.s >> tmp.t >> tmp.w;
68             G.push_back(tmp);
69         }
70         sort(G.begin(), G.end());
71         if(!kruskal()){
72             cout << "forest";
73         }
74         cout << '\n';
75     }
76 }

```

3 Other

3.1 KM

```

1 // uva12083
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 const int M = 500+5;
6 struct people{
7     int high;
8     char sex;
9     string music, sport;
10 };
11
12 vector<int> G[M];
13 people Class[M];
14 int used[M] = {0};
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])){
31             //v找到還沒配對的人或前一個v配對的人找到別人
32             Last[v] = x;
33             return true;
34         }
35     }
36     return false;
37 }
38
39 int Ans(int n){
40     int Max = 0;
41     memset(Last, -1, sizeof(Last));
42     for(int i = 0; i < n; i++){
43         memset(used, 0, sizeof(used));
44         if(KM(i)){
45             Max++;
46         }
47     }
48     return Max;
49 }
50
51 int main(){
52     int Cas;
53     cin >> Cas;
54     while(Cas--){
55         int n;
56         cin >> n;
57         for(int i = 0; i < n; i++){
58             G[i].clear();
59             cin >> Class[i].high >> Class[i].sex >>
60                 Class[i].music >> Class[i].sport;
61         }
62         for(int i = 0; i < n; i++){
63             if(Class[i].sex == 'M') continue;
64             for(int j = 0; j < n; j++){
65                 if(i == j) continue;
66                 if(!Check(Class[i], Class[j])){
67                     G[i].push_back(j);
68                 }
69             }
70         }
71         int MaxPeople = n - Ans(n);
72         cout << MaxPeople << '\n';
73     }
74 }

```

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]);
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],
33                                     dp[i][j-1]);
34             }
35         }
36         cout << dp[asize][bsize] << '\n';
37     }
38 }
39
40
41
42 int n1 = s1.size(), n2 = s2.size();
43 int dp[2][N] = {};
44 for (int i = 1; i <= n1; i++)
45 {
46     int cur = i % 2;
47     int old = 1 - cur;
48     for (int j = 1; j <= n2; ++j)
49     {
50         if (s1[i - 1] == s2[j - 1])
51             dp[cur][j] = dp[old][j - 1] + 1;
52         else
53             dp[cur][j] = max(dp[old][j], dp[cur][j - 1]);
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    }
31 }

```

```

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<(const 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 }

```