

Contents

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

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

```
1 const int maxn = 1e6 + 10;
2
3 int n, m; // len(a), len(b)
4 int f[maxn]; // failure function
5 char a[maxn], b[maxn];
6
7 void failureFuntion() { // f[0] = 0
8     for(int i=1, j=0; i<m; ) {
9         if(b[i] == b[j]) f[i++] = ++j;
10        else if(j) j = f[j-1];
11        else f[i++] = 0;
12    }
13 }
14
15 int kmp() {
16     int i = 0, j = 0, res = 0;
17     while(i < n) {
18         if(a[i] == b[j]) i++, j++;
19         else if(j) j = f[j-1];
20         else i++;
21         if(j == m) {
22             res++; // 找到答案
23             j = 0; // non-overlapping
24         }
25     }
26     return res;
27 }
28
29 // Problem: 所有在b裡，前後綴相同的長度
30 // b = ababcababababcabab
31 // f = 001201234123456789
32 // 前9 = 後9
33 // 前4 = 前9的後4 = 後4
34 // 前2 = 前4的後2 = 前9的後2 = 後2
35 for(int j=m; j; j=f[j-1]) {
36     // j 是答案
37 }
```

1.3 Z Algorithm

```
1 const int maxn = 1e6 + 10;
2
3 int z[maxn]; // s[0:z[i]] = s[i:z[i]]
4 string s;
5
6 void makeZ() { // z[0] = 0
7     for(int i=1, l=0, r=0; i<s.length(); i++) {
8         if(i<=r && z[i-l]<=r-i+1) z[i] = z[i-l];
9         else {
10            z[i] = max(0, r-i+1);
11            while(i+z[i]<=s.length() &&
12                s[z[i]]==s[i+z[i]]) z[i]++;
13        }
14        if(i+z[i]-1 > r) l = i, r = i+z[i]-1;
15    }
```

2 math

2.1 公式

- 最多因數數

Range	最多因數數	因數個數
10^9	735134400	1344
2^{31}	2095133040	1600
10^{18}	897612484786617600	103680
2^{64}	9200527969062830400	161280

2. Faulhaber's formula

$$\sum_{k=1}^n k^p = \frac{1}{p+1} \sum_{r=0}^p \binom{p+1}{r} B_r n^{p-r+1}$$

where $B_0 = 1, B_r = 1 - \sum_{i=0}^{r-1} \binom{r}{i} \frac{B_i}{r-i+1}$

也可用高斯消去法找 $deg(p+1)$ 的多項式，例：

$$\sum_{k=1}^n k^2 = a_3 n^3 + a_2 n^2 + a_1 n + a_0$$

$$\begin{bmatrix} 0^3 & 0^2 & 0^1 & 0^0 \\ 1^3 & 1^2 & 1^1 & 1^0 \\ 2^3 & 2^2 & 2^1 & 2^0 \\ 3^3 & 3^2 & 3^1 & 3^0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0^2 & 0^2 & 0^2 & 0^2 \\ 0^2 + 1^2 & 0^2 + 1^2 & 0^2 + 1^2 & 0^2 + 1^2 \\ 0^2 + 1^2 + 2^2 & 0^2 + 1^2 + 2^2 & 0^2 + 1^2 + 2^2 & 0^2 + 1^2 + 2^2 \\ 0^2 + 1^2 + 2^2 + 3^2 & 0^2 + 1^2 + 2^2 + 3^2 & 0^2 + 1^2 + 2^2 + 3^2 & 0^2 + 1^2 + 2^2 + 3^2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 & 5 \\ 27 & 9 & 3 & 1 & 14 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 4 & 6 & 7 & 3 \\ 0 & 0 & 6 & 11 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}, \sum_{k=1}^n k^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

3. SG 函式

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

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

4. Fibonacci

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

5. 皮克定理

給定頂點座標均是整點（或正方形格子點）的簡單多邊形，其面積 A 和內部格點數目 i 、邊上格點數目 b 的關係為

$$A = i + \frac{b}{2} - 1$$

6. 連通平面圖的歐拉定理

對於有 V 個點、 E 條邊、 F 個面的連通平面圖

$$F + V - E = 2$$

2.2 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(),
7         vector<ll>(B[0].size()));
8     for(int i=0; i<A.size(); i++) {
9         for(int j=0; j<B[0].size(); j++) {
10            for(int k=0; k<B.size(); k++) {
11                res[i][j] += A[i][k] *
12                    B[k][j] % mod;
13                res[i][j] %= mod;
14            }
15        }
16    }
17    return res;
18 }
19 mat I = ;
20 // compute matrix M^n
21 // 需先 init I 矩陣
22 mat mpow(mat& M, int n) {
23     if(n <= 1) return n ? M : I;
24     mat v = mpow(M, n>>1);
25     return (n & 1) ? v*v*M : v*v;
26 }
```

```

27 // 迴圈版本
28 mat mpow(mat M, int n) {
29     mat res(M.size(),
30         vector<ll>(M[0].size()));
31     for(int i=0; i<res.size(); i++)
32         res[i][i] = 1;
33     for(; n; n>=1) {
34         if(n & 1) res = res * M;
35         M = M * M;
36     }
37     return res;
38 }

```

2.3 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≤N≤106) 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!=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 }

```

2.4 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-ans/i;
8             while(n%i==0) n/=i;
9         }
10     if(n>1) ans=ans-ans/n;
11     return ans;
12 }

```

2.5 乘法逆元、組合數

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

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

若 $p \in \text{prime}$ ，根據費馬小定理，則

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

```

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 // 前提：mod 為質數
17 void build() {
18     auto qpow = [&](ll a, int b) {
19         ll res = 1;
20         for(; b; b>=1) {
21             if(b & 1) res = res * a % mod;
22             a = a * a % mod;
23         }
24         return res;
25     };
26
27     for(int x=2; x<maxn; x++) {
28         fact[x] = (ll)x * fact[x-1] % mod;
29         invFact[x] = qpow(fact[x], mod-2);
30     }
31 }
32
33 // C(a, b) % mod
34 int comb(int a, int b) {
35     if(a < b) return 0;
36     ll x = fact[a];
37     ll y = (ll)invFact[b] * invFact[a-b] % mod;
38     return x * y % mod;
39 }
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 可用三分搜找二次函數最小值。
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     Point operator+(const Point &rhs) const{
18         return Point(x+rhs.x,y+rhs.y,z+rhs.z);
19     }
20     Point operator-(const Point &rhs) const{
21         return Point(x-rhs.x,y-rhs.y,z-rhs.z);
22     }
23     Point operator*(const double &d) const{
24         return Point(x*d,y*d,z*d);
25     }
26     Point operator/(const double &d) const{
27         return Point(x/d,y/d,z/d);
28     }
29     double dist(const Point &rhs) const{
30         double res = 0;
31         res+=(x-rhs.x)*(x-rhs.x);
32         res+=(y-rhs.y)*(y-rhs.y);
33         res+=(z-rhs.z)*(z-rhs.z);
34         return res;
35     }
36 };
37 int main(){
38     IOS; //輸入優化
39     int T;
40     cin>>T;
41     for(int ti=1;ti<=T;++ti){
42         double time;
43         Point x1,y1,d1,x2,y2,d2;
44         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) <<
        sqrt(ans) << '\n';
58 }
59 }

```

3.2 差分

用途：在區間 $[l, r]$ 加上一個數字 v 。

$b[1] += v$; ($b[0 \sim 1]$ 加上 v)

$b[r+1] -= v$; ($b[r+1 \sim n]$ 減去 v ($b[r]$ 仍保留 v))

給的 $a[]$ 是前綴和數列，建構 $b[]$ ，

因為 $a[i] = b[0] + b[1] + b[2] + \dots + b[i]$ ，

所以 $b[i] = a[i] - a[i-1]$ 。

在 $b[1]$ 加上 v ， $b[r+1]$ 減去 v ，

最後再從 0 跑到 n 使 $b[i] += b[i-1]$ 。

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

刪數字問題

//problem

給定一個數字 $N(\leq 10^{100})$ ，需要刪除 K 個數字，

請問刪除 K 個數字後最小的數字為何？

//solution

刪除滿足第 i 位數大於第 $i+1$ 位數的最左邊第 i 位數，

扣除高位數的影響較扣除低位數的大。

//code

```

9 int main(){
10     string s;
11     int k;
12     cin>>s>>k;
13     for(int i=0;i<k;++i){
14         if((int)s.size()==0) break;
15         int pos =(int)s.size()-1;
16         for(int j=0;j<(int)s.size()-1;++j){
17             if(s[j]>s[j+1]){
18                 pos=j;
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批序，
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 依照右界由小到大批序，
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         }
93         sort(a, a+n);
94         int ans=1, R=a[0].R;
95         for(int i=1; i<n; ++i){
96             if(a[i].L>=R){
97                 ++ans;
98                 R=a[i].R;
99             }
100         }
101         cout<<ans<<'\n';

```

```

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  $\max Li$  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  $\rightarrow$  烏龜重量
148 工作期限  $\rightarrow$  烏龜可承受重量
149 多少工作不延期  $\rightarrow$  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 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
    單位懲罰，
182 請問最少會受到多少單位懲罰。
183 //solution
184 依照懲罰由大到小排序，
185 每項工作依序嘗試可不可以放在
    Di-Ti+1, Di-Ti, ..., 1, 0，
186 如果有空間就放進去，否則延後執行。
187 //problem
188 給定 N 項工作，每項工作的需要處理時長為 Ti，
189 期限是 Di，如果第 i 項工作在期限內完成會獲得 ai
    單位獎勵，
190 請問最多會獲得多少單位獎勵。
191 //solution
192 和上題相似，這題變成依照獎勵由大到小排序。
193 //code
194 struct Work{
195     int d,p;
196     bool operator<(const Work &rhs)const{
197         return p>rhs.p;
198     }
199 };
200 int main(){
201     int n;
202     Work a[100005];
203     bitset<100005> ok;
204     while(cin>>n){
205         ok.reset();
206         for(int i=0;i<n;++i)
207             cin>>a[i].d>>a[i].p;
208         sort(a,a+n);
209         int ans=0;
210         for(int i=0;i<n;++i){
211             int j=a[i].d;
212             while(j--){
213                 if(!ok[j]){
214                     ans+=a[i].p;
215                     ok[j]=true;
216                     break;
217                 }
218             }
219         }
220         cout<<ans<<'\\n';
221     }
222 }

```

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 &&
                e.cap>e.flow) {
32                 level[e.t] = level[e.s] + 1;
33                 q.push(e.t);
34             }
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()
        && lim; i++) {
43         Edge& e = E[G[cur][i]];
44         if(level[e.s]+1 != level[e.t])
            continue;
45         int flow = dfs(e.t, min(lim,
            e.cap-e.flow));
46         if(flow <= 0) continue;
47         e.flow += flow;
48         result += flow;
49         E[G[cur][i]^1].flow -= flow;
50         lim -= flow;
51     }
52     return result;
53 }
54 int dinic() { // O(V^2)E
55     int result = 0;
56     while(bfs()) {
57         memset(dfs_idx, 0, sizeof(dfs_idx));
58         result += dfs(S, inf);
59     }
60     return result;
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     }
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 %
        MOD);
73     return 0;
74 }

```

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] >=
                dfn[u])
                isCut = true;
21         }
22         else if(v != parent)
23             low[u] = min(low[u], dfn[v]);
24     }
25     //If u is root of DFS
    //tree->有兩個以上的children
26     if(parent == -1 && child >= 2)
27         isCut = true;
28     if(isCut) ++res;
29 }
30 int main() {
31     char input[105];
32     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             {
54                 v = atoi(token);
55                 G[u].emplace_back(v);
56                 G[v].emplace_back(u);
57             }
58         tarjan(1, -1);
59         printf("%d\n", res);
60     }
61 }

```

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) {
40                 if(cur == root) break;
41                 vis[cur] = i;
42                 cur = pre[cur];
43             }
44             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| -
2    /最大匹配數/，用匈牙利演算法找出最大匹配數 */
3 vector<Student> boys;
4 vector<Student> girls;
5 vector<vector<int>> G;
6 bool used[505];
7 int p[505];
8 bool match(int i) {
9     for (int j: G[i]) {
10         if (!used[j]) {
11             used[j] = true;
12             if (p[j] == -1 || match(p[j])) {
13                 p[j] = i;
14                 return true;
15             }
16         }
17     }
18     return false;
19 }
20 void maxMatch(int n) {
21     memset(p, -1, sizeof(p));
22     int res = 0;
23     for (int i = 0; i < boys.size(); ++i) {
24         memset(used, false, sizeof(used));
25         if (match(i))
26             ++res;
27     }
28     cout << n - res << '\n';
29 }

```

3.9 JosephusProblem

```

1 //JosephusProblem，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
65         }
66     }
67 }
68 int main() {
69     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 }

```

```

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 +
                1), (q) ? ' ' : '\n');
70         }
71     }
72     return 0;
73 }

```

```

38     edge.cost)) {
39         dis[edge.v] = dis[u] +
40             edge.cost;
41         parent[edge.v] = edgeIndex;
42         outFlow[edge.v] =
43             min(outFlow[u], (long
44                 long)(edge.cap -
45                     edge.flow));
46         if (!inqueue[edge.v]) {
47             q.push(edge.v);
48             inqueue[edge.v] = true;
49         }
50     }
51 }
52 //如果dis[t] > 0代表根本不賺還倒賠
53 if (dis[t] > 0)
54     return false;
55 maxFlow += outFlow[t];
56 minCost += dis[t] * outFlow[t];
57 //一路更新回去這次最短路流完後要維護的
58 //MaxFlow演算法相關(如反向邊等)
59 int curr = t;
60 while (curr != s) {
61     edges[parent[curr]].flow +=
62         outFlow[t];
63     edges[parent[curr] ^ 1].flow -=
64         outFlow[t];
65     curr = edges[parent[curr]].u;
66 }
67 return true;
68 }
69 long long MCMF() {
70     long long maxFlow = 0;
71     long long minCost = 0;
72     while (SPFA(maxFlow, minCost))
73         ;
74     return minCost;
75 }
76 int main() {
77     int T;
78     scanf("%d", &T);
79     for (int Case = 1; Case <= T; ++Case){
80         //總共幾個月，囤貨成本
81         int M, I;
82         scanf("%d %d", &M, &I);
83         //node size
84         n = M + M + 2;
85         G.assign(n + 5, vector<int>());
86         edges.clear();
87         s = 0;
88         t = M + M + 1;
89         for (int i = 1; i <= M; ++i) {
90             int produceCost, produceMax,
91                 sellPrice, sellMax,
92                 inventoryMonth;
93             scanf("%d %d %d %d %d",
94                 &produceCost, &produceMax,
95                 &sellPrice, &sellMax,
96                 &inventoryMonth);
97             addEdge(s, i, produceMax,
98                 produceCost);
99             addEdge(M + i, t, sellMax,
100                 -sellPrice);
101             for (int j = 0; j <=
102                 inventoryMonth; ++j) {
103                 if (i + j <= M)
104                     addEdge(i, M + i + j, INF,
105                         I * j);
106             }
107         }
108         printf("Case %d: %lld\n", Case,
109             -MCMF());
110     }
111     return 0;
112 }

```

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;
45     //往上找，一起跳，但x、y不能重疊
46     for (int i = 30; i >= 0 && y != x; --i) {
47         if (fa[x][i] != fa[y][i]) {
48             res += dis[x][i] + dis[y][i];
49             x = fa[x][i];
50             y = fa[y][i];
51         }
52     }
53     return res;
54 }

```

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
15     cost) {
16     edges.emplace_back(Edge{u, v, cap, 0,
17         cost});
18     edges.emplace_back(Edge{v, u, 0, 0,
19         -cost});
20     m = edges.size();
21     G[u].emplace_back(m - 2);
22     G[v].emplace_back(m - 1);
23 }
24 //一邊求最短路的同時一邊MaxFlow
25 bool SPFA(long long& maxFlow, long long&
26     minCost) {
27     // memset(outFlow, 0x3f,
28     sizeof(outFlow));
29     memset(dis, 0x3f, sizeof(dis));
30     memset(inqueue, false, sizeof(inqueue));
31     queue<int> q;
32     q.push(s);
33     dis[s] = 0;
34     inqueue[s] = true;
35     outFlow[s] = INF;
36     while (!q.empty()) {
37         int u = q.front();
38         q.pop();
39         inqueue[u] = false;
40         for (const int edgeIndex: G[u]) {
41             const Edge& edge =
42                 edges[edgeIndex];
43             if ((edge.cap > edge.flow) &&
44                 (dis[edge.v] > dis[u] +
45                     edge.cost)) {
46                 dis[edge.v] = dis[u] +
47                     edge.cost;
48                 parent[edge.v] = edgeIndex;
49                 outFlow[edge.v] =
50                     min(outFlow[u], (long
51                         long)(edge.cap -
52                             edge.flow));
53                 if (!inqueue[edge.v]) {
54                     q.push(edge.v);
55                     inqueue[edge.v] = true;
56                 }
57             }
58         }
59     }
60     maxFlow += outFlow[t];
61     minCost += dis[t] * outFlow[t];
62     //一路更新回去這次最短路流完後要維護的
63     //MaxFlow演算法相關(如反向邊等)
64     int curr = t;
65     while (curr != s) {
66         edges[parent[curr]].flow +=
67             outFlow[t];
68         edges[parent[curr] ^ 1].flow -=
69             outFlow[t];
70         curr = edges[parent[curr]].u;
71     }
72     return true;
73 }
74 long long MCMF() {
75     long long maxFlow = 0;
76     long long minCost = 0;
77     while (SPFA(maxFlow, minCost))
78         ;
79     return minCost;
80 }
81 int main() {
82     int T;
83     scanf("%d", &T);
84     for (int Case = 1; Case <= T; ++Case){
85         //總共幾個月，囤貨成本
86         int M, I;
87         scanf("%d %d", &M, &I);
88         //node size
89         n = M + M + 2;
90         G.assign(n + 5, vector<int>());
91         edges.clear();
92         s = 0;
93         t = M + M + 1;
94         for (int i = 1; i <= M; ++i) {
95             int produceCost, produceMax,
96                 sellPrice, sellMax,
97                 inventoryMonth;
98             scanf("%d %d %d %d %d",
99                 &produceCost, &produceMax,
100                 &sellPrice, &sellMax,
101                 &inventoryMonth);
102             addEdge(s, i, produceMax,
103                 produceCost);
104             addEdge(M + i, t, sellMax,
105                 -sellPrice);
106             for (int j = 0; j <=
107                 inventoryMonth; ++j) {
108                 if (i + j <= M)
109                     addEdge(i, M + i + j, INF,
110                         I * j);
111             }
112         }
113         printf("Case %d: %lld\n", Case,
114             -MCMF());
115     }
116     return 0;
117 }

```

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]=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,
19        ++colSize[c];
20        U[seq]=c, D[seq]=D[c], U[D[c]]=seq,
21        D[c]=seq;
22        if(rowHead[r]) {
23            L[seq]=rowHead[r],
24            R[seq]=R[rowHead[r]];
25            L[R[rowHead[r]]]=seq,
26            R[rowHead[r]]=seq;
27        } else {
28            rowHead[r] = L[seq] = R[seq] =
29            seq;
30        }
31    }
32    void remove(int c) {
33        L[R[c]] = L[c], R[L[c]] = R[c];
34        for(int i=D[c]; i!=c; i=D[i]) {
35            for(int j=R[i]; j!=i; j=R[j]) {
36                U[D[j]] = U[j];
37                D[U[j]] = D[j];
38                --colSize[col[j]];
39            }
40        }
41    }
42    void recover(int c) {
43        for(int i=U[c]; i!=c; i=U[i]) {
44            for(int j=L[i]; j!=i; j=L[j]) {
45                U[D[j]] = D[U[j]] = j;
46                ++colSize[col[j]];
47            }
48        }
49        L[R[c]] = R[L[c]] = c;
50    }
51    bool dfs(int idx=0) { // 判斷其中一解版
52        if(R[0] == 0) {
53            resSize = idx;
54            return true;
55        }
56        int c = R[0];
57        for(int i=R[0]; i; i=R[i]) {
58            if(colSize[i] < colSize[c]) c = i;
59        }
60        remove(c);
61        for(int i=D[c]; i!=c; i=D[i]) {
62            result[idx] = row[i];
63            for(int j=R[i]; j!=i; j=R[j])
64                remove(col[j]);
65            if(dfs(idx+1)) return true;
66            for(int j=L[i]; j!=i; j=L[j])
67                recover(col[j]);
68        }
69        recover(c);
70        return false;
71    }
72    void dfs(int idx=0) { // 判斷最小 dfs
73        depth 版
74        if(R[0] == 0) {
75            resSize = min(resSize, idx); //
76            注意init值
77        }
78    }
79    return;
80    }
81    int c = R[0];
82    for(int i=R[0]; i; i=R[i]) {
83        if(colSize[i] < colSize[c]) c = i;
84    }
85    remove(c);
86    for(int i=D[c]; i!=c; i=D[i]) {
87        for(int j=R[i]; j!=i; j=R[j]) {
88            remove(col[j]);
89            dfs(idx+1);
90            for(int j=L[i]; j!=i; j=L[j])
91                recover(col[j]);
92        }
93    }
94    recover(c);
95    }
96    }
97    }
98    }
99    }
100    }
101    }
102    }
103    }
104    }
105    }
106    }
107    }
108    }
109    }
110    }
111    }
112    }
113    }
114    }
115    }
116    }
117    }
118    }
119    }
120    }
121    }
122    }
123    }
124    }
125    }
126    }
127    }
128    }
129    }
130    }
131    }
132    }
133    }
134    }
135    }
136    }
137    }
138    }
139    }
140    }
141    }
142    }
143    }
144    }
145    }
146    }
147    }
148    }
149    }
150    }
151    }
152    }
153    }
154    }
155    }
156    }
157    }
158    }
159    }
160    }
161    }
162    }
163    }
164    }
165    }
166    }
167    }
168    }
169    }
170    }
171    }
172    }
173    }
174    }
175    }
176    }
177    }
178    }
179    }
180    }
181    }
182    }
183    }
184    }
185    }
186    }
187    }
188    }
189    }
190    }
191    }
192    }
193    }
194    }
195    }
196    }
197    }
198    }
199    }
200    }
201    }
202    }
203    }
204    }
205    }
206    }
207    }
208    }
209    }
210    }
211    }
212    }
213    }
214    }
215    }
216    }
217    }
218    }
219    }
220    }
221    }
222    }
223    }
224    }
225    }
226    }
227    }
228    }
229    }
230    }
231    }
232    }
233    }
234    }
235    }
236    }
237    }
238    }
239    }
240    }
241    }
242    }
243    }
244    }
245    }
246    }
247    }
248    }
249    }
250    }
251    }
252    }
253    }
254    }
255    }
256    }
257    }
258    }
259    }
260    }
261    }
262    }
263    }
264    }
265    }
266    }
267    }
268    }
269    }
270    }
271    }
272    }
273    }
274    }
275    }
276    }
277    }
278    }
279    }
280    }
281    }
282    }
283    }
284    }
285    }
286    }
287    }
288    }
289    }
290    }
291    }
292    }
293    }
294    }
295    }
296    }
297    }
298    }
299    }
300    }
301    }
302    }
303    }
304    }
305    }
306    }
307    }
308    }
309    }
310    }
311    }
312    }
313    }
314    }
315    }
316    }
317    }
318    }
319    }
320    }
321    }
322    }
323    }
324    }
325    }
326    }
327    }
328    }
329    }
330    }
331    }
332    }
333    }
334    }
335    }
336    }
337    }
338    }
339    }
340    }
341    }
342    }
343    }
344    }
345    }
346    }
347    }
348    }
349    }
350    }
351    }
352    }
353    }
354    }
355    }
356    }
357    }
358    }
359    }
360    }
361    }
362    }
363    }
364    }
365    }
366    }
367    }
368    }
369    }
370    }
371    }
372    }
373    }
374    }
375    }
376    }
377    }
378    }
379    }
380    }
381    }
382    }
383    }
384    }
385    }
386    }
387    }
388    }
389    }
390    }
391    }
392    }
393    }
394    }
395    }
396    }
397    }
398    }
399    }
400    }
401    }
402    }
403    }
404    }
405    }
406    }
407    }
408    }
409    }
410    }
411    }
412    }
413    }
414    }
415    }
416    }
417    }
418    }
419    }
420    }
421    }
422    }
423    }
424    }
425    }
426    }
427    }
428    }
429    }
430    }
431    }
432    }
433    }
434    }
435    }
436    }
437    }
438    }
439    }
440    }
441    }
442    }
443    }
444    }
445    }
446    }
447    }
448    }
449    }
450    }
451    }
452    }
453    }
454    }
455    }
456    }
457    }
458    }
459    }
460    }
461    }
462    }
463    }
464    }
465    }
466    }
467    }
468    }
469    }
470    }
471    }
472    }
473    }
474    }
475    }
476    }
477    }
478    }
479    }
480    }
481    }
482    }
483    }
484    }
485    }
486    }
487    }
488    }
489    }
490    }
491    }
492    }
493    }
494    }
495    }
496    }
497    }
498    }
499    }
500    }
501    }
502    }
503    }
504    }
505    }
506    }
507    }
508    }
509    }
510    }
511    }
512    }
513    }
514    }
515    }
516    }
517    }
518    }
519    }
520    }
521    }
522    }
523    }
524    }
525    }
526    }
527    }
528    }
529    }
530    }
531    }
532    }
533    }
534    }
535    }
536    }
537    }
538    }
539    }
540    }
541    }
542    }
543    }
544    }
545    }
546    }
547    }
548    }
549    }
550    }
551    }
552    }
553    }
554    }
555    }
556    }
557    }
558    }
559    }
560    }
561    }
562    }
563    }
564    }
565    }
566    }
567    }
568    }
569    }
570    }
571    }
572    }
573    }
574    }
575    }
576    }
577    }
578    }
579    }
580    }
581    }
582    }
583    }
584    }
585    }
586    }
587    }
588    }
589    }
590    }
591    }
592    }
593    }
594    }
595    }
596    }
597    }
598    }
599    }
600    }
601    }
602    }
603    }
604    }
605    }
606    }
607    }
608    }
609    }
610    }
611    }
612    }
613    }
614    }
615    }
616    }
617    }
618    }
619    }
620    }
621    }
622    }
623    }
624    }
625    }
626    }
627    }
628    }
629    }
630    }
631    }
632    }
633    }
634    }
635    }
636    }
637    }
638    }
639    }
640    }
641    }
642    }
643    }
644    }
645    }
646    }
647    }
648    }
649    }
650    }
651    }
652    }
653    }
654    }
655    }
656    }
657    }
658    }
659    }
660    }
661    }
662    }
663    }
664    }
665    }
666    }
667    }
668    }
669    }
670    }
671    }
672    }
673    }
674    }
675    }
676    }
677    }
678    }
679    }
680    }
681    }
682    }
683    }
684    }
685    }
686    }
687    }
688    }
689    }
690    }
691    }
692    }
693    }
694    }
695    }
696    }
697    }
698    }
699    }
700    }
701    }
702    }
703    }
704    }
705    }
706    }
707    }
708    }
709    }
710    }
711    }
712    }
713    }
714    }
715    }
716    }
717    }
718    }
719    }
720    }
721    }
722    }
723    }
724    }
725    }
726    }
727    }
728    }
729    }
730    }
731    }
732    }
733    }
734    }
735    }
736    }
737    }
738    }
739    }
740    }
741    }
742    }
743    }
744    }
745    }
746    }
747    }
748    }
749    }
750    }
751    }
752    }
753    }
754    }
755    }
756    }
757    }
758    }
759    }
760    }
761    }
762    }
763    }
764    }
765    }
766    }
767    }
768    }
769    }
770    }
771    }
772    }
773    }
774    }
775    }
776    }
777    }
778    }
779    }
780    }
781    }
782    }
783    }
784    }
785    }
786    }
787    }
788    }
789    }
790    }
791    }
792    }
793    }
794    }
795    }
796    }
797    }
798    }
799    }
800    }
801    }
802    }
803    }
804    }
805    }
806    }
807    }
808    }
809    }
810    }
811    }
812    }
813    }
814    }
815    }
816    }
817    }
818    }
819    }
820    }
821    }
822    }
823    }
824    }
825    }
826    }
827    }
828    }
829    }
830    }
831    }
832    }
833    }
834    }
835    }
836    }
837    }
838    }
839    }
840    }
841    }
842    }
843    }
844    }
845    }
846    }
847    }
848    }
849    }
850    }
851    }
852    }
853    }
854    }
855    }
856    }
857    }
858    }
859    }
860    }
861    }
862    }
863    }
864    }
865    }
866    }
867    }
868    }
869    }
870    }
871    }
872    }
873    }
874    }
875    }
876    }
877    }
878    }
879    }
880    }
881    }
882    }
883    }
884    }
885    }
886    }
887    }
888    }
889    }
890    }
891    }
892    }
893    }
894    }
895    }
896    }
897    }
898    }
899    }
900    }
901    }
902    }
903    }
904    }
905    }
906    }
907    }
908    }
909    }
910    }
911    }
912    }
913    }
914    }
915    }
916    }
917    }
918    }
919    }
920    }
921    }
922    }
923    }
924    }
925    }
926    }
927    }
928    }
929    }
930    }
931    }
932    }
933    }
934    }
935    }
936    }
937    }
938    }
939    }
940    }
941    }
942    }
943    }
944    }
945    }
946    }
947    }
948    }
949    }
950    }
951    }
952    }
953    }
954    }
955    }
956    }
957    }
958    }
959    }
960    }
961    }
962    }
963    }
964    }
965    }
966    }
967    }
968    }
969    }
970    }
971    }
972    }
973    }
974    }
975    }
976    }
977    }
978    }
979    }
980    }
981    }
982    }
983    }
984    }
985    }
986    }
987    }
988    }
989    }
990    }
991    }
992    }
993    }
994    }
995    }
996    }
997    }
998    }
999    }
1000   }
```

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     // 隨題目改變 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 = 1 + ((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 = 1 + ((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     }
50     int mid = 1 + ((r - l) >> 1);
51     if (tag[i] && l != r) {
52         //如果當前懶標有值則更新左右節點
53         st[i * 2] += tag[i] * (mid - l + 1);
54         st[i * 2 + 1] += tag[i] * (r - mid);
55         tag[i * 2] += tag[i];
56         tag[i * 2 + 1] += tag[i];
57         tag[i] = 0;
58     }
59     if (ql <= mid)
60         update(ql, qr, l, mid, i * 2, c);
61     if (qr > mid)
62         update(ql, qr, mid + 1, r, i * 2 + 1, c);
63 }
64 }
```

```

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的
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     modifyY(1, 1, N, val, yPos, index, false);
45 }
46 void queryY(int index, int l, int r, int yql, int yqr, int xIndex, int& vmax, int& vmin) {
47     if (yql <= l && r <= yqr) {
48         vmax = max(vmax,
49         maxST[xIndex][index]);
50     }
51     else {
52         int mid = (l + r) / 2;
53         if (yql <= mid)
54             queryY(index * 2, l, mid, yql, yqr, xIndex, vmax, vmin);
55         if (yqr > mid)
56             queryY(index * 2 + 1, mid + 1, r, yql, yqr, xIndex, vmax, vmin);
57     }
58 }
```

```

41     vmin = min(vmin,
42               minST[xIndex][index]);
43 }
44 else
45 {
46     int mid = (l + r) / 2;
47     if (yql <= mid)
48         queryY(index * 2, l, mid, yql,
49               yqr, xIndex, vmax, vmin);
50     if (mid < yqr)
51         queryY(index * 2 + 1, mid + 1, r,
52               yql, yqr, xIndex, vmax,
53               vmin);
54 }
55 }
56 void queryX(int index, int l, int r, int
57 xql, int xqr, int yql, int yqr, int&
58 vmax, int& vmin) {
59     if (xql <= l && r <= xqr) {
60         queryY(1, 1, N, yql, yqr, index,
61               vmax, vmin);
62     }
63     else {
64         int mid = (l + r) / 2;
65         if (xql <= mid)
66             queryX(index * 2, l, mid, xql,
67                   xqr, yql, yqr, vmax, vmin);
68         if (mid < xqr)
69             queryX(index * 2 + 1, mid + 1, r,
70                   xql, xqr, yql, yqr, vmax,
71                   vmin);
72     }
73 }
74 int main() {
75     while (scanf("%d", &N) != EOF) {
76         int val;
77         for (int i = 1; i <= N; ++i) {
78             for (int j = 1; j <= N; ++j) {
79                 scanf("%d", &val);
80                 modifyX(1, 1, N, val, i, j);
81             }
82         }
83         int q;
84         int vmax, vmin;
85         int xql, xqr, yql, yqr;
86         char op;
87         scanf("%d", &q);
88         while (q--) {
89             getchar(); //for \n
90             scanf("%c", &op);
91             if (op == 'q') {
92                 scanf("%d %d %d %d", &xql,
93                       &yql, &xqr, &yqr);
94                 vmax = -0x3f3f3f3f;
95                 vmin = 0x3f3f3f3f;
96                 queryX(1, 1, N, xql, xqr,
97                       yql, yqr, vmax, vmin);
98                 printf("%d %d\n", vmax, vmin);
99             }
100             else {
101                 scanf("%d %d %d", &xql, &yql,
102                       &val);
103                 modifyX(1, 1, N, val, xql,
104                       yql);
105             }
106         }
107     }
108     return 0;
109 }

```

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         ++st[index];
12     return;
13 }
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                 {
69                     printf("%d\n", query(1, 1,
70                           stSize, k));
71                     ++k;
72                     ++getCount;
73                 }
74             else {
75                 update(1, 1, stSize,
76                       nums[addCount + 1]);
77                 ++addCount;
78             }
79         }
80     }
81     return 0;
82 }

```

4.4 Trie

```

1 const int maxc = 26; // 單字字數
2 const char minc = 'a'; // 首個 ASCII
3
4 struct TrieNode {
5     int cnt;
6     TrieNode* child[maxc];
7
8     TrieNode() {
9         cnt = 0;
10         for(auto& node : child) {
11             node = nullptr;
12         }
13     }
14 };
15
16 struct Trie {
17     TrieNode* root;
18
19     Trie() { root = new TrieNode(); }
20
21     void insert(string word) {
22         TrieNode* cur = root;
23         for(auto& ch : word) {
24             int c = ch - minc;
25             if(!cur->child[c])
26                 cur->child[c] = new TrieNode();
27             cur = cur->child[c];
28         }
29         cur->cnt++;
30     }
31
32     void remove(string word) {
33         TrieNode* cur = root;
34         for(auto& ch : word) {
35             int c = ch - minc;
36             if(!cur->child[c]) return;
37             cur = cur->child[c];
38         }
39         cur->cnt--;
40     }
41
42     // 字典裡有出現 word
43     bool search(string word, bool prefix=0) {
44         TrieNode* cur = root;
45         for(auto& ch : word) {
46             int c = ch - minc;
47             if(!cur->child[c]) return false;
48             cur = cur->child[c];
49         }
50         return cur->cnt || prefix;
51     }
52
53     // 字典裡有 word 的前綴為 prefix
54     bool startsWith(string prefix) {
55         return search(prefix, true);
56     }
57 };

```

4.5 AC Trie

```

1 const int maxn = 1e4 + 10; // 單字字數
2 const int maxl = 50 + 10; // 單字字長
3 const int maxc = 128; // 單字字數
4 const char minc = ' '; // 首個 ASCII
5
6 int trie[maxn*maxl][maxc]; // 原字典樹
7 int val[maxn*maxl]; // 結尾(單字編號)
8 int cnt[maxn*maxl]; // 結尾(重複個數)
9 int fail[maxn*maxl]; // failure link
10 bool vis[maxn*maxl]; // 同單字不重複
11
12 struct ACTrie {
13     int seq, root;
14
15     ACTrie() {

```



```

16 seq = 0;
17 root = newNode();
18 }
19
20 int newNode() {
21     for(int i=0; i<maxc; i++) trie[seq][i]=0;
22     val[seq] = cnt[seq] = fail[seq] = 0;
23     return seq++;
24 }
25
26 void insert(char* s, int wordId=0) {
27     int p = root;
28     for(; *s; s++) {
29         int c = *s - minc;
30         if(!trie[p][c]) trie[p][c] = newNode();
31         p = trie[p][c];
32     }
33     val[p] = wordId;
34     cnt[p]++;
35 }
36
37 void build() {
38     queue<int> q({root});
39     while(!q.empty()) {
40         int p = q.front();
41         q.pop();
42         for(int i=0; i<maxc; i++) {
43             int& t = trie[p][i];
44             if(t) {
45                 fail[t] = p?trie[fail[p]][i]:root;
46                 q.push(t);
47             } else {
48                 t = trie[fail[p]][i];
49             }
50         }
51     }
52 }
53
54 // 要存 wordId 才要 vec
55 // 同單字重複match要把所有vis取消掉
56 int match(char* s, vector<int>& vec) {
57     int res = 0;
58     memset(vis, 0, sizeof(vis));
59     for(int p=root; *s; s++) {
60         p = trie[p][*s-minc];
61         for(int k=p; k && !vis[k]; k=fail[k]) {
62             vis[k] = true;
63             res += cnt[k];
64             if(cnt[k]) vec.push_back(val[k]);
65         }
66     }
67     return res; // 匹配到的單字量
68 }
69 };
70
71 ACTrie ac; // 建構, 初始化
72 ac.insert(s); // 加字典單字
73 // 加完字典後
74 ac.build(); // !!! 建 failure link !!!
75 ac.match(s); // 多模式匹配(加vec存編號)

```

4.6 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
47 int main(){
48     cin>>n>>k; //每k個連續的數
49     for(int i=1;i<=n;i++) cin>>a[i];
50     getmin();
51     getmax();
52     return 0;
53 }

```

5 Geometry

5.1 Template

```

1 using DBL = double;
2 using TP = DBL; // 存點的型態
3
4 const DBL pi = acos(-1);
5 const DBL eps = 1e-8;
6 const TP inf = 1e30;
7 const int maxn = 5e4 + 10;
8
9 struct Vector {
10     TP x, y;
11     Vector(TP x=0, TP y=0): x(x), y(y) {}
12     DBL length();
13 };
14
15 Vector operator+(Vector a, Vector b) {
16     return Vector(a.x+b.x, a.y+b.y); }
17 Vector operator-(Vector a, Vector b) {
18     return Vector(a.x-b.x, a.y-b.y); }
19 Vector operator*(Vector a, DBL b) {
20     return Vector(a.x*b, a.y*b); }
21 Vector operator/(Vector a, DBL b) {
22     return Vector(a.x/b, a.y/b); }
23
24 TP dot(Vector a, Vector b) {
25     return a.x*b.x + a.y*b.y;
26 }
27 TP cross(Vector a, Vector b) {
28     return a.x*b.y - a.y*b.x;
29 }
30 DBL Vector::length() {
31     return sqrt(dot(*this, *this));
32 }
33 Vector unit_normal_vector(Vector v) {
34     DBL len = v.length();
35     return Vector(-v.y/len, v.x/len);

```

```

36 }
37
38 using Point = Vector;
39 using Polygon = vector<Point>;
40
41 struct Line {
42     Point p;
43     Vector v;
44     DBL ang;
45     Line(Point _p={}, Vector _v={}) {
46         p = _p;
47         v = _v;
48         ang = atan2(v.y, v.x);
49     }
50     bool operator<(const Line& l) const {
51         return ang < l.ang;
52     }
53 };

```

5.2 凸包

- TP 為 Point 裡 x 和 y 的型態
- struct Point 需要加入並另外計算的 variables:
 1. ang, 該點與基準點的 atan2 值
 2. d2, 該點與基準點的 (距離)²

```

1 using TP = long long;
2 using Polygon = vector<Point>;
3
4 const TP inf = 1e9; // 座標點最大值
5
6 Polygon convex_hull(Point* p, int n) {
7     auto dblcmp = [](DBL a, DBL b=0.0) {
8         return (a>b) - (a<b);
9     };
10     auto rmv = [&](Point a, Point b, Point c) {
11         return cross(b-a, c-b) <= 0; // 非浮點數
12         return dblcmp(cross(b-a, c-b)) <= 0;
13     };
14
15     // 選最下裡最左的當基準點，可在輸入時計算
16     TP lx = inf, ly = inf;
17     for(int i=0; i<n; i++) {
18         if(p[i].y<ly || (p[i].y==ly&&p[i].x<lx)){
19             lx = p[i].x, ly = p[i].y;
20         }
21     }
22
23     for(int i=0; i<n; i++) {
24         p[i].ang=atan2(p[i].y-ly,p[i].x-lx);
25         p[i].d2 = (p[i].x-lx)*(p[i].x-lx) +
26             (p[i].y-ly)*(p[i].y-ly);
27     }
28     sort(p, p+n, [&](Point& a, Point& b) {
29         if(dblcmp(a.ang, b.ang))
30             return a.ang < b.ang;
31         return a.d2 < b.d2;
32     });
33
34     int m = 1; // stack size
35     Point st[n] = {p[n]=p[0]};
36     for(int i=1; i<=n; i++) {
37         for(;m>1&&rmv(st[m-2],st[m-1],p[i]);m--);
38         st[m++] = p[i];
39     }
40     return Polygon(st, st+m-1);
41 }

```

5.3 半平面相交

```

1 using DBL = double;
2 using TP = DBL; // 存點的型態
3 using Polygon = vector<Point>;

```

```

4
5 const int maxn = 5e4 + 10;
6
7 Point intersection(Line a, Line b) {
8     Vector u = a.p - b.p;
9     DBL t = 1.0*cross(b.v, u)/cross(a.v, b.v);
10    return a.p + a.v*t;
11 }
12
13 // Return: 能形成半平面交的凸包邊界點
14 Polygon halfplaneIntersect(vector<Line>&nar){
15     sort(nar.begin(), nar.end());
16     // DBL 跟 0 比較, 沒浮點數不用
17     auto dblcmp=[](DBL v){return (v>0)-(v<0)};
18     // p 是否在 l 的左半平面
19     auto lft = [&](Point p, Line l) {
20         return dblcmp(cross(l.v, p-l.p)) > 0;
21     };
22
23     int ql = 0, qr = 0;
24     Line L[maxn] = {nar[0]};
25     Point P[maxn];
26
27     for(int i=1; i<nar.size(); i++) {
28         for(; ql<qr&&!lft(P[qr-1], nar[i]); qr--);
29         for(; ql<qr&&!lft(P[ql], nar[i]); ql++);
30         L[++qr] = nar[i];
31         if(dblcmp(cross(L[qr].v, L[qr-1].v))==0) {
32             if(lft(nar[i].p, L[qr-1])) L[qr]=nar[i];
33         }
34         if(ql < qr)
35             P[qr-1] = intersection(L[qr-1], L[qr]);
36     }
37     for(; ql<qr && !lft(P[qr-1], L[ql]); qr--);
38     if(qr-ql <= 1) return {};
39     P[qr] = intersection(L[qr], L[ql]);
40     return Polygon(P+ql, P+qr+1);
41 }

```

5.4 Polygon

```

1 // 判斷點 (point) 是否在凸包 (p) 內
2 bool inConvex(Polygon& p, Point point) {
3     // 根據 TP 型態來寫, 沒浮點數不用 dblcmp
4     auto dblcmp=[](DBL v){return (v>0)-(v<0)};
5     // 不包含線上, 改 '>=' 為 '<'
6     auto test = [&](Point& p0, Point& p1) {
7         return dblcmp(cross(p1-p0, point-p0))>=0;
8     };
9     p.push_back(p[0]);
10    for(int i=1; i<p.size(); i++) {
11        if(!test(p[i-1], p[i])) {
12            p.pop_back();
13            return false;
14        }
15    }
16    p.pop_back();
17    return true;
18 }
19
20 // 計算簡單多邊形的面積
21 // ! p 為排序過的點 !
22 DBL polygonArea(Polygon& p) {
23     DBL sum = 0;
24     for(int i=0, n=p.size(); i<n; i++)
25         sum += cross(p[i], p[(i+1)%n]);
26     return abs(sum) / 2.0;
27 }

```

5.5 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        a = e.y - s.y;
12        b = s.x - e.x;
13        c = a * s.x + b * s.y;
14    }
15 };
16
17 // 用克拉馬公式求二元一次解
18 Point2D intersection2D(Line2D l1, Line2D l2){
19     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 }

```

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 RangedDP

```

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] = l;
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

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

Longest Palindromic Subsequence

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

6.6 樹 DP 有幾個 path 長度為 k

```

1 #define maxn 5005
2 #define maxk 505
3 //dp[u][u的child且距離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  $O(n + n + n) \rightarrow O(n)*$ 
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     }
36     //自己
37     this->subtreeSize[u] += 1;
38     return this->subtreeSize[u];
39 }
40 //求res[0], 0到所有點的距離
41 int dfs2(int u, int dis) {
42     this->visited[u] = true;
43     int sum = 0;
44     for (int v: this->G[u])
45         if (!visited[v])

```

```

40         sum += this->dfs2(v, dis + 1);
41     //要加上自己的距離
42     return sum + dis;
43 }
44 //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] +
50                 n - 2 *
51                 this->subtreeSize[v];
52             this->dfs3(v, n);
53         }
54     }
55 }

```

6.8 Weighted LIS

```

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,
7     long long v) {
8     if (l == r) {
9         st[index] = v;
10        return;
11    }
12    int mid = (l + r) >> 1;
13    if (p <= mid)
14        update(p, (index << 1), l, mid, v);
15    else
16        update(p, (index << 1) + 1, mid + 1, r, v);
17    st[index] =
18        max(st[index << 1], st[(index << 1) + 1]);
19 }
20 long long query(int index, int l, int r, int
21     ql, int qr) {
22     if (ql <= l && r <= qr)
23         return st[index];
24     int mid = (l + r) >> 1;
25     long long res = -1;
26     if (ql <= mid)
27         res =
28             max(res, query(index << 1, l, mid, ql, qr));
29     if (mid < qr)
30         res =
31             max(res, query((index << 1) + 1, mid + 1, r, ql, qr));
32     return res;
33 }
34 int main() {
35     int n;
36     scanf("%d", &n);
37     for (int i = 1; i <= n; ++i)
38         scanf("%lld", &height[i]);
39     for (int i = 1; i <= n; ++i)
40         scanf("%lld", &B[i]);
41     long long res = B[1];
42     update(height[1], 1, 1, n, B[1]);
43     for (int i = 2; i <= n; ++i) {
44         long long temp;
45         if (height[i] - 1 >= 1)
46             temp =
47                 B[i] + query(1, 1, n, 1, height[i] - 1);
48         else
49             temp = B[i];
50         update(height[i], 1, 1, n, temp);
51         res = max(res, temp);
52     }
53     printf("%lld\n", res);
54     return 0;
55 }

```