

Contents

1 字串

1.1	最長迴文子字串	1
1.2	KMP	1
2	math	1
2.1	模逆元	1
2.2	SG	1
2.3	Fibonacci	1
2.4	矩陣快速冪	1
2.5	質數與因數	1
2.6	歐拉函數	2
2.7	乘法逆元 and 組合數	2
3	algorithm	2
3.1	三分搜	2
3.2	差分	2
3.3	greedy	2
3.4	dinic	3
3.5	SCC Tarjan	4
3.6	ArticulationPoints Tarjan	4
3.7	最小樹狀圖	4
3.8	二分圖最大匹配	5
3.9	JosephusProblem	5
3.10	KM	5
3.11	LCA 倍增法	5
3.12	MCMF	6
3.13	Dancing Links	6
4	DataStructure	7
4.1	線段樹 1D	7
4.2	線段樹 2D	7
4.3	權值線段樹	8
4.4	Trie	8
4.5	單調隊列	8
5	geometry	8
5.1	intersection	8
5.2	半平面相交	9
5.3	凸包	9
6	DP	10
6.1	抽屜	10
6.2	Deque 最大差距	10
6.3	LCS 和 LIS	10
6.4	RangeDP	10
6.5	stringDP	10
6.6	樹 DP 有幾個 path 長度為 k	10
6.7	TreeDP reroot	10
6.8	WeightedLIS	11

1 字串

1.1 最長迴文子字串

```

1 #include<bits/stdc++.h>
2 #define T(x) ((x)%2 ? s[(x)/2] : '. ')
3 using namespace std;
4
5 string s;
6 int n;
7
8 int ex(int l,int r){
9     int i=0;
10    while(l-i>=0&&r+i<n&&T(l-i)==T(r+i)) i++;
11    return i;
12 }
13
14 int main(){
15     cin>>s;
16     n=2*s.size()+1;
17     int mx=0;
18     int center=0;
19     vector<int> r(n);
20     int ans=1;
21     r[0]=1;
22     for(int i=1;i<n;i++){
23         int ii=center-(i-center);
24         int len=mx-i+1;
25         if(i>mx){
26             r[i]=ex(i,i);
27             center=i;
28             mx=i+r[i]-1;
29         }
30         else if(r[ii]==len){
31             r[i]=len+ex(i-len,i+len);
32             center=i;

```

```

33         mx=i+r[i]-1;
34     }
35     else r[i]=min(r[ii],len);
36     ans=max(ans,r[i]);
37 }
38 cout<<ans-1<<"\n";
39 return 0;
40 }

```

1.2 KMP

```

2 #define maxn 1000005
2 int nextArr[maxn];
2 void getNextArr(const string& s) {
3     nextArr[0] = 0;
4     int prefixLen = 0;
5     for (int i = 1; i < s.size(); ++i) {
6         prefixLen = nextArr[i - 1];
7         //如果不一樣就在之前算過的prefix中
8         //搜有沒有更短的前後綴
9         while (prefixLen>0 && s[prefixLen]!=s[i])
10            prefixLen = nextArr[prefixLen - 1];
11        //一樣就繼承之前的前後綴長度+1
12        if (s[prefixLen] == s[i])
13            ++prefixLen;
14        nextArr[i] = prefixLen;
15    }
16    for (int i = 0; i < s.size() - 1; ++i) {
17        vis[nextArr[i]] = true;
18    }
19 }
20 }

```

2 math

2.1 模逆元

$$x^{-1} \bmod m = \begin{cases} 1, & \text{if } x = 1 \\ -\lfloor \frac{m}{x} \rfloor (m \bmod x)^{-1}, & \text{otherwise} \end{cases} \quad (\bmod m)$$

$$= (m - \lfloor \frac{m}{x} \rfloor)(m \bmod x)^{-1} \quad (\bmod m)$$

若 $p \in \text{prime}$, 根據費馬小定理, 則

$$\begin{aligned} \therefore ax &\equiv 1 \pmod{p} \\ \therefore ax &\equiv a^{p-1} \pmod{p} \\ \therefore x &\equiv a^{p-2} \pmod{p} \end{aligned}$$

2.2 SG

$$SG(x) = \text{mex}\{SG(y) | x \rightarrow y\}$$

$$\text{mex}(S) = \min\{n | n \in \mathbb{N}, n \notin S\}$$

2.3 Fibonacci

$$\begin{bmatrix} f_{n-1} & f_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} f_n & f_{n+1} \end{bmatrix}$$

$$\begin{bmatrix} f_n & f_{n+1} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^p = \begin{bmatrix} f_{n+p} & f_{n+p+1} \end{bmatrix}, p \in \mathbb{N}$$

2.4 矩陣快速冪

```

1 using ll = long long;
2 using mat = vector<vector<ll>>;
3 const int mod = 1e9 + 7;
4
5 mat operator*(mat A, mat B) {
6     mat res(A.size(), vector<ll>(B[0].size()));
7     for(int i=0; i<A.size(); i++) {
8         for(int j=0; j<B[0].size(); j++) {
9             for(int k=0; k<B.size(); k++) {
10                res[i][j] += A[i][k] * B[k][j] % mod;
11            }
12            res[i][j] %= mod;
13        }
14    }
15 }

```

```

14 }
15 return res;
16 }
17
18 mat I = ;
19 // compute matrix M^n
20 // 需先 init I 矩陣
21 mat mpow(mat& M, int n) {
22     if(n <= 1) return n ? M : I;
23     mat v = mpow(M, n>>1);
24     return (n & 1) ? v*v*M : v*v;
25 }
26
27 // 迴圈版本
28 mat mpow(mat M, int n) {
29     mat res(M.size(), vector<ll>(M[0].size()));
30     for(int i=0; i<res.size(); i++)
31         res[i][i] = 1;
32     for(; n; n>>=1) {
33         if(n & 1) res = res * M;
34         M = M * M;
35     }
36     return res;
37 }

```

2.5 質數與因數

```

1 歐拉篩O(n)
2 #define MAXN 47000 //sqrt(2^31)=46,340...
3 bool isPrime[MAXN];
4 int p[MAXN];
5 int pSize=0;
6 void getPrimes(){
7     memset(isPrime,true,sizeof(isPrime));
8     isPrime[0]=isPrime[1]=false;
9     for(int i=2;i<MAXN;i++){
10        if(isPrime[i]) p[pSize++]=i;
11        for(int j=0;j<pSize&&i*p[j]<=MAXN;j++){
12            isPrime[i*p[j]]=false;
13            if(i%p[j]==0) break;
14        }
15    }
16 }
17
18 最大公因數 O(log(min(a,b)))
19 int GCD(int a, int b){
20     if(b == 0) return a;
21     return GCD(b, a%b);
22 }
23
24 質因數分解
25 void primeFactorization(int n){
26     for(int i=0; i<p.size(); ++i) {
27         if(p[i]*p[i] > n) break;
28         if(n % p[i]) continue;
29         cout << p[i] << ' ';
30         while(n%p[i] == 0) n /= p[i];
31     }
32     if(n != 1) cout << n << ' ';
33     cout << '\n';
34 }
35
36 擴展歐幾里得算法 ax + by = GCD(a, b)
37 int ext_euc(int a, int b, int &x, int &y) {
38     if(b == 0){
39         x = 1, y = 0;
40         return a;
41     }
42     int d = ext_euc(b, a%b, y, x);
43     y -= a/b*x;
44     return d;
45 }
46 int main(){
47     int a, b, x, y;
48     cin >> a >> b;
49     ext_euc(a, b, x, y);

```

```

50 cout << x << ' ' << y << endl;
51 return 0;
52 }
53
54 歌德巴赫猜想
55 解：把偶數 N ( $6 \leq N \leq 10^6$ ) 寫成兩個質數的和。
56 #define N 2000000
57 int ox[N], p[N], pr;
58 void PrimeTable(){
59     ox[0] = ox[1] = 1;
60     pr = 0;
61     for(int i=2; i<N; i++){
62         if(!ox[i]) p[pr++] = i;
63         for(int j=0; i*p[j]<N&&j<pr; j++)
64             ox[i*p[j]] = 1;
65     }
66 }
67
68 int main(){
69     PrimeTable();
70     int n;
71     while(cin>>n, n){
72         int x;
73         for(x=1;; x+=2)
74             if(!ox[x] && !ox[n-x]) break;
75         printf("%d = %d + %d\n", n, x, n-x);
76     }
77 }
78
79 problem :
80 給定整數 N，求N最少可以拆成多少個質數的和。
81 如果N是質數，則答案為 1。
82 如果N是偶數 ( $N \neq 2$ )，則答案為2(強歌德巴赫猜想)。
83 如果N是奇數且  $N-2$  是質數，則答案為2(2+質數)。
84 其他狀況答案為 3 (弱歌德巴赫猜想)。

```

```

85
86 bool isPrime(int n){
87     for(int i=2; i<n; ++i){
88         if(i*i>n) return true;
89         if(n%i==0) return false;
90     }
91     return true;
92 }
93
94 int main(){
95     int n;
96     cin>>n;
97     if(isPrime(n)) cout<<"1\n";
98     else if(n%2==0 || isPrime(n-2)) cout<<"2\n";
99     else cout<<"3\n";
100 }

```

2.6 歐拉函數

```

1 //計算閉區間 [1, n] 中有幾個正整數與 n 互質
2
3 int phi(){
4     int ans=n;
5     for(int i=2; i*i<=n; i++){
6         if(n%i==0){
7             ans=ans-i;
8             while(n%i==0) n/=i;
9         }
10     if(n>1) ans=ans-ans/n;
11     return ans;
12 }

```

2.7 乘法逆元 and 組合數

```

1 using ll = long long;
2 const int maxn = 2e5 + 10;
3 const int mod = 1e9 + 7;
4
5 int fact[maxn] = {1, 1}; // x! % mod

```

```

6 int inv[maxn] = {1, 1}; // x^(-1) % mod
7 int invFact[maxn] = {1, 1}; // (x!)^(-1) % mod
8
9 void build() {
10     for(int x=2; x<maxn; x++) {
11         fact[x] = (ll)x * fact[x-1] % mod;
12         inv[x] = (ll)(mod-mod/x)*inv[mod%x]%mod;
13         invFact[x] = (ll)invFact[x-1]*inv[x]%mod;
14     }
15 }
16
17 // 前提: mod 為質數
18 void build() {
19     auto qpow = [&](ll a, int b) {
20         ll res = 1;
21         for(; b; b>>=1) {
22             if(b & 1) res = res * a % mod;
23             a = a * a % mod;
24         }
25         return res;
26     };
27
28     for(int x=2; x<maxn; x++) {
29         fact[x] = (ll)x * fact[x-1] % mod;
30         invFact[x] = qpow(fact[x], mod-2);
31     }
32 }
33
34 // C(a, b) % mod
35 int comb(int a, int b) {
36     if(a < b) return 0;
37     ll x = fact[a];
38     ll y = (ll)invFact[b] * invFact[a-b] % mod;
39     return x * y % mod;
40 }

```

3 algorithm

3.1 三分搜

```

1 題意
2 給定兩射線方向和速度，問兩射線最近距離。
3 題解
4 假設 F(t) 為兩射線在時間 t 的距離，F(t)
5 為二次函數，
6 可用三分查找二次函數最小值。
7 struct Point{
8     double x, y, z;
9     Point() {}
10     Point(double _x, double _y, double _z):
11         x(_x), y(_y), z(_z){}
12     friend istream& operator>>(istream& is,
13         Point& p) {
14         is >> p.x >> p.y >> p.z;
15         return is;
16     }
17 }
18 Point operator+(const Point &rhs) const{
19     return Point(x+rhs.x, y+rhs.y, z+rhs.z);
20 }
21 Point operator-(const Point &rhs) const{
22     return Point(x-rhs.x, y-rhs.y, z-rhs.z);
23 }
24 Point operator*(const double &d) const{
25     return Point(x*d, y*d, z*d);
26 }
27 Point operator/(const double &d) const{
28     return Point(x/d, y/d, z/d);
29 }
30 double dist(const Point &rhs) const{
31     double res = 0;
32     res+=(x-rhs.x)*(x-rhs.x);
33     res+=(y-rhs.y)*(y-rhs.y);
34     res+=(z-rhs.z)*(z-rhs.z);
35     return res;
36 }
37
38 int main(){
39     IOS; //輸入優化
40     int T;

```

```

38 cin>>T;
39 for(int ti=1; ti<=T; ++ti){
40     double time;
41     Point x1, y1, d1, x2, y2, d2;
42     cin>>time>>x1>>y1>>x2>>y2;
43     d1=(y1-x1)/time;
44     d2=(y2-x2)/time;
45     double L=0, R=1e8, m1, m2, f1, f2;
46     double ans = x1.dist(x2);
47     while(abs(L-R)>1e-10){
48         m1=(L+R)/2;
49         m2=(m1+R)/2;
50         f1=((d1*m1)+x1).dist((d2*m1)+x2);
51         f2=((d1*m2)+x1).dist((d2*m2)+x2);
52         ans = min(ans, min(f1, f2));
53         if(f1<f2) R=m2;
54         else L=m1;
55     }
56     cout<<"Case "<<ti<<": ";
57     cout << fixed << setprecision(4) <<
58         sqrt(ans) << '\n';
59 }

```

3.2 差分

```

1 用途：在區間 [l, r] 加上一個數字 v。
2 b[l] += v; (b[0~l] 加上v)
3 b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v))
4 給的 a[] 是前綴和數列，建構 b[]，
5 因為 a[i] = b[0] + b[1] + b[2] + ... + b[i]，
6 所以 b[i] = a[i] - a[i-1]。
7 在 b[l] 加上 v，b[r+1] 減去 v，
8 最後再從 0 跑到 n 使 b[i] += b[i-1]。
9 這樣一來，b[] 是一個在某區間加上v的前綴和。
10 int a[1000], b[1000];
11 // a: 前綴和數列, b: 差分數列
12 int main(){
13     int n, l, r, v;
14     cin >> n;
15     for(int i=1; i<=n; i++){
16         cin >> a[i];
17         b[i] = a[i] - a[i-1]; //建構差分數列
18     }
19     cin >> l >> r >> v;
20     b[l] += v;
21     b[r+1] -= v;
22     for(int i=1; i<=n; i++){
23         b[i] += b[i-1];
24         cout << b[i] << ' ';
25     }
26 }

```

3.3 greedy

```

1 刪數字問題
2 //problem
3 給定一個數字 N ( $\leq 10^4$ )，需要刪除 K 個數字，
4 請問刪除 K 個數字後最小的數字為何？
5 //solution
6 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i
7 位數，
8 扣除高位數的影響較扣除低位數的大。
9 //code
10 int main(){
11     string s;
12     int k;
13     cin>>s>>k;
14     for(int i=0; i<k; ++i){
15         if((int)s.size()==0) break;
16         int pos = (int)s.size()-1;
17         for(int j=0; j<(int)s.size()-1; ++j){
18             if(s[j]>s[j+1]){
19                 pos=j;
20             }
21         }
22         s.erase(pos);
23     }
24     cout << s << endl;
25 }

```

```

19         break;
20     }
21 }
22 s.erase(pos,1);
23 }
24 while((int)s.size()>0&&s[0]=='0')
25     s.erase(0,1);
26 if((int)s.size()) cout<<s<<'\\n';
27 else cout<<0<<'\\n';
28 }
29 最小區間覆蓋長度
30 //problem
31 給定 n 條線段區間為 [Li,Ri],
32 請問最少要選幾個區間才能完全覆蓋 [0,S]?
33 //solution
34 先將所有區間依照左界由小到大排序,
35 對於當前區間 [Li,Ri],要從左界 >Ri 的所有區間中,
36 找到有著最大的右界的區間,連接當前區間。
37
38 //problem
39 長度 n 的直線中有數個加熱器,
40 在 x 的加熱器可以讓 [x-r,x+r] 內的物品加熱,
41 問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
42 //solution
43 對於最左邊沒加熱的點a,選擇最遠可以加熱a的加熱器,
44 更新已加熱範圍,重複上述動作繼續尋找加熱器。
45 //code
46 int main(){
47     int n, r;
48     int a[1005];
49     cin>>n>>r;
50     for(int i=1;i<=n;++i) cin>>a[i];
51     int i=1,ans=0;
52     while(i<=n){
53         int R=min(i+r-1,n),L=max(i-r+1,0)
54         int nextR=-1;
55         for(int j=R;j>=L;--j){
56             if(a[j]){
57                 nextR=j;
58                 break;
59             }
60         }
61         if(nextR==-1){
62             ans=-1;
63             break;
64         }
65         ++ans;
66         i=nextR+r;
67     }
68     cout<<ans<<'\\n';
69 }
70 最多不重疊區間
71 //problem
72 給你 n 條線段區間為 [Li,Ri],
73 請問最多可以選擇幾條不重疊的線段(頭尾可相連)?
74 //solution
75 依照右界由小到大排序,
76 每次取到一個不重疊的線段,答案 +1。
77 //code
78 struct Line{
79     int L,R;
80     bool operator<(const Line &rhs)const{
81         return R<rhs.R;
82     }
83 };
84 int main(){
85     int t;
86     cin>>t;
87     Line a[30];
88     while(t--){
89         int n=0;
90         while(cin>>a[n].L>>a[n].R,a[n].L||a[n].R)
91             ++n;
92         sort(a,a+n);
93         int ans=1,R=a[0].R;
94         for(int i=1;i<=n;i++){
95             if(a[i].L>=R){
96                 ++ans;
97                 R=a[i].R;
98             }
99         }
100         cout<<ans<<'\\n';
101     }
102 }
103 最小化最大延遲問題
104 //problem
105 給定 N 項工作, 每項工作的需要處理時長為 Ti,
106 期限是 Di, 第 i 項工作延遲的時間為
107     Li=max(0,Fi-Di),
108 原本Fi 為第 i 項工作的完成時間,
109 求一種工作排序使 maxLi 最小。
110 //solution
111 按照到期時間從早到晚處理。
112 //code
113 struct Work{
114     int t, d;
115     bool operator<(const Work &rhs)const{
116         return d<rhs.d;
117     };
118 }
119 int main(){
120     int n;
121     Work a[10000];
122     cin>>n;
123     for(int i=0;i<n;++i)
124         cin>>a[i].t>>a[i].d;
125     sort(a,a+n);
126     int maxL=0,sumT=0;
127     for(int i=0;i<n;++i){
128         sumT+=a[i].t;
129         maxL=max(maxL,sumT-a[i].d);
130     }
131     cout<<maxL<<'\\n';
132 }
133 最少延遲數量問題
134 //problem
135 給定 N 個工作, 每個工作的需要處理時長為 Ti,
136 期限是 Di, 求一種工作排序使得逾期工作數量最小。
137 //solution
138 期限越早到期的工作越先做。
139 將工作依照到期時間從早到晚排序,
140 依序放入工作列表中, 如果發現有工作預期,
141 就從目前選擇的工作中, 移除耗時最長的工作。
142 上述方法為 Moore-Hodgson s Algorithm。
143 //problem
144 給定烏龜的重量和可承受重量, 問最多可以疊幾隻烏龜?
145 //solution
146 和最少延遲數量問題是相同的問題, 只要將題敘做轉換。
147 工作處理時長 → 烏龜重量
148 工作期限 → 烏龜可承受重量
149 多少工作不延期 → 可以疊幾隻烏龜
150 //code
151 struct Work{
152     int t, d;
153     bool operator<(const Work &rhs)const{
154         return d<rhs.d;
155     }
156 };
157 int main(){
158     int n=0;
159     Work a[10000];
160     priority_queue<int> pq;
161     while(cin>>a[n].t>>a[n].d)
162         ++n;
163     sort(a,a+n);
164     int sumT=0,ans=n;
165     for(int i=0;i<n;++i){
166         pq.push(a[i].t);
167         sumT+=a[i].t;
168         if(a[i].d<sumT){
169             int x=pq.top();
170             pq.pop();
171             sumT-=x;
172             --ans;
173         }
174     }
175     cout<<ans<<'\\n';
176 }
177 }
178 任務調度問題
179 //problem
180 給定 N 項工作, 每項工作的需要處理時長為 Ti,
181 期限是 Di, 如果第 i 項工作延遲需要受到 pi
182     單位懲罰,
183 請問最少會受到多少單位懲罰。
184 //solution
185 依照懲罰由大到小排序,
186 每項工作依序嘗試可不可以放在
187     Di-Ti+1,Di-Ti,...,1,0,
188 如果有空閒就放進去, 否則延後執行。
189 //problem
190 給定 N 項工作, 每項工作的需要處理時長為 Ti,
191 期限是 Di, 如果第 i 項工作在期限內完成會獲得 ai
192     單位獎勵,
193 請問最多會獲得多少單位獎勵。
194 //solution
195 和上題相似, 這題變成依照獎勵由大到小排序。
196 //code
197 struct Work{
198     int d,p;
199     bool operator<(const Work &rhs)const{
200         return p>rhs.p;
201     }
202 };
203 int main(){
204     int n;
205     Work a[100005];
206     bitset<100005> ok;
207     while(cin>>n){
208         ok.reset();
209         for(int i=0;i<n;++i)
210             cin>>a[i].d>>a[i].p;
211         sort(a,a+n);
212         int ans=0;
213         for(int i=0;i<n;++i){
214             int j=a[i].d;
215             while(j--){
216                 if(!ok[j]){
217                     ans+=a[i].p;
218                     ok[j]=true;
219                     break;
220                 }
221             }
222         }
223     }
224     cout<<ans<<'\\n';
225 }

```

3.4 dinic

```

1 const int maxn = 1e5 + 10;
2 const int inf = 0x3f3f3f3f;
3 struct Edge {
4     int s, t, cap, flow;
5 };
6 int n, m, S, T;
7 int level[maxn], dfs_idx[maxn];
8 vector<Edge> E;
9 vector<vector<int>> G;
10 void init() {
11     S = 0;
12     T = n + m;
13     E.clear();
14     G.assign(maxn, vector<int>());
15 }
16 void addEdge(int s, int t, int cap) {
17     E.push_back({s, t, cap, 0});
18     E.push_back({t, s, 0, 0});
19     G[s].push_back(E.size()-2);
20     G[t].push_back(E.size()-1);
21 }
22 bool bfs() {

```

```

23 queue<int> q({S});
24 memset(level, -1, sizeof(level));
25 level[S] = 0;
26 while(!q.empty()) {
27     int cur = q.front();
28     q.pop();
29     for(int i : G[cur]) {
30         Edge e = E[i];
31         if(level[e.t]==-1 &&
32             e.cap>e.flow) {
33             level[e.t] = level[e.s] + 1;
34             q.push(e.t);
35         }
36     }
37     return ~level[T];
38 }
39 int dfs(int cur, int lim) {
40     if(cur==T || lim==0) return lim;
41     int result = 0;
42     for(int& i=dfs_idx[cur]; i<G[cur].size()
43         && lim; i++) {
44         Edge& e = E[G[cur][i]];
45         if(level[e.s]+1 != level[e.t])
46             continue;
47         int flow = dfs(e.t, min(lim,
48             e.cap-e.flow));
49         if(flow <= 0) continue;
50         e.flow += flow;
51         result += flow;
52         E[G[cur][i]^1].flow -= flow;
53         lim -= flow;
54     }
55     return result;
56 }
57 int dinic() { // O((V^2)E)
58     int result = 0;
59     while(bfs()) {
60         memset(dfs_idx, 0, sizeof(dfs_idx));
61         result += dfs(S, inf);
62     }
63     return result;
64 }

```

```

32 //如果是 SCC
33 if (dfn[u] == low[u]) {
34     long long minCost = 0x3f3f3f3f;
35     int currWays = 0;
36     ++SCC;
37     while (1) {
38         int v = sk.top();
39         inStack[v] = 0;
40         sk.pop();
41         if (minCost > cost[v]) {
42             minCost = cost[v];
43             currWays = 1;
44         }
45         else if (minCost == cost[v]) {
46             ++currWays;
47         }
48         if (v == u)
49             break;
50     }
51     totalCost += minCost;
52     ways = (ways * currWays) % MOD;
53 }
54 }
55 int main() {
56     int n;
57     scanf("%d", &n);
58     for (int i = 1; i <= n; ++i)
59         scanf("%lld", &cost[i]);
60     G.assign(n + 5, vector<int>());
61     int m;
62     scanf("%d", &m);
63     int u, v;
64     for (int i = 0; i < m; ++i) {
65         scanf("%d %d", &u, &v);
66         G[u].emplace_back(v);
67     }
68     for (int i = 1; i <= n; ++i) {
69         if (dfn[i] == 0)
70             dfs(i);
71     }
72     printf("%lld %lld\n", totalCost, ways %
73         MOD);
74     return 0;

```

```

29 if (isCut) ++res;
30 }
31 int main() {
32     char input[105];
33     char* token;
34     while (scanf("%d", &N) != EOF && N) {
35         G.assign(105, vector<int>());
36         memset(visited, false,
37             sizeof(visited));
38         memset(low, 0, sizeof(low));
39         memset(dfn, 0, sizeof(visited));
40         timer = 0;
41         res = 0;
42         getchar(); // for \n
43         while (fgets(input, 105, stdin)) {
44             if (input[0] == '\0')
45                 break;
46             int size = strlen(input);
47             input[size - 1] = '\0';
48             --size;
49             token = strtok(input, " ");
50             int u = atoi(token);
51             int v;
52             while (token = strtok(NULL, " ")) {
53                 v = atoi(token);
54                 G[u].emplace_back(v);
55                 G[v].emplace_back(u);
56             }
57             tarjan(1, -1);
58             printf("%d\n", res);
59         }
60         return 0;
61     }

```

3.5 SCC Tarjan

```

1 //單純考 SCC，每個 SCC 中找成本最小的 蓋，如果有多個一樣小
2 //的要數出來，因為題目要方法數
3 //注意以下程式有縮點，但沒存起來，
4 //存法就是開一個 array -> ID[u] = SCCID
5 #define maxn 100005
6 #define MOD 1000000007
7 long long cost[maxn];
8 vector<vector<int>> G;
9 int SCC = 0;
10 stack<int> sk;
11 int dfn[maxn];
12 int low[maxn];
13 bool inStack[maxn];
14 int dfsTime = 1;
15 long long totalCost = 0;
16 long long ways = 1;
17 void dfs(int u) {
18     dfn[u] = low[u] = dfsTime;
19     ++dfsTime;
20     sk.push(u);
21     inStack[u] = true;
22     for (int v: G[u]) {
23         if (dfn[v] == 0) {
24             dfs(v);
25             low[u] = min(low[u], low[v]);
26         }
27         else if (inStack[v]) {
28             //屬於同個 SCC 且是我的 back edge
29             low[u] = min(low[u], dfn[v]);
30         }
31     }

```

3.6 ArticulationPoints Tarjan

```

1 vector<vector<int>> G;
2 int N, timer;
3 bool visited[105];
4 int dfn[105]; // 第一次 visit 的時間
5 int low[105];
6 //最小能回到的父節點
7 //(不能是自己的 parent) 的 visTime
8 int res;
9 //求割點數量
10 void tarjan(int u, int parent) {
11     int child = 0;
12     bool isCut = false;
13     visited[u] = true;
14     dfn[u] = low[u] = ++timer;
15     for (int v: G[u]) {
16         if (!visited[v]) {
17             ++child;
18             tarjan(v, u);
19             low[u] = min(low[u], low[v]);
20             if (parent != -1 && low[v] >=
21                 dfn[u])
22                 isCut = true;
23         }
24         else if (v != parent)
25             low[u] = min(low[u], dfn[v]);
26     }
27     //If u is root of DFS
28     //tree->有兩個以上的 children
29     if (parent == -1 && child >= 2)
30         isCut = true;

```

3.7 最小樹狀圖

```

1 const int maxn = 60 + 10;
2 const int inf = 0x3f3f3f3f;
3 struct Edge {
4     int s, t, cap, cost;
5 }; // cap 為頻寬 (optional)
6 int n, m, c;
7 int inEdge[maxn], idx[maxn], pre[maxn],
8     vis[maxn];
9 // 對於每個點，選擇對它入度最小的那條邊
10 // 找環，如果沒有則 return;
11 // 進行縮環並更新其他點到環的距離。
12 int dirMST(vector<Edge> edges, int low) {
13     int result = 0, root = 0, N = n;
14     while(true) {
15         memset(inEdge, 0x3f, sizeof(inEdge));
16         // 找所有點的 in edge 放進 inEdge
17         // optional: low 為最小 cap 限制
18         for(const Edge& e : edges) {
19             if(e.cap < low) continue;
20             if(e.s!=e.t &&
21                 e.cost<inEdge[e.t]) {
22                 inEdge[e.t] = e.cost;
23                 pre[e.t] = e.s;
24             }
25         }
26         for(int i=0; i<N; i++) {
27             if(i!=root && inEdge[i]==inf)
28                 return -1; //除了 root 還有點沒有 in
29                 edge
30         }
31         int seq = inEdge[root] = 0;
32         memset(idx, -1, sizeof(idx));
33         memset(vis, -1, sizeof(vis));
34         // 找所有的 cycle，一起編號為 seq
35         for(int i=0; i<N; i++) {
36             result += inEdge[i];
37             int cur = i;
38             while(vis[cur]!=i &&
39                 idx[cur]==-1) {

```

```

36         if(cur == root) break;
37         vis[cur] = i;
38         cur = pre[cur];
39     }
40     if(cur != root && idx[cur] == -1) {
41         for(int j = pre[cur]; j != cur;
42             j = pre[j])
43             idx[j] = seq;
44         idx[cur] = seq++;
45     }
46     if(seq == 0) return result; // 沒有
47     cycle
48     for(int i = 0; i < N; i++)
49         // 沒有被縮點的點
50         if(idx[i] == -1) idx[i] = seq++;
51     // 縮點並重新編號
52     for(Edge& e : edges) {
53         if(idx[e.s] != idx[e.t])
54             e.cost -= inEdge[e.t];
55         e.s = idx[e.s];
56         e.t = idx[e.t];
57     }
58     N = seq;
59     root = idx[root];
60 }

```

3.8 二分圖最大匹配

```

1 /* 核心：最大點獨立集 = |V| -
   /最大匹配數/，用匈牙利演算法找出最大匹配數 */
2 vector<Student> boys;
3 vector<Student> girls;
4 vector<vector<int>> G;
5 bool used[505];
6 int p[505];
7 bool match(int i) {
8     for (int j: G[i]) {
9         if (!used[j]) {
10             used[j] = true;
11             if (p[j] == -1 || match(p[j])) {
12                 p[j] = i;
13                 return true;
14             }
15         }
16     }
17     return false;
18 }
19 void maxMatch(int n) {
20     memset(p, -1, sizeof(p));
21     int res = 0;
22     for (int i = 0; i < boys.size(); ++i) {
23         memset(used, false, sizeof(used));
24         if (match(i))
25             ++res;
26     }
27     cout << n - res << '\n';
28 }

```

3.9 Josephus Problem

```

1 //Josephus Problem，只是規定要先砍1號
2 //所以當作有 n - 1 個人，目標的13順移成12
3 //再者從0開始比較好算，所以目標12順移成11
4 int getWinner(int n, int k) {
5     int winner = 0;
6     for (int i = 1; i <= n; ++i)
7         winner = (winner + k) % i;
8     return winner;
9 }
10 int main() {
11     int n;
12     while (scanf("%d", &n) != EOF && n){

```

```

13         --n;
14         for (int k = 1; k <= n; ++k){
15             if (getWinner(n, k) == 11){
16                 printf("%d\n", k);
17                 break;
18             }
19         }
20     }
21     return 0;
22 }

```

3.10 KM

```

1 #define maxn 505
2 int W[maxn][maxn];
3 int Lx[maxn], Ly[maxn];
4 bool S[maxn], T[maxn];
5 //L[i] = j -> S_i配給T_j, -1 for 還沒匹配
6 int L[maxn];
7 int n;
8 bool match(int i) {
9     S[i] = true;
10    for (int j = 0; j < n; ++j) {
11        // KM重點
12        // Lx + Ly >= selected_edge(x, y)
13        // 要想辦法降低Lx + Ly
14        // 所以選Lx + Ly == selected_edge(x, y)
15        if (Lx[i] + Ly[j] == W[i][j] &&
16            !T[j]) {
17            T[j] = true;
18            if ((L[j] == -1) || match(L[j])) {
19                L[j] = i;
20                return true;
21            }
22        }
23    }
24    return false;
25 }
26 //修改二分圖上的交錯路徑上點的權重
27 //此舉是在通過調整vertex labeling看看
28 //能不能產生出新的增廣路
29 //(KM的增廣路要求Lx[i] + Ly[j] == W[i][j])
30 //在這裡優先從最小的diff調看，才能保證最大權重匹配
31 void update()
32 {
33     int diff = 0x3f3f3f3f;
34     for (int i = 0; i < n; ++i) {
35         if (S[i]) {
36             for (int j = 0; j < n; ++j) {
37                 if (!T[j])
38                     diff = min(diff, Lx[i] +
39                                 Ly[j] - W[i][j]);
40             }
41         }
42     }
43     for (int i = 0; i < n; ++i) {
44         if (S[i]) Lx[i] -= diff;
45         if (T[i]) Ly[i] += diff;
46     }
47 }
48 void KM()
49 {
50     for (int i = 0; i < n; ++i) {
51         L[i] = -1;
52         Lx[i] = Ly[i] = 0;
53         for (int j = 0; j < n; ++j)
54             Lx[i] = max(Lx[i], W[i][j]);
55     }
56     for (int i = 0; i < n; ++i) {
57         while(1) {
58             memset(S, false, sizeof(S));
59             memset(T, false, sizeof(T));
60             if (match(i))
61                 break;
62             else
63                 update(); //去調整vertex
64                             labeling以增加增廣路徑

```

```

62     }
63 }
64 }
65 int main() {
66     while (scanf("%d", &n) != EOF) {
67         for (int i = 0; i < n; ++i)
68             for (int j = 0; j < n; ++j)
69                 scanf("%d", &W[i][j]);
70         KM();
71         int res = 0;
72         for (int i = 0; i < n; ++i) {
73             if (i != 0)
74                 printf(" %d", Lx[i]);
75             else
76                 printf("%d", Lx[i]);
77             res += Lx[i];
78         }
79         puts("");
80         for (int i = 0; i < n; ++i) {
81             if (i != 0)
82                 printf(" %d", Ly[i]);
83             else
84                 printf("%d", Ly[i]);
85             res += Ly[i];
86         }
87         puts("");
88         printf("%d\n", res);
89     }
90     return 0;
91 }

```

3.11 LCA 倍增法

```

1 //倍增法預處理O(nlogn)，查詢O(logn)，
2 //利用lca找樹上任兩點距離
3 #define maxn 100005
4 struct Edge {
5     int u, v, w;
6 };
7 vector<vector<Edge>> G; // tree
8 int fa[maxn][31]; //fa[u][i] -> u的第2^i個祖先
9 long long dis[maxn][31];
10 int dep[maxn]; //深度
11 void dfs(int u, int p) { //預處理fa
12     fa[u][0] = p; //因為u的第2^0 = 1的祖先就是p
13     dep[u] = dep[p] + 1;
14     //第2^i的祖先是(第2^(i-1)個祖先)的
15     //第2^(i-1)的祖先
16     //ex: 第8個祖先是 (第4個祖先)的第4個祖先
17     for (int i = 1; i < 31; ++i) {
18         fa[u][i] = fa[fa[u][i-1]][i-1];
19         dis[u][i] = dis[fa[u][i-1]][i-1]
20             + dis[u][i-1];
21     }
22     //遍歷子節點
23     for (Edge& edge: G[u]) {
24         if (edge.v == p)
25             continue;
26         dis[edge.v][0] = edge.w;
27         dfs(edge.v, u);
28     }
29 }
30 long long lca(int x, int y) {
31     //此函數是找lca同時計算x、y的距離 -> dis(x,
32     //lca) + dis(lca, y)
33     //讓y比x深
34     if (dep[x] > dep[y])
35         swap(x, y);
36     int deltaDep = dep[y] - dep[x];
37     long long res = 0;
38     //讓y與x在同一個深度
39     for (int i = 0; deltaDep != 0; ++i,
40         deltaDep >>= 1)
41         if (deltaDep & 1)
42             res += dis[y][i], y = fa[y][i];
43     if (y == x) //x = y -> x、y彼此是彼此的祖先
44         return res;

```



```

42 //往上找，一起跳，但x、y不能重疊
43 for (int i = 30; i >= 0 && y != x; --i) {
44     if (fa[x][i] != fa[y][i]) {
45         res += dis[x][i] + dis[y][i];
46         x = fa[x][i];
47         y = fa[y][i];
48     }
49 }
50 //最後發現不能跳了，此時x的第2^0 =
    1個祖先(或說y的第2^0 =
    1的祖先)即為x、y的lca
51 res += dis[x][0] + dis[y][0];
52 return res;
53 }
54 int main() {
55     int n, q;
56     while (~scanf("%d", &n) && n) {
57         int v, w;
58         G.assign(n + 5, vector<Edge>());
59         for (int i = 1; i <= n - 1; ++i) {
60             scanf("%d %d", &v, &w);
61             G[i + 1].push_back({i + 1, v + 1, w});
62             G[v + 1].push_back({v + 1, i + 1, w});
63         }
64         dfs(1, 0);
65         scanf("%d", &q);
66         int u;
67         while (q--) {
68             scanf("%d %d", &u, &v);
69             printf("%lld%c", lca(u + 1, v +
70                 1), (q) ? ' ' : '\n');
71         }
72         return 0;
73 }

```

3.12 MCMF

```

1 #define maxn 225
2 #define INF 0x3f3f3f3f
3 struct Edge {
4     int u, v, cap, flow, cost;
5 };
6 //node size, edge size, source, target
7 int n, m, s, t;
8 vector<vector<int>>> G;
9 vector<Edge> edges;
10 bool inqueue[maxn];
11 long long dis[maxn];
12 int parent[maxn];
13 long long outFlow[maxn];
14 void addEdge(int u, int v, int cap, int
    cost) {
15     edges.emplace_back(Edge{u, v, cap, 0,
        cost});
16     edges.emplace_back(Edge{v, u, 0, 0,
        -cost});
17     m = edges.size();
18     G[u].emplace_back(m - 2);
19     G[v].emplace_back(m - 1);
20 }
21 //一邊求最短路的同時一邊MaxFlow
22 bool SPFA(long long& maxFlow, long long&
    minCost) {
23     // memset(outFlow, 0x3f,
        sizeof(outFlow));
24     memset(dis, 0x3f, sizeof(dis));
25     memset(inqueue, false, sizeof(inqueue));
26     queue<int> q;
27     q.push(s);
28     dis[s] = 0;
29     inqueue[s] = true;
30     outFlow[s] = INF;
31     while (!q.empty()) {
32         int u = q.front();
33         q.pop();
34         inqueue[u] = false;

```

```

35     for (const int edgeIndex: G[u]) {
36         const Edge& edge =
            edges[edgeIndex];
37         if ((edge.cap > edge.flow) &&
            (dis[edge.v] > dis[u] +
            edge.cost)) {
38             dis[edge.v] = dis[u] +
                edge.cost;
39             parent[edge.v] = edgeIndex;
40             outFlow[edge.v] =
                min(outFlow[u], (long
                long)(edge.cap -
                edge.flow));
41             if (!inqueue[edge.v]) {
42                 q.push(edge.v);
43                 inqueue[edge.v] = true;
44             }
45         }
46     }
47 }
48 //如果dis[t] > 0代表根本不賺還倒賠
49 if (dis[t] > 0)
50     return false;
51 maxFlow += outFlow[t];
52 minCost += dis[t] * outFlow[t];
53 //一路更新回去這次最短路流完後要維護的
54 //MaxFlow演算法相關(如反向邊等)
55 int curr = t;
56 while (curr != s) {
57     edges[parent[curr]].flow +=
        outFlow[t];
58     edges[parent[curr] ^ 1].flow -=
        outFlow[t];
59     curr = edges[parent[curr]].u;
60 }
61 return true;
62 }
63 long long MCMF() {
64     long long maxFlow = 0;
65     long long minCost = 0;
66     while (SPFA(maxFlow, minCost))
67         ;
68     return minCost;
69 }
70 int main() {
71     int T;
72     scanf("%d", &T);
73     for (int Case = 1; Case <= T; ++Case){
74         //總共幾個月，囤貨成本
75         int M, I;
76         scanf("%d %d", &M, &I);
77         //node size
78         n = M + M + 2;
79         G.assign(n + 5, vector<int>());
80         edges.clear();
81         s = 0;
82         t = M + M + 1;
83         for (int i = 1; i <= M; ++i) {
84             int produceCost, produceMax,
                sellPrice, sellMax,
                inventoryMonth;
85             scanf("%d %d %d %d %d",
                &produceCost, &produceMax,
                &sellPrice, &sellMax,
                &inventoryMonth);
86             addEdge(s, i, produceMax,
                produceCost);
87             addEdge(M + i, t, sellMax,
                -sellPrice);
88             for (int j = 0; j <=
                inventoryMonth; ++j) {
89                 if (i + j <= M)
90                     addEdge(i, M + i + j, INF,
91                         I * j);
92             }
93             printf("Case %d: %lld\n", Case,
                -MCMF());

```

```

94 }
95 return 0;
96 }

```

3.13 Dancing Links

```

1 struct DLX {
2     int seq, resSize;
3     int col[maxn], row[maxn];
4     int U[maxn], D[maxn], R[maxn], L[maxn];
5     int rowHead[maxn], colSize[maxn];
6     int result[maxn];
7     DLX(int r, int c) {
8         for(int i=0; i<=c; i++) {
9             L[i] = i-1, R[i] = i+1;
10            U[i] = D[i] = i;
11        }
12        L[R[seq]=0]=0=c;
13        resSize = -1;
14        memset(rowHead, 0, sizeof(rowHead));
15        memset(colSize, 0, sizeof(colSize));
16    }
17    void insert(int r, int c) {
18        row[++seq]=r, col[seq]=c,
        ++colSize[c];
19        U[seq]=c, D[seq]=D[c], U[D[c]]=seq,
        D[c]=seq;
20        if(rowHead[r]) {
21            L[seq]=rowHead[r],
22            R[seq]=R[rowHead[r]];
23            L[R[rowHead[r]]]=seq,
24            R[rowHead[r]]=seq;
25        } else {
26            rowHead[r] = L[seq] = R[seq] =
                seq;
27        }
28    }
29    void remove(int c) {
30        L[R[c]] = L[c], R[L[c]] = R[c];
31        for(int i=D[c]; i!=c; i=D[i]) {
32            for(int j=R[i]; j!=i; j=R[j]) {
33                U[D[j]] = U[j];
34                D[U[j]] = D[j];
35                --colSize[col[j]];
36            }
37        }
38    }
39    void recover(int c) {
40        for(int i=U[c]; i!=c; i=U[i]) {
41            for(int j=L[i]; j!=i; j=L[j]) {
42                U[D[j]] = D[U[j]] = j;
43                ++colSize[col[j]];
44            }
45        }
46        L[R[c]] = R[L[c]] = c;
47    }
48    bool dfs(int idx=0) { // 判斷其中一解版
49        if(R[0] == 0) {
50            resSize = idx;
51            return true;
52        }
53        int c = R[0];
54        for(int i=R[0]; i; i=R[i]) {
55            if(colSize[i] < colSize[c]) c = i;
56        }
57        remove(c);
58        for(int i=D[c]; i!=c; i=D[i]) {
59            result[idx] = row[i];
60            for(int j=R[i]; j!=i; j=R[j])
61                remove(col[j]);
62            if(dfs(idx+1)) return true;
63            for(int j=L[i]; j!=i; j=L[j])
64                recover(col[j]);
65        }
66        recover(c);
67        return false;
68    }
69 }

```

```

67 void dfs(int idx=0) { // 判斷最小 dfs
68     depth 版
69     if(R[0] == 0) {
70         resSize = min(resSize, idx); //
71         注意init值
72         return;
73     }
74     int c = R[0];
75     for(int i=R[0]; i; i=R[i]) {
76         if(colSize[i] < colSize[c]) c = i;
77     }
78     remove(c);
79     for(int i=D[c]; i!=c; i=D[i]) {
80         for(int j=R[i]; j!=i; j=R[j])
81             remove(col[j]);
82         dfs(idx+1);
83         for(int j=L[i]; j!=i; j=L[j])
84             recover(col[j]);
85     }
86     recover(c);
87 }

```

4 DataStructure

4.1 線段樹 1D

```

1 #define MAXN 1000
2 int data[MAXN]; //原數據
3 int st[4 * MAXN]; //線段樹
4 int tag[4 * MAXN]; //懶標
5 inline int pull(int l, int r) {
6     // 隨意自改變sum~max~min
7     // l~r是左右樹的index
8     return st[l] + st[r];
9 }
10 void build(int l, int r, int i) {
11     // 在[l, r]區間建樹，目前根的index為i
12     if (l == r) {
13         st[i] = data[l];
14         return;
15     }
16     int mid = l + ((r - l) >> 1);
17     build(l, mid, i * 2);
18     build(mid + 1, r, i * 2 + 1);
19     st[i] = pull(i * 2, i * 2 + 1);
20 }
21 int query(int ql, int qr, int l, int r, int i) {
22     // [ql, qr]是查詢區間，[l, r]是當前節點包含的區間
23     if (ql <= l && r <= qr)
24         return st[i];
25     int mid = l + ((r - l) >> 1);
26     if (tag[i]) {
27         //如果當前懶標有值則更新左右節點
28         st[i * 2] += tag[i] * (mid - l + 1);
29         st[i * 2 + 1] += tag[i] * (r - mid);
30         tag[i * 2] += tag[i]; //下傳懶標至左節點
31         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
32         tag[i] = 0;
33     }
34     int sum = 0;
35     if (ql <= mid)
36         sum += query(ql, qr, l, mid, i * 2);
37     if (qr > mid)
38         sum += query(ql, qr, mid + 1, r, i * 2 + 1);
39     return sum;
40 }
41 void update(int ql, int qr, int l, int r, int i, int c) {
42     // [ql, qr]是查詢區間，[l, r]是當前節點包含的區間
43     // c是變化量
44     if (ql <= l && r <= qr) {
45         st[i] += (r - l + 1) * c;
46         //求和,此需乘上區間長度
47         tag[i] += c;
48         return;
49     }

```

```

49     int mid = l + ((r - l) >> 1);
50     if (tag[i] && l != r) {
51         //如果當前懶標有值則更新左右節點
52         st[i * 2] += tag[i] * (mid - l + 1);
53         st[i * 2 + 1] += tag[i] * (r - mid);
54         tag[i * 2] += tag[i]; //下傳懶標至左節點
55         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
56         tag[i] = 0;
57     }
58     if (ql <= mid) update(ql, qr, l, mid, i * 2, c);
59     if (qr > mid) update(ql, qr, mid + 1, r, i * 2 + 1, c);
60     st[i] = pull(i * 2, i * 2 + 1);
61 }
62 //如果是直接改值而不是加值，query與update中的tag與st
63 //改值從+=改成=

```

4.2 線段樹 2D

```

1 //純2D segment tree 區間查詢單點修改最大最小值
2 #define maxn 2005 //500 * 4 + 5
3 int maxST[maxn][maxn], minST[maxn][maxn];
4 int N;
5 void modifyY(int index, int l, int r, int val, int yPos, int xIndex, bool xIsLeaf) {
6     if (l == r) {
7         if (xIsLeaf) {
8             maxST[xIndex][index] =
9             minST[xIndex][index] = val;
10            return;
11        }
12        maxST[xIndex][index] =
13        max(maxST[xIndex * 2][index],
14        maxST[xIndex * 2 + 1][index]);
15        minST[xIndex][index] =
16        min(minST[xIndex * 2][index],
17        minST[xIndex * 2 + 1][index]);
18    }
19    else {
20        int mid = (l + r) / 2;
21        if (yPos <= mid)
22            modifyY(index * 2, l, mid, val, yPos, xIndex, xIsLeaf);
23        else
24            modifyY(index * 2 + 1, mid + 1, r, val, yPos, xIndex, xIsLeaf);
25    }
26    maxST[xIndex][index] =
27    max(maxST[xIndex][index * 2],
28    maxST[xIndex][index * 2 + 1]);
29    minST[xIndex][index] =
30    min(minST[xIndex][index * 2],
31    minST[xIndex][index * 2 + 1]);
32 }
33 void modifyX(int index, int l, int r, int val, int xPos, int yPos) {
34     if (l == r) {
35         modifyY(1, 1, N, val, yPos, index, true);
36     }
37     else {
38         int mid = (l + r) / 2;
39         if (xPos <= mid)
40             modifyX(index * 2, l, mid, val, xPos, yPos);
41         else
42             modifyX(index * 2 + 1, mid + 1, r, val, xPos, yPos);
43     }
44 }

```

```

38 void queryY(int index, int l, int r, int yql, int yqr, int xIndex, int& vmax, int& vmin) {
39     if (yql <= l && r <= yqr) {
40         vmax = max(vmax,
41             maxST[xIndex][index]);
42         vmin = min(vmin,
43             minST[xIndex][index]);
44     }
45     else {
46         int mid = (l + r) / 2;
47         if (yql <= mid)
48             queryY(index * 2, l, mid, yql, yqr, xIndex, vmax, vmin);
49         if (mid < yqr)
50             queryY(index * 2 + 1, mid + 1, r, yql, yqr, xIndex, vmax, vmin);
51     }
52 }
53 void queryX(int index, int l, int r, int xql, int xqr, int yql, int yqr, int& vmax, int& vmin) {
54     if (xql <= l && r <= xqr) {
55         queryY(1, 1, N, yql, yqr, index, vmax, vmin);
56     }
57     else {
58         int mid = (l + r) / 2;
59         if (xql <= mid)
60             queryX(index * 2, l, mid, xql, xqr, yql, yqr, vmax, vmin);
61         if (mid < xqr)
62             queryX(index * 2 + 1, mid + 1, r, xql, xqr, yql, yqr, vmax, vmin);
63     }
64 }
65 int main() {
66     while (scanf("%d", &N) != EOF) {
67         int val;
68         for (int i = 1; i <= N; ++i) {
69             for (int j = 1; j <= N; ++j) {
70                 scanf("%d", &val);
71                 modifyX(1, 1, N, val, i, j);
72             }
73         }
74         int q;
75         int vmax, vmin;
76         int xql, xqr, yql, yqr;
77         char op;
78         scanf("%d", &q);
79         while (q--) {
80             getchar(); //for \n
81             scanf("%c", &op);
82             if (op == 'q') {
83                 scanf("%d %d %d %d", &xql, &yql, &xqr, &yqr);
84                 vmax = -0x3f3f3f3f;
85                 vmin = 0x3f3f3f3f;
86                 queryX(1, 1, N, xql, xqr, yql, yqr, vmax, vmin);
87                 printf("%d %d\n", vmax, vmin);
88             }
89             else {
90                 scanf("%d %d %d", &xql, &yql, &val);
91                 modifyX(1, 1, N, val, xql, yql);
92             }
93         }
94     }
95     return 0;

```

4.3 權值線段樹

```

1 //權值線段樹 + 離散化 解決區間第k小問題
2 //其他網路上的解法: 2個heap, Treap, AVL tree
3 #define maxn 30005
4 int nums[maxn];
5 int getArr[maxn];
6 int id[maxn];
7 int st[maxn << 2];
8 void update(int index, int l, int r, int qx)
9 {
10     if (l == r)
11     {
12         ++st[index];
13         return;
14     }
15     int mid = (l + r) / 2;
16     if (qx <= mid)
17         update(index * 2, l, mid, qx);
18     else
19         update(index * 2 + 1, mid + 1, r, qx);
20     st[index] = st[index * 2] + st[index * 2
21         + 1];
22 }
23 //找區間第k個小的
24 int query(int index, int l, int r, int k) {
25     if (l == r)
26         return id[l];
27     int mid = (l + r) / 2;
28     //k比左子樹小
29     if (k <= st[index * 2])
30         return query(index * 2, l, mid, k);
31     else
32         return query(index * 2 + 1, mid + 1,
33             r, k - st[index * 2]);
34 }
35 int main() {
36     int t;
37     cin >> t;
38     bool first = true;
39     while (t--) {
40         if (first)
41             first = false;
42         else
43             puts("");
44         memset(st, 0, sizeof(st));
45         int m, n;
46         cin >> m >> n;
47         for (int i = 1; i <= m; ++i) {
48             cin >> nums[i];
49             id[i] = nums[i];
50         }
51         for (int i = 0; i < n; ++i)
52             cin >> getArr[i];
53         //離散化
54         //防止m == 0
55         if (m)
56             sort(id + 1, id + m + 1);
57         int stSize = unique(id + 1, id + m +
58             1) - (id + 1);
59         for (int i = 1; i <= m; ++i) {
60             nums[i] = lower_bound(id + 1, id
61                 + stSize + 1, nums[i]) - id;
62         }
63         int addCount = 0;
64         int getCount = 0;
65         int k = 1;
66         while (getCount < n) {
67             if (getArr[getCount] == addCount)
68                 printf("%d\n", query(1, 1,
69                     stSize, k));
70             ++k;
71             ++getCount;
72         }
73     }
74     else {

```

```

69         update(1, 1, stSize,
70             nums[addCount + 1]);
71         ++addCount;
72     }
73 }
74 return 0;
75 }

```

4.4 Trie

```

1 const int maxn = 300000 + 10;
2 const int mod = 20071027;
3 int dp[maxn];
4 int mp[4000*100 + 10][26];
5 char str[maxn];
6 struct Trie {
7     int seq;
8     int val[maxn];
9     Trie() {
10         seq = 0;
11         memset(val, 0, sizeof(val));
12         memset(mp, 0, sizeof(mp));
13     }
14     void insert(char* s, int len) {
15         int r = 0;
16         for (int i = 0; i < len; i++) {
17             int c = s[i] - 'a';
18             if (!mp[r][c]) mp[r][c] = ++seq;
19             r = mp[r][c];
20         }
21         val[r] = len;
22         return;
23     }
24     int find(int idx, int len) {
25         int result = 0;
26         for (int r = 0; idx < len; idx++) {
27             int c = str[idx] - 'a';
28             if (!(r = mp[r][c])) return result;
29             if (val[r])
30                 result = (result + dp[idx +
31                     1]) % mod;
32         }
33         return result;
34     };
35 }
36 int main() {
37     int n, tc = 1;
38     while (~scanf("%s", str, &n)) {
39         Trie tr;
40         int len = strlen(str);
41         char word[100+10];
42         memset(dp, 0, sizeof(dp));
43         dp[len] = 1;
44         while (n--) {
45             scanf("%s", word);
46             tr.insert(word, strlen(word));
47         }
48         for (int i = len - 1; i >= 0; i--)
49             dp[i] = tr.find(i, len);
50         printf("Case %d: %d\n", tc++, dp[0]);
51     }
52     return 0;
53 }
54 //****Input****
55 * abcd
56 * 4
57 * a b cd ab
58 *****
59 *****Output***
60 * Case 1: 2
61 *****//

```

4.5 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以退役了。"
3
4 example
5
6 給出一個長度為 n 的數組，
7 輸出每 k 個連續的數中的最大值和最小值。
8
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14 //得到這個隊列裡的最小值，直接找到最後的就行了
15 void getmin() {
16     int head=0, tail=0;
17     for (int i=1; i<k; i++) {
18         while (head<=tail && a[q[tail]]>=a[i])
19             tail--;
20         q[++tail]=i;
21     }
22     for (int i=k; i<=n; i++) {
23         while (head<=tail && a[q[tail]]>=a[i])
24             tail--;
25         q[++tail]=i;
26         while (q[head]<=i-k) head++;
27         cout<<a[q[head]]<<" ";
28     }
29     cout<<endl;
30 }
31 //和上面同理
32 void getmax() {
33     int head=0, tail=0;
34     for (int i=1; i<k; i++) {
35         while (head<=tail && a[q[tail]]<=a[i]) tail--;
36         q[++tail]=i;
37     }
38     for (int i=k; i<=n; i++) {
39         while (head<=tail && a[q[tail]]<=a[i]) tail--;
40         q[++tail]=i;
41         while (q[head]<=i-k) head++;
42         cout<<a[q[head]]<<" ";
43     }
44     cout<<endl;
45 }
46 int main() {
47     cin >> n >> k; //每k個連續的數
48     for (int i=1; i<=n; i++) cin >> a[i];
49     getmin();
50     getmax();
51     return 0;
52 }

```

5 geometry

5.1 intersection

```

1 using LL = long long;
2
3 struct Point2D {
4     LL x, y;
5 };
6
7 struct Line2D {
8     Point2D s, e;
9     LL a, b, c; // L: ax + by = c
10     Line2D(Point2D s, Point2D e): s(s), e(e)
11     {
12         a = e.y - s.y;
13         b = s.x - e.x;
14         c = a * s.x + b * s.y;
15     }
16 };
17 // 用克拉馬公式求二元一次解
18 Point2D intersection2D(Line2D l1, Line2D l2)
19 {
20     LL D = l1.a * l2.b - l2.a * l1.b;

```



```

20 LL Dx = l1.c * l2.b - l2.c * l1.b;
21 LL Dy = l1.a * l2.c - l2.a * l1.c;
22
23 if(D) { // intersection
24     double x = 1.0 * Dx / D;
25     double y = 1.0 * Dy / D;
26 } else {
27     if(Dx || Dy) // Parallel lines
28     else // Same line
29 }
30 }

```

5.2 半平面相交

```

1 // Q: 給定一張凸包(已排序的點),
2 // 找出圖中離凸包外最遠的距離
3
4 const int maxn = 100 + 10;
5 const double eps = 1e-7;
6
7 struct Vector {
8     double x, y;
9     Vector(double x=0.0, double y=0.0):
10         x(x), y(y) {}
11
12     Vector operator+(Vector v) {
13         return Vector(x+v.x, y+v.y);
14     }
15     Vector operator-(Vector v) {
16         return Vector(x-v.x, y-v.y);
17     }
18     Vector operator*(double val) {
19         return Vector(x*val, y*val);
20     }
21     double dot(Vector v) { return x*v.x + y*v.y; }
22     double cross(Vector v) { return x*v.y - y*v.x; }
23     double length() { return sqrt(dot(*this)); }
24     Vector unit_normal_vector() {
25         double len = length();
26         return Vector(-y/len, x/len);
27     }
28 };
29 using Point = Vector;
30
31 struct Line {
32     Point p;
33     Vector v;
34     double ang;
35     Line(Point p={}, Vector v={}): p(p),
36         v(v) {
37         ang = atan2(v.y, v.x);
38     }
39     bool operator<(const Line& l) const {
40         return ang < l.ang;
41     }
42     Point intersection(Line l) {
43         Vector u = p - l.p;
44         double t = l.v.cross(u) / v.cross(l.v);
45         return p + v*t;
46     }
47 };
48 int n, m;
49 Line narrow[maxn]; // 要判斷的直線
50 Point poly[maxn]; // 能形成半平面交的凸包邊界點
51
52 // return true if point p is on the left of line l
53 bool onLeft(Point p, Line l) {
54     return l.v.cross(p-l.p) > 0;
55 }

```

```

56
57 int halfplaneIntersection() {
58     int l, r;
59     Line L[maxn]; // 排序後的向量隊列
60     Point P[maxn]; // s[i] 跟 s[i-1] 的交點
61
62     L[l=r=0] = narrow[0]; // notice: narrow is sorted
63     for(int i=1; i<n; i++) {
64         while(l<r && !onLeft(P[r-1], narrow[i])) r--;
65         while(l<r && !onLeft(P[l], narrow[i])) l++;
66
67         L[++r] = narrow[i];
68         if(l < r) P[r-1] = L[r-1].intersection(L[r]);
69     }
70
71     while(l<r && !onLeft(P[r-1], L[l])) r--;
72     if(r-l <= 1) return 0;
73
74     P[r] = L[r].intersection(L[l]);
75
76     int m=0;
77     for(int i=l; i<=r; i++) {
78         poly[m++] = P[i];
79     }
80     return m;
81 }
82
83 Point pt[maxn];
84 Vector vec[maxn];
85 Vector normal[maxn]; // normal[i] = vec[i] 的單位法向量
86
87 double bsearch(double l=0.0, double r=1e4) {
88     if(abs(r-l) < eps) return l;
89
90     double mid = (l + r) / 2;
91
92     for(int i=0; i<n; i++) {
93         narrow[i] = Line(pt[i]+normal[i]*mid, vec[i]);
94     }
95
96     if(halfplaneIntersection())
97         return bsearch(mid, r);
98     else return bsearch(l, mid);
99 }
100
101 int main() {
102     while(~scanf("%d", &n) && n) {
103         for(int i=0; i<n; i++) {
104             double x, y;
105             scanf("%lf%lf", &x, &y);
106             pt[i] = {x, y};
107         }
108         for(int i=0; i<n; i++) {
109             vec[i] = pt[(i+1)%n] - pt[i];
110             normal[i] = vec[i].unit_normal_vector();
111         }
112
113         printf("%.6lf\n", bsearch());
114     }
115     return 0;
116 }
117 }

```

5.3 凸包

```

1 //Q: 平面上給定多個區域, 由多個座標點所形成, 再給定
2 //多點(x,y), 判斷有落點的區域(destroyed)的面積總和。
3 const int maxn = 500 + 10;
4 const int maxCoordinate = 500 + 10;

```

```

5 struct Point {
6     int x, y;
7 };
8 int n;
9 bool destroyed[maxn];
10 Point arr[maxn];
11 vector<Point> polygons[maxn];
12 void scanAndSortPoints() {
13     int minX = maxCoordinate, minY = maxCoordinate;
14     for(int i=0; i<n; i++) {
15         int x, y;
16         scanf("%d%d", &x, &y);
17         arr[i] = (Point){x, y};
18         if(y < minY || (y == minY && x < minX)) {
19             // If there are floating points, use:
20             // if(y<minY || (abs(y-minY)<eps && x<minX)) {
21             minX = x, minY = y;
22         }
23     }
24     sort(arr, arr+n, [minX, minY](Point& a, Point& b){
25         double theta1 = atan2(a.y - minY, a.x - minX);
26         double theta2 = atan2(b.y - minY, b.x - minX);
27         return theta1 < theta2;
28     });
29     return;
30 }
31
32 // returns cross product of u(AB) x v(AC)
33 int cross(Point& A, Point& B, Point& C) {
34     int u[2] = {B.x - A.x, B.y - A.y};
35     int v[2] = {C.x - A.x, C.y - A.y};
36     return (u[0] * v[1]) - (u[1] * v[0]);
37 }
38
39 // size of arr = n >= 3
40 // st = the stack using vector, m = index of the top
41 vector<Point> convex_hull() {
42     vector<Point> st(arr, arr+3);
43     for(int i=3, m=2; i<n; i++, m++) {
44         while(m >= 2) {
45             if(cross(st[m], st[m-1], arr[i]) < 0)
46                 break;
47             st.pop_back();
48             m--;
49         }
50         st.push_back(arr[i]);
51     }
52     return st;
53 }
54
55 bool inPolygon(vector<Point>& vec, Point p) {
56     vec.push_back(vec[0]);
57     for(int i=1; i<vec.size(); i++) {
58         if(cross(vec[i-1], vec[i], p) < 0) {
59             vec.pop_back();
60             return false;
61         }
62     }
63     vec.pop_back();
64     return true;
65 }
66
67 1 | x1 x2 x3 x4 x5 xn |
68 A = -- | x x x x x ... x |
69 2 | y1 y2 y3 y4 y5 yn |
70 double calculateArea(vector<Point>& v) {
71     v.push_back(v[0]); // make v[n] = v[0]
72     double result = 0.0;
73     for(int i=1; i<v.size(); i++)
74         result +=

```

```

75     v[i-1].x*v[i].y - v[i-1].y*v[i].x;
76     v.pop_back();
77     return result / 2.0;
78 }
79
80 int main() {
81     int p = 0;
82     while(~scanf("%d", &n) && (n != -1)) {
83         scanAndSortPoints();
84         polygons[p++] = convex_hull();
85     }
86     int x, y;
87     double result = 0.0;
88     while(~scanf("%d%d", &x, &y))
89         for(int i=0; i<p; i++)
90             if(inPolygon(polygons[i],
91                           (Point){x, y}))
92                 destroyed[i] = true;
93     for(int i=0; i<p; i++)
94         if(destroyed[i])
95             result +=
96                 calculateArea(polygons[i]);
97     printf("%.2lf\n", result);
98     return 0;
99 }

```

6 DP

6.1 抽屜

```

1 long long dp[70][70][2];
2 // 初始條件
3 dp[1][0][0] = dp[1][1][1] = 1;
4 for (int i = 2; i <= 66; ++i){
5     // i個抽屜0個安全且上方0 =
6     // (底下i - 1個抽屜且1個安全且最上面L) +
7     // (底下n - 1個抽屜0個安全且最上方為0)
8     dp[i][0][0]=dp[i-1][1][1]+dp[i-1][0][0];
9     for (int j = 1; j <= i; ++j) {
10         dp[i][j][0] =
11             dp[i-1][j+1][1]+dp[i-1][j][0];
12         dp[i][j][1] =
13             dp[i-1][j-1][1]+dp[i-1][j-1][0];
14     }
15 } //答案在 dp[n][s][0] + dp[n][s][1];

```

6.2 Deque 最大差距

```

1 /*定義dp[l][r]是l ~ r時與先手最大差異值
2 轉移式: dp[l][r] = max{a[l] - solve(l + 1,
3     r), a[r] - solve(l, r - 1)}
4 裡面用減的主要是因為求的是相減且會一直換手，
5 所以正負正負...*/
6 #define maxn 3005
7 bool vis[maxn][maxn];
8 long long dp[maxn][maxn];
9 long long a[maxn];
10 long long solve(int l, int r) {
11     if (l > r) return 0;
12     if (vis[l][r]) return dp[l][r];
13     vis[l][r] = true;
14     long long res = a[l] - solve(l + 1, r);
15     res = max(res, a[r] - solve(l, r - 1));
16     return dp[l][r] = res;
17 }
18 int main() {
19     ...
20     printf("%lld\n", solve(1, n));
21 }

```

6.3 LCS 和 LIS

```

1 //LCS 和 LIS 題目轉換
2 LIS 轉成 LCS
3 1. A 為原序列， B=sort(A)

```

```

4 2. 對 A,B 做 LCS
5 LCS 轉成 LIS
6 1. A, B 為原本的兩序列
7 2. 最 A 序列作編號轉換，將轉換規則套用在 B
8 3. 對 B 做 LIS
9 4. 重複的數字在編號轉換時後要變成不同的數字，
10 越早出現的數字要越小
11 5. 如果有數字在 B 裡面而不在 A 裡面，
12 直接忽略這個數字不做轉換即可

```

6.4 RangedP

```

1 //區間dp
2 int dp[55][55];
3 // dp[i][j] -> [i,j] 切割區間中最小的cost
4 int cuts[55];
5 int solve(int i, int j) {
6     if (dp[i][j] != -1)
7         return dp[i][j];
8     //代表沒有其他切法，只能是cuts[j] - cuts[i]
9     if (i == j - 1)
10         return dp[i][j] = 0;
11     int cost = 0x3f3f3f3f;
12     for (int m = i + 1; m < j; ++m) {
13         //枚舉區間中間切點
14         cost = min(cost, solve(i, m) +
15                     solve(m, j) + cuts[j] - cuts[i]);
16     }
17     return dp[i][j] = cost;
18 }
19 int main() {
20     int l, n;
21     while (scanf("%d", &l) != EOF && l){
22         scanf("%d", &n);
23         for (int i = 1; i <= n; ++i)
24             scanf("%d", &cuts[i]);
25         cuts[0] = 0;
26         cuts[n + 1] = 1;
27         memset(dp, -1, sizeof(dp));
28         printf("ans = %d.\n", solve(0, n+1));
29     }
30     return 0;
31 }

```

6.5 stringDP

Edit distance S_1 最少需要經過幾次增、刪或換字變成 S_2

$$dp[i][j] = \begin{cases} \begin{matrix} i+1 & \text{if } j = -1 \\ j+1 & \text{if } i = -1 \\ \min \left\{ \begin{matrix} dp[i-1][j-1] & \text{if } S_1[i] = S_2[j] \\ dp[i-1][j] & \text{if } S_1[i] \neq S_2[j] \end{matrix} \right\} & \end{matrix} + 1 \end{cases}$$

Longest Palindromic Subsequence

$$dp[l][r] = \begin{cases} 1 & \text{if } l = r \\ \max \{ dp[l+1][r-1], dp[l][r-1] \} & \text{if } S[l] = S[r] \\ \max \{ dp[l+1][r], dp[l][r-1] \} & \text{if } S[l] \neq S[r] \end{cases}$$

6.6 樹 DP 有幾個 path 長度為 k

```

1 #define maxn 50005
2 #define maxk 505
3 //dp[u][u的child距距離u長度k的數量]
4 long long dp[maxn][maxk];
5 vector<vector<int>> G;
6 int n, k;
7 long long res = 0;
8 void dfs(int u, int p) {
9     //u自己
10     dp[u][0] = 1;
11     for (int v: G[u]) {
12         if (v == p)
13             continue;

```

```

14     dfs(v, u);
15     for (int i = 1; i <= k; ++i) {
16         //子樹v距離i - 1的等於對於u來說距離i的
17         dp[u][i] += dp[v][i - 1];
18     }
19 }
20 //統計在u子樹中距離u為k的數量
21 res += dp[u][k];
22 long long cnt = 0;
23 for (int v: G[u]) {
24     if (v == p)
25         continue; //重點算法
26     for (int x = 0; x <= k - 2; ++x) {
27         cnt +=
28             dp[v][x]*(dp[u][k-x-1]-dp[v][k-x-2]);
29     }
30 }
31 res += cnt / 2;
32 }
33 int main() {
34     ...
35     dfs(1, -1);
36     printf("%lld\n", res);
37     return 0;
38 }

```

6.7 TreeDP reroot

```

1 /*re-root dp on tree O(n + n + n) -> O(n)*/
2 class Solution {
3 public:
4     vector<int> sumOfDistancesInTree(int n,
5         vector<vector<int>>& edges) {
6         this->res.assign(n, 0);
7         G.assign(n + 5, vector<int>());
8         for (vector<int>& edge: edges) {
9             G[edge[0]].emplace_back(edge[1]);
10             G[edge[1]].emplace_back(edge[0]);
11         }
12         memset(this->visited, 0,
13             sizeof(this->visited));
14         this->dfs(0);
15         memset(this->visited, 0,
16             sizeof(this->visited));
17         this->res[0] = this->dfs2(0, 0);
18         memset(this->visited, 0,
19             sizeof(this->visited));
20         this->dfs3(0, n);
21         return this->res;
22     }
23 private:
24     vector<vector<int>> G;
25     bool visited[30005];
26     int subtreeSize[30005];
27     vector<int> res;
28     //求subtreeSize
29     int dfs(int u) {
30         this->visited[u] = true;
31         for (int v: this->G[u])
32             if (!this->visited[v])
33                 this->subtreeSize[u] +=
34                     this->dfs(v);
35         //自己
36         this->subtreeSize[u] += 1;
37         return this->subtreeSize[u];
38     }
39     //求res[0], 0到所有點的距離
40     int dfs2(int u, int dis) {
41         this->visited[u] = true;
42         int sum = 0;
43         for (int v: this->G[u])
44             if (!visited[v])
45                 sum += this->dfs2(v, dis + 1);
46         //要加上自己的距離
47         return sum + dis;
48     }
49     //算出所有的res

```

```

45 void dfs3(int u, int n) {
46     this->visited[u] = true;
47     for (int v: this->G[u]) {
48         if (!visited[v]) {
49             this->res[v] = this->res[u] +
                    n - 2 *
                    this->subtreeSize[v];
50             this->dfs3(v, n);
51         }
52     }
53 }
54 };

```

6.8 WeightedLIS

```

1 #define maxn 200005
2 long long dp[maxn];
3 long long height[maxn];
4 long long B[maxn];
5 long long st[maxn << 2];
6 void update(int p, int index, int l, int r,
            long long v) {
7     if (l == r) {
8         st[index] = v;
9         return;
10    }
11    int mid = (l + r) >> 1;
12    if (p <= mid)
13        update(p, (index << 1), l, mid, v);
14    else
15        update(p, (index << 1)+1, mid+1, r, v);
16    st[index] =
17        max(st[index<<1], st[(index<<1)+1]);
18 }
19 long long query(int index, int l, int r, int
            ql, int qr) {
20     if (ql <= l && r <= qr)
21         return st[index];
22     int mid = (l + r) >> 1;
23     long long res = -1;
24     if (ql <= mid)
25         res =
26             max(res, query(index<<1, l, mid, ql, qr));
27     if (mid < qr)
28         res =
29             max(res, query((index<<1)+1, mid+1, r, ql, qr));
30     return res;
31 }
32 int main() {
33     int n;
34     scanf("%d", &n);
35     for (int i = 1; i <= n; ++i)
36         scanf("%lld", &height[i]);
37     for (int i = 1; i <= n; ++i)
38         scanf("%lld", &B[i]);
39     long long res = B[1];
40     update(height[1], 1, 1, n, B[1]);
41     for (int i = 2; i <= n; ++i) {
42         long long temp;
43         if (height[i] - 1 >= 1)
44             temp =
45                 B[i]+query(1, 1, n, 1, height[i]-1);
46         else
47             temp = B[i];
48         update(height[i], 1, 1, n, temp);
49         res = max(res, temp);
50     }
51     printf("%lld\n", res);
52     return 0;
53 }

```