

## Contents

### 1 字串

1.1 最長迴文子字串	1
1.2 KMP	1
1.3 Z Algorithm	1

### 2 math

2.1 公式	1
2.2 模逆元	1
2.3 SG	1
2.4 Fibonacci	1
2.5 矩陣快速幂	1
2.6 質數與因數	2
2.7 歐拉函數	2
2.8 乘法逆元 and 組合數	2

### 3 algorithm

3.1 三分搜	2
3.2 差分	3
3.3 greedy	3
3.4 dinic	4
3.5 SCC Tarjan	4
3.6 ArticulationPoints Tarjan	4
3.7 最小樹狀圖	5
3.8 二分圖最大匹配	5
3.9 JosephusProblem	5
3.10 KM	5
3.11 LCA 倍增法	6
3.12 MCMF	6
3.13 Dancing Links	6

### 4 DataStructure

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 AC Trie	8
4.6 單調隊列	8

### 5 geometry

5.1 intersection	9
5.2 半平面相交	9
5.3 凸包	9

### 6 DP

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	11
6.6 樹 DP 有幾個 path 長度為 k	11
6.7 TreeDP reroot	11
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

```
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 = ababcababababababab
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: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

• Faulhaber's formula

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

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

也可用高斯消去法找 $deg(p+1)$ 的多項式，例：

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

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

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

$$\sum_{k=1}^n k^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

### 2.2 模逆元

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

若  $p$  是 prime, 根據費馬小定理, 則

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

### 2.3 SG

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

### 2.4 Fibonacci

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

### 2.5 矩陣快速幂

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

```

## 2.6 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 最大公因數 O(log(min(a,b)))
18 int GCD(int a, int b){
19     if(b == 0) return a;
20     return GCD(b, a%b);
21 }
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≤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;

```

```

65         for(int j=0; i*p[j]<N&&j<pr; j++)
66             ox[i*p[j]] = 1;
67     }
68 }
69 int main(){
70     PrimeTable();
71     int n;
72     while(cin>>n, n){
73         int x;
74         for(x=1;; x+=2)
75             if(!ox[x] && !ox[n-x]) break;
76         printf("%d = %d + %d\n", n, x, n-x);
77     }
78 }
79
80 problem :
81 給定整數 N，求N最少可以拆成多少個質數的和。
82 如果N是質數，則答案為 1。
83 如果N是偶數(N!=2)，則答案為2(強歌德巴赫猜想)。
84 如果N是奇數且N-2是質數，則答案為2(2+質數)。
85 其他狀況答案為 3 (弱歌德巴赫猜想)。
86
87 bool isPrime(int n){
88     for(int i=2;i<n;++i){
89         if(i*i>n) return true;
90         if(n%i==0) return false;
91     }
92     return true;
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 }

```

## 2.7 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 }

```

## 2.8 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)
    為二次函數，
5 可用三分查找二次函數最小值。
6 struct Point{
7     double x, y, z;
8     Point() {}
9     Point(double _x,double _y,double _z):
10         x(_x),y(_y),z(_z){}
11     friend istream& operator>>(istream& is,
12         Point& p) {
13         is >> p.x >> p.y >> p.z;
14         return is;
15     }
16     Point operator+(const Point &rhs) const{
17         return Point(x+rhs.x,y+rhs.y,z+rhs.z);
18     }
19     Point operator-(const Point &rhs) const{
20         return Point(x-rhs.x,y-rhs.y,z-rhs.z);
21     }
22     Point operator*(const double &d) const{
23         return Point(x*d,y*d,z*d);
24     }
25     Point operator/(const double &d) const{
26         return Point(x/d,y/d,z/d);
27     }
28     double dist(const Point &rhs) const{
29         double res = 0;
30         res+=(x-rhs.x)*(x-rhs.x);
31         res+=(y-rhs.y)*(y-rhs.y);
32         res+=(z-rhs.z)*(z-rhs.z);
33         return res;
34     };
35 int main(){
36     IOS; //輸入優化
37     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) <<
        sqrt(ans) << '\n';
58 }
59 }

```

## 3.2 差分

用途：在區間  $[l, r]$  加上一個數字  $v$ 。

$b[l] += v$ ; ( $b[0..l]$  加上  $v$ )

$b[r+1] -= v$ ; ( $b[r+1..n]$  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  $b[l]$  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  $N(≤10^{100})$ ，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 }

```

最小區間覆蓋長度

**//problem**

給定  $n$  條線段區間為  $[Li, Ri]$ ，

請問最少要選幾個區間才能完全覆蓋  $[0, S]$ ？

**//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++;
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         }
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 }
104 最小化最大延遲問題
105 //problem
106 給定  $N$  項工作，每項工作的需要處理時長為  $Ti$ ，
107 期限是  $Di$ ，第  $i$  項工作延遲的時間為
108  $Li=\max(0, Fi-Di)$ ，
109 原本  $Fi$  為第  $i$  項工作的完成時間，
110 求一種工作排序使  $\max Li$  最小。
111 //solution
112 按照到期時間從早到晚處理。

```

```

111 //code
112 struct Work{
113     int t, d;
114     bool operator<(const Work &rhs)const{
115         return d<rhs.d;
116     }
117 };
118 int main(){
119     int n;
120     Work a[10000];
121     cin>>n;
122     for(int i=0; i<n; ++i)
123         cin>>a[i].t>>a[i].d;
124     sort(a, a+n);
125     int maxL=0, sumT=0;
126     for(int i=0; i<n; ++i){
127         sumT+=a[i].t;
128         maxL=max(maxL, sumT-a[i].d);
129     }
130     cout<<maxL<<'\n';
131 }
132 最少延遲數量問題
133 //problem
134 給定  $N$  個工作，每個工作的需要處理時長為  $Ti$ ，
135 期限是  $Di$ ，求一種工作排序使得逾期工作數量最小。
136 //solution
137 期限越早到期的工作越先做。
138 將工作依照到期時間從早到晚排序，
139 依序放入工作列表中，如果發現有工作預期，
140 就從目前選擇的工作中，移除耗時最長的工作。
141 上述方法為 Moore-Hodgson's Algorithm。
142
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     }
164     sort(a, a+n);
165     int sumT=0, ans=n;
166     for(int i=0; i<n; ++i){
167         pq.push(a[i].t);
168         sumT+=a[i].t;
169         if(a[i].d<sumT){
170             int x=pq.top();
171             pq.pop();
172             sumT-=x;
173             --ans;
174         }
175     }
176     cout<<ans<<'\n';
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  $Di-Ti+1, Di-Ti, \dots, 1, 0$ ，
188 如果有空閒就放進去，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
190 單位獎勵，
191 請問最多會獲得多少單位獎勵。
192 //solution
193 和上題相似，這題變成依照獎勵由大到小排序。
194 //code
195 struct Work{
196     int d,p;
197     bool operator<(const Work &rhs)const{
198         return p>rhs.p;
199     }
200 };
201 int main(){
202     int n;
203     Work a[100005];
204     bitset<100005> ok;
205     while(cin>>n){
206         ok.reset();
207         for(int i=0;i<n;++i)
208             cin>>a[i].d>>a[i].p;
209         sort(a,a+n);
210         int ans=0;
211         for(int i=0;i<n;++i){
212             int j=a[i].d;
213             while(j--){
214                 if(!ok[j]){
215                     ans+=a[i].p;
216                     ok[j]=true;
217                     break;
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 &&
32                 e.cap>e.flow) {
33                 level[e.t] = level[e.s] + 1;
34                 q.push(e.t);
35             }
36         }
37     }

```

```

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 }

```

### 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 %
73         MOD);
74     return 0;
75 }

```

### 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;
31     if (isCut) ++res;
32 }
33 int main() {
34     char input[105];
35     char* token;
36     while (scanf("%d", &N) != EOF && N) {
37         G.assign(105, vector<int>());
38         memset(visited, false,
39             sizeof(visited));
40         memset(low, 0, sizeof(low));
41         memset(dfn, 0, sizeof(dfn));
42         timer = 0;
43         res = 0;
44         getchar(); // for \\n
45         while (fgets(input, 105, stdin)) {

```

```

43     if (input[0] == '0')
44         break;
45     int size = strlen(input);
46     input[size - 1] = '\0';
47     --size;
48     token = strtok(input, " ");
49     int u = atoi(token);
50     int v;
51     while (token = strtok(NULL, " "))
52     {
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 }

```

### 3.7 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) {
40                 if(cur == root) break;
41                 vis[cur] = i;
42                 cur = pre[cur];
43             }
44             if(cur!=root && idx[cur]==-1) {
45                 for(int j=pre[cur]; j!=cur;
46                     j=pre[j])
47                     idx[j] = seq;
48                 idx[cur] = seq++;
49             }
50         }
51         if(seq == 0) return result; // 沒有
52                             // cycle
53         for(int i=0; i<N; i++)
54             // 沒有被縮點的點

```

```

49         if(idx[i] == -1) idx[i] = seq++;
50         // 縮點並重新編號
51         for(Edge& e : edges) {
52             if(idx[e.s] != idx[e.t])
53                 e.cost -= inEdge[e.t];
54             e.s = idx[e.s];
55             e.t = idx[e.t];
56         }
57         N = seq;
58         root = idx[root];
59     }
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     int diff = 0x3f3f3f3f;
33     for (int i = 0; i < n; ++i) {
34         if (S[i]) {
35             for (int j = 0; j < n; ++j) {
36                 if (!T[j])
37                     diff = min(diff, Lx[i] +
38                                 Ly[j] - W[i][j]);
39             }
40         }
41     }
42     for (int i = 0; i < n; ++i) {
43         if (S[i]) Lx[i] -= diff;
44         if (T[i]) Ly[i] += diff;
45     }
46 }
47 void KM() {
48     for (int i = 0; i < n; ++i) {
49         L[i] = -1;
50         Lx[i] = Ly[i] = 0;
51         for (int j = 0; j < n; ++j)
52             Lx[i] = max(Lx[i], W[i][j]);
53     }
54     for (int i = 0; i < n; ++i) {
55         while(1) {
56             memset(S, false, sizeof(S));
57             memset(T, false, sizeof(T));
58             if (match(i))
59                 break;
60             else
61                 update(); //去調整vertex
62                             // labeling以增加增廣路徑
63         }
64     }
65 }
66 int main() {
67     while (scanf("%d", &n) != EOF) {
68         for (int i = 0; i < n; ++i)
69             for (int j = 0; j < n; ++j)
70                 scanf("%d", &W[i][j]);
71         KM();
72         int res = 0;
73         for (int i = 0; i < n; ++i) {
74             if (i != 0)
75                 printf(" %d", Lx[i]);
76             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 $O(n\log n)$ ，查詢 $O(\log n)$ ，
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] + dis[u][i-1];
20     }
21     //遍歷子節點
22     for (Edge& edge: G[u]) {
23         if (edge.v == p)
24             continue;
25         dis[edge.v][0] = edge.w;
26         dfs(edge.v, u);
27     }
28 }
29 long long lca(int x, int y) {
30     //此函數是找lca同時計算x、y的距離 -> dis(x, lca) + dis(lca, y)
31     //讓y比x深
32     if (dep[x] > dep[y])
33         swap(x, y);
34     int deltaDep = dep[y] - dep[x];
35     long long res = 0;
36     //讓y與x在同一個深度
37     for (int i = 0; deltaDep != 0; ++i, deltaDep >>= 1)
38         if (deltaDep & 1)
39             res += dis[y][i], y = fa[y][i];
40     if (y == x) //x = y -> x、y彼此是彼此的祖先
41         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 + 1), (q ? ' ' : '\n'));
70         }
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 //如果dis[t] > 0代表根本不賺還倒賠
48 if (dis[t] > 0)
49     return false;
50 maxFlow += outFlow[t];
51 minCost += dis[t] * outFlow[t];
52 //一路更新回去這次最短路流完後要維護的
53 //MaxFlow演算法相關(如反向邊等)
54 int curr = t;
55 while (curr != s) {
56     edges[parent[curr]].flow += outFlow[t];
57     edges[parent[curr] ^ 1].flow -= outFlow[t];
58     curr = edges[parent[curr]].u;
59 }
60 return true;
61 }
62 long long MCMF() {
63     long long maxFlow = 0;
64     long long minCost = 0;
65     while (SPFA(maxFlow, minCost))
66         ;
67     return minCost;
68 }
69 int main() {
70     int T;
71     scanf("%d", &T);
72     for (int Case = 1; Case <= T; ++Case) {
73         //總共幾個月，囤貨成本
74         int M, I;
75         scanf("%d %d", &M, &I);
76         //node size
77         n = M + M + 2;
78         G.assign(n + 5, vector<int>());
79         edges.clear();
80         s = 0;
81         t = M + M + 1;
82         for (int i = 1; i <= M; ++i) {
83             int produceCost, produceMax, sellPrice, sellMax, inventoryMonth;
84             scanf("%d %d %d %d %d", &produceCost, &produceMax, &sellPrice, &sellMax, &inventoryMonth);
85             addEdge(s, i, produceMax, produceCost);
86             addEdge(M + i, t, sellMax, -sellPrice);
87             for (int j = 0; j <= inventoryMonth; ++j) {
88                 if (i + j <= M)
89                     addEdge(i, M + i + j, INF, I * j);
90             }
91         }
92         printf("Case %d: %lld\n", Case, -MCMF());
93     }
94     return 0;
95 }
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=c]]=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          return;
78      }
79      int c = R[0];
80      for(int i=R[0]; i; i=R[i]) {
81          if(colSize[i] < colSize[c]) c = i;
82      }
83      remove(c);
84      for(int i=D[c]; i!=c; i=D[i]) {
85          for(int j=R[i]; j!=i; j=R[j])

```

```

79      remove(col[j]);
80      dfs(idx+1);
81      for(int j=L[i]; j!=i; j=L[j])
82          recover(col[j]);
83      }
84      recover(c);
85  }
86  };

```

## 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 = 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
22 i) {
23     // [ql, qr]是查詢區間,[l, r]是當前節點包含的區間
24     if (ql <= l && r <= qr)
25         return st[i];
26     int mid = l + ((r - l) >> 1);
27     if (tag[i]) {
28         //如果當前懶標有值則更新左右節點
29         st[i * 2] += tag[i] * (mid - l + 1);
30         st[i * 2 + 1] += tag[i] * (r - mid);
31         tag[i * 2] += tag[i]; //下傳懶標至左節點
32         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
33         tag[i] = 0;
34     }
35     int sum = 0;
36     if (ql <= mid)
37         sum += query(ql, qr, l, mid, i * 2);
38     if (qr > mid)
39         sum += query(ql, qr, mid + 1, r,
40 i * 2 + 1);
41     return sum;
42 }
43 void update(int ql, int qr, int l, int r, int
44 i, int c) {
45     // [ql, qr]是查詢區間,[l, r]是當前節點包含的區間
46     // c是變化量
47     if (ql <= l && r <= qr) {
48         st[i] += (r - l + 1) * c;
49         //求和,此需乘上區間長度
50         tag[i] += c;
51         return;
52     }
53     int mid = l + ((r - l) >> 1);
54     if (tag[i] && l != r) {
55         //如果當前懶標有值則更新左右節點
56         st[i * 2] += tag[i] * (mid - l + 1);
57         st[i * 2 + 1] += tag[i] * (r - mid);
58         tag[i * 2] += tag[i]; //下傳懶標至左節點
59         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
60         tag[i] = 0;
61     }
62     if (ql <= mid) update(ql, qr, l, mid, i
63 * 2, c);
64     if (qr > mid) update(ql, qr, mid + 1, r,
65 i * 2 + 1, c);
66     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
6 val, int yPos, int xIndex, bool
7 xIsLeaf) {
8     if (l == r) {
9         if (xIsLeaf) {
10             maxST[xIndex][index] =
11             minST[xIndex][index] = val;
12             return;
13         }
14         maxST[xIndex][index] =
15         max(maxST[xIndex * 2][index],
16         maxST[xIndex * 2 + 1][index]);
17         minST[xIndex][index] =
18         min(minST[xIndex * 2][index],
19         minST[xIndex * 2 + 1][index]);
20     }
21     else {
22         int mid = (l + r) / 2;
23         if (yPos <= mid)
24             modifyY(index * 2, l, mid, val,
25             yPos, xIndex, xIsLeaf);
26         else
27             modifyY(index * 2 + 1, mid + 1,
28             r, val, yPos, xIndex,
29             xIsLeaf);
30     }
31     maxST[xIndex][index] =
32     max(maxST[xIndex][index * 2],
33     maxST[xIndex][index * 2 + 1]);
34     minST[xIndex][index] =
35     min(minST[xIndex][index * 2],
36     minST[xIndex][index * 2 + 1]);
37 }
38 void modifyX(int index, int l, int r, int
39 val, int xPos, int yPos) {
40     if (l == r) {
41         modifyY(1, 1, N, val, yPos, index,
42         true);
43     }
44     else {
45         int mid = (l + r) / 2;
46         if (xPos <= mid)
47             modifyX(index * 2, l, mid, val,
48             xPos, yPos);
49         else
50             modifyX(index * 2 + 1, mid + 1,
51             r, val, xPos, yPos);
52         modifyY(1, 1, N, val, yPos, index,
53         false);
54     }
55 }
56 void queryY(int index, int l, int r, int
57 yql, int yqr, int xIndex, int& vmax,
58 int& vmin) {
59     if (yql <= l && r <= yqr) {
60         vmax = max(vmax,
61         maxST[xIndex][index]);
62         vmin = min(vmin,
63         minST[xIndex][index]);
64     }
65     else
66     {
67         int mid = (l + r) / 2;
68         if (yql <= mid)
69             queryY(index * 2, l, mid, yql,
70             yqr, xIndex, vmax, vmin);

```

```

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

```

### 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     if (l == r)
10     {
11         ++st[index];
12         return;
13     }

```

```

14     int mid = (l + r) / 2;
15     if (qx <= mid)
16         update(index * 2, l, mid, qx);
17     else
18         update(index * 2 + 1, mid + 1, r, qx);
19     st[index] = st[index * 2] + st[index * 2
20         + 1];
21 }
22 //找區間第k個小的
23 int query(int index, int l, int r, int k) {
24     if (l == r)
25         return id[l];
26     int mid = (l + r) / 2;
27     //k比左子樹小
28     if (k <= st[index * 2])
29         return query(index * 2, l, mid, k);
30     else
31         return query(index * 2 + 1, mid + 1,
            r, k - st[index * 2]);
32 }
33 int main() {
34     int t;
35     cin >> t;
36     bool first = true;
37     while (t--) {
38         if (first)
39             first = false;
40         else
41             puts("");
42         memset(st, 0, sizeof(st));
43         int m, n;
44         cin >> m >> n;
45         for (int i = 1; i <= m; ++i) {
46             cin >> nums[i];
47             id[i] = nums[i];
48         }
49         for (int i = 0; i < n; ++i)
50             cin >> getArr[i];
51         //離散化
52         //防止m == 0
53         if (m)
54             sort(id + 1, id + m + 1);
55         int stSize = unique(id + 1, id + m +
            1) - (id + 1);
56         for (int i = 1; i <= m; ++i) {
57             nums[i] = lower_bound(id + 1, id
                + stSize + 1, nums[i]) - id;
58         }
59         int addCount = 0;
60         int getCount = 0;
61         int k = 1;
62         while (getCount < n) {
63             if (getArr[getCount] == addCount)
64                 printf("%d\n", query(1, 1,
                    stSize, k));
65             ++k;
66             ++getCount;
67         }
68         else {
69             update(1, 1, stSize,
                nums[addCount + 1]);
70             ++addCount;
71         }
72     }
73 }
74 return 0;
75 }

```

### 4.4 Trie

```

1 const int maxc = 26; // 單字字符數
2 const char minc = 'a'; // 首個 ASCII
3
4 struct TrieNode {
5     bool isWord;

```

```

6     TrieNode* child[maxc];
7
8     TrieNode() {
9         isWord = false;
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->isWord = true;
30     }
31
32     // 字典裡有出現 word
33     bool search(string word, bool prefix=0) {
34         TrieNode* cur = root;
35         for(auto& ch : word) {
36             int c = ch - minc;
37             if(!cur->child[c]) return false;
38         }
39         return cur->isWord || prefix;
40     }
41
42     // 字典裡有 word 的前綴為 prefix
43     bool startsWith(string prefix) {
44         return search(prefix, true);
45     }
46 };

```

### 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; // 單字字符數
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() {

```

```

31 int head=0,tail=0;
32 for(int i=1;i<k;i++) {
33     while(head<=tail&&a[q[tail]]<=a[i])tail++;
34     q[++tail]=i;
35 }
36 for(int i=k; i<=n;i++) {
37     while(head<=tail&&a[q[tail]]<=a[i])tail++;
38     q[++tail]=i;
39     while(q[head]<=i-k) head++;
40     cout<<a[q[head]]<<" ";
41 }
42 cout<<endl;
43 }
44
45 int main(){
46     cin>>n>>k; //每k個連續的數
47     for(int i=1;i<=n;i++) cin>>a[i];
48     getmin();
49     getmax();
50     return 0;
51 }

```

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

```

```

16 }
17 Vector operator*(double val) {
18     return Vector(x*val, y*val);
19 }
20 double dot(Vector v) { return x*v.x +
21     y*v.y; }
22 double cross(Vector v) { return x*v.y -
23     y*v.x; }
24 double length() { return
25     sqrt(dot(*this)); }
26 Vector unit_normal_vector() {
27     double len = length();
28     return Vector(-y/len, x/len);
29 }
30
31 using Point = Vector;
32
33 struct Line {
34     Point p;
35     Vector v;
36     double ang;
37     Line(Point p={}, Vector v={}): p(p),
38         v(v) {
39         ang = atan2(v.y, v.x);
40     }
41     bool operator<(const Line& l) const {
42         return ang < l.ang;
43     }
44     Point intersection(Line l) {
45         Vector u = p - l.p;
46         double t = l.v.cross(u) /
47             v.cross(l.v);
48         return p + v*t;
49     }
50 };
51
52 int n, m;
53 Line narrow[maxn]; // 要判斷的直線
54 Point poly[maxn]; // 能形成半平面交的凸包邊界點
55
56 // return true if point p is on the left of
57 // line l
58 bool onLeft(Point p, Line l) {
59     return l.v.cross(p-l.p) > 0;
60 }
61
62 int halfplaneIntersection() {
63     int l, r;
64     Line L[maxn]; // 排序後的向量隊列
65     Point P[maxn]; // s[i] 跟 s[i-1]
66     // 的交點
67
68     L[l=r=0] = narrow[0]; // notice: narrow
69     // is sorted
70     for(int i=1; i<=n; i++) {
71         while(l<r && !onLeft(P[r-1],
72             narrow[i])) r--;
73         while(l<r && !onLeft(P[l],
74             narrow[i])) l++;
75
76         L[++r] = narrow[i];
77         if(l < r) P[r-1] =
78             L[r-1].intersection(L[r]);
79     }
80
81     while(l<r && !onLeft(P[r-1], L[l])) r--;
82     if(r-l <= 1) return 0;
83
84     P[r] = L[r].intersection(L[l]);
85
86     int m=0;
87     for(int i=l; i<=r; i++) {
88         poly[m++] = P[i];
89     }
90     return m;

```

```

82 }
83
84 Point pt[maxn];
85 Vector vec[maxn];
86 Vector normal[maxn]; // normal[i] = vec[i]
    的單位法向量
87
88 double bsearch(double l=0.0, double r=1e4) {
89     if(abs(r-l) < eps) return l;
90
91     double mid = (l + r) / 2;
92
93     for(int i=0; i<n; i++) {
94         narrow[i] = Line(pt[i]+normal[i]*mid,
95             vec[i]);
96     }
97
98     if(halfplaneIntersection())
99         return bsearch(mid, r);
100     else return bsearch(l, mid);
101 }
102
103 int main() {
104     while(~scanf("%d", &n) && n) {
105         for(int i=0; i<n; i++) {
106             double x, y;
107             scanf("%lf%lf", &x, &y);
108             pt[i] = {x, y};
109         }
110         for(int i=0; i<n; i++) {
111             vec[i] = pt[(i+1)%n] - pt[i];
112             normal[i] =
113                 vec[i].unit_normal_vector();
114         }
115         printf("%.6lf\n", bsearch());
116     }
117     return 0;
118 }

```

## 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 =
14         maxCoordinate;
15     for(int i=0; i<n; i++) {
16         int x, y;
17         scanf("%d%d", &x, &y);
18         arr[i] = (Point){x, y};
19         if(y < minY || (y == minY && x <
20             minX)) {
21             minX = x, minY = y;
22         }
23     }
24     sort(arr, arr+n, [minX, minY](Point& a,
25         Point& b){
26         double theta1 = atan2(a.y - minY, a.x
27             - minX);
28         double theta2 = atan2(b.y - minY, b.x
29             - minX);
30         return theta1 < theta2;
31     });
32     return;
33 }

```

```

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])
46                 < 0)
47                 break;
48             st.pop_back();
49             m--;
50         }
51         st.push_back(arr[i]);
52     }
53     return st;
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 solve(int l, int r) {
10     if (l > r) return 0;
11     if (vis[l][r]) return dp[l][r];
12     vis[l][r] = true;
13     long long res = a[l] - solve(l + 1, r);
14     res = max(res, a[r] - solve(l, r - 1));
15     return dp[l][r] = res;
16 }
17 int main() {
18     ...
19     printf("%lld\n", solve(1, n));
20 }

```

## 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 }

```

```

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  $S_2$

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

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 50005
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) -> 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
50     void dfs3(int u, int n) {
51         this->visited[u] = true;
52         for (int v: this->G[u]) {
53             if (!visited[v]) {
54                 this->res[v] = this->res[u] +
55                     n - 2 *
56                     this->subtreeSize[v];
57                 this->dfs3(v, n);
58             }
59         }
60     }
61 }

```

## 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;

```

```

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

```