

Contents

1	math	1
1.1	公式	1
1.2	矩陣快速幂	1
1.3	質數與因數	1
1.4	Pisano Period	2
1.5	歐拉函數	2
1.6	乘法逆元、組合數	2
1.7	大步小步	2
1.8	高斯消去	3
2	字串	3
2.1	最長迴文字串	3
2.2	KMP	3
2.3	Z Algorithm	4
3	algorithm	4
3.1	三分搜	4
3.2	差分	4
3.3	greedy	4
3.4	dinic	5
3.5	SCC Tarjan	6
3.6	SCC Kosaraju	6
3.7	ArticulationPoints Tarjan	6
3.8	最小樹狀圖	6
3.9	二分圖最大匹配	7
3.10	Blossom Algorithm	7
3.11	Astar	7
3.12	JosephusProblem	8
3.13	KM	8
3.14	LCA 倍增法	8
3.15	LCA 樹壓平 RMQ	9
3.16	LCA 樹鍊剖分	9
3.17	MCMF	10
3.18	莫隊	10
3.19	Dancing Links	11
4	DataSetructure	11
4.1	BIT	11
4.2	ChthollyTree	11
4.3	線段樹 1D	12
4.4	線段樹 2D	12
4.5	權值線段樹	13
4.6	Trie	13
4.7	AC Trie	13
4.8	單調隊列	14
5	Geometry	14
5.1	公式	14
5.2	Template	14
5.3	Polygon	15
5.4	Intersection	15
5.5	最小圓覆蓋	15
5.6	旋轉卡尺	15
5.7	凸包	15
5.8	半平面相交	15
6	DP	16
6.1	以價值為主的背包	16
6.2	抽屜	16
6.3	Barcode	16
6.4	Deque 最大差距	16
6.5	LCS 和 LIS	16
6.6	RangeDP	16
6.7	stringDP	17
6.8	樹 DP 有幾個 path 長度為 k	17
6.9	TreeDP reroot	17
6.10	WeightedLIS	17

1 math

1.1 公式

1. Most Divisor Number

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. Catalan Number

$$C_n = \frac{1}{n} \binom{2n}{n}, C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

$$C = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \dots$$

3. 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：

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

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

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

$$A = \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \\ 0 \end{bmatrix}, \quad \sum_{k=1}^n k^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

4. Lagrange Polynomial

拉格朗日插值法：找出 n 次多項函數 $f(x)$ 的點

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

$$L(x) = \sum_{j=0}^n y_j l_j(x)$$

$$l_j(x) = \prod_{i=0, i \neq j}^n \frac{x - x_i}{x_j - x_i}$$

5. SG Function

$$\begin{aligned} SG(x) &= \text{mex}\{SG(y) \mid x \rightarrow y\} \\ \text{mex}(S) &= \min\{n \mid n \in \mathbb{N}, n \notin S\} \end{aligned}$$

6. Fibonacci

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

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

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right]$$

7. Pick's Theorem

給定頂點座標均是整點（或正方形格子點）的簡單多邊形：

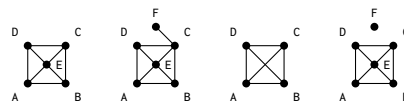
其面積 A 和內部格點數目 i 、邊上格點數目 b 的關係為

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

8. Euler's Formula

對於有 V 個點、 E 條邊、 F 個面 (含外部) 的連通平面圖

$$F + V - E = 2$$



(1)、(2) ○; (3) ×, \overline{AC} 與 \overline{BD} 相交; (4) ×, 非連通圖

9. Simpson Integral

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$

1.2 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(), vector<ll>(B[0].size()));
7     for(int i=0; i<A.size(); i++) {
8         for(int j=0; j<B[0].size(); j++) {
9             for(int k=0; k<B.size(); k++) {
10                 res[i][j] += A[i][k] * B[k][j] % mod;
11                 res[i][j] %= mod;
12             }
13         }
14     }
15     return res;
16 }
17
18 mat I = ;
19 // compute matrix M^n
20 // 需先 init I 矩阵
21 mat mpow(mat& M, int n) {
22     if(n <= 1) return n ? M : I;
23     mat v = mpow(M, n>>1);
24     return (n & 1) ? v*v*M : v*v;
25 }
26
27 // 迴圈版本
28 mat mpow(mat M, int n) {
29     mat res(M.size(), vector<ll>(M[0].size()));
30     for(int i=0; i<res.size(); i++)
31         res[i][i] = 1;
32     for(; n; n>>=1) {
33         if(n & 1) res = res * M;
34         M = M * M;
35     }
36     return res;
37 }

```

1.3 質數與因數

```

1  欧拉筛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 最大公因数 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 质因数分解
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 擴展歐幾里得算法 ax + by = GCD(a, b)
36
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
47 int main(){
48     int a, b, x, y;
49     cin >> a >> b;
50     ext_euc(a, b, x, y);
51     cout << x << ' ' << y << endl;
52     return 0;
53 }
54
55 歌德巴赫猜想
56 解：把偶數 N (6≤N≤10^6) 寫成兩個質數的 和。
57 #define N 2000000
58 int ox[N], p[N], pr;
59 void PrimeTable(){
60     ox[0] = ox[1] = 1;
61     pr = 0;
62     for(int i=2;i<N;i++){
63         if(!ox[i]) p[pr++] = i;
64         for(int j=0; i*p[j]<N&&j<pr; j++)
65             ox[i*p[j]] = 1;
66     }
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
95 int main(){
96     int n;
97     cin>>n;
98     if(isPrime(n)) cout<<"1\n";
99     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
100    else cout<<"3\n";

```

1.4 Pisano Period

```

1 #include <stdio>
2 #include <vector>
3 using namespace std;
4
5 /*
6  Pisano Period + 快速幂 + mod
7  Pisano Period:
8      費氏數列在mod n的情況下會有循環週期，

```

```

9      且週期的結束判斷會在fib[i - 1] == 0 &&
10         fib[i] == 1時，
11      此時循環週期長度是i - 1
12
13 所以這題是在找出循環週期後，
14 用快速幂並mod(循環週期長度)即可AC(快速幂記得mod)，
15 此外fib要mod n，也要找週期，所以用預處理的方式列表
16 */
17 #define maxn 1005
18
19 /*
20  Pisano period可證一個週期的長度會在[n, n ^ n]之間
21 */
22 //很可惜，會爆
23 // int fib[maxn][maxn * maxn];
24 //改用vector
25 vector<int> fib[maxn];
26 int period[maxn];
27
28 int qpow(int a, unsigned long long b, int
29         mod)
30 {
31     if (b == 0)
32         return a;
33     long long res = 1;
34     while (b)
35     {
36         if (b & 1)
37             res = ((a % mod) * (res % mod)) % mod;
38         a = ((a % mod) * (a % mod)) % mod;
39         b >>= 1;
40     }
41     return res;
42 }
43
44 int main()
45 {
46     int t;
47     unsigned long long a, b;
48     int n;
49
50     //注意：這裡沒算mod 1的循環長度，
51     //因為mod 1都等於0，沒有週期
52     for (int i = 2; i < maxn; ++i)
53     {
54         fib[i].emplace_back(0);
55         fib[i].emplace_back(1);
56         for (int j = 2; j < maxn * maxn; ++j)
57         {
58             fib[i].emplace_back(
59                 (fib[i][j-1]*fib[i][j-2]%i)%i
60             );
61             if (fib[i][j-1]==0&&fib[i][j]==1)
62             {
63                 period[i] = j - 1;
64                 break;
65             }
66         }
67     }
68
69     scanf("%d", &t);
70
71     while (t--)
72     {
73         scanf("%llu %llu %d", &a, &b, &n);
74         if (a == 0)
75             puts("0");
76         else if (n == 1) //當mod 1時任何數都是0，
77             //所以直接輸出0，避免我們沒算
78             //fib[1][i]的問題(Runtime
79             error)
80             printf("%d\n",
81                 fib[n][qpow(a % period[n], b,
82                     period[n])]);
83         return 0;
84     }
85 }

```

```

82 }
83
84 //計算閉區間 [1,n] 中有幾個正整數與 n 互質
85
86 int phi(){
87     int ans=n;
88     for(int i=2;i*i<=n;i++){
89         if(n%i==0){
90             ans=ans-ans/i;
91             while(n%i==0) n/=i;
92         }
93     }
94     if(n>1) ans=ans-ans/n;
95     return ans;
96 }

```

1.5 歐拉函數

1.6 乘法逆元、組合數

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

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

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
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 }

```

1.7 大步小步

1 題意
 2 給定 B, N, P , 求出 L 滿足 $B^L \equiv N \pmod{P}$ 。
 3 題解
 4 餘數的循環節長度必定為 P 的因數, 因此
 $B^0, B^1, B^2, \dots, B^{(P+1)}, \dots$,
 5 也就是說如果有解則 $L < N$, 枚舉 $0, 1, 2, L-1$
 能得到結果, 但會超時。
 6 將 L 拆成 $mx+y$, 只要分別枚舉 x, y 就能得到答案,
 7 設 $m=\sqrt{P}$ 能保證最多枚舉 $2\sqrt{P}$ 次。
 8 $B^{mx+y} \equiv N \pmod{P}$
 9 $B^{mx} B^y \equiv N \pmod{P}$
 10 $B^y \equiv N(B^{(-m)})^x \pmod{P}$
 11 先求出 $B^0, B^1, B^2, \dots, B^{(m-1)}$,
 12 再枚舉 $N(B^{(-m)}), N(B^{(-m)})^2, \dots$ 查看是否有對應的
 B^y 。
 13 這種算法稱為大步小步演算法,
 14 大步指的是枚舉 x (一次跨 m 步),
 15 小步指的是枚舉 y (一次跨 1 步)。
 16 複雜度分析
 17 利用 `map/unordered_map` 存放
 $B^0, B^1, B^2, \dots, B^{(m-1)}$,
 18 枚舉 x 查詢 `map/unordered_map` 是否有對應的 B^y ,
 19 存放和查詢最多 $2\sqrt{P}$ 次, 時間複雜度為
 $O(\sqrt{P} \log \sqrt{P}) / O(\sqrt{P})$ 。

```

21 using LL = long long;
22 LL B, N, P;
23 LL fpow(LL a, LL b, LL c){
24     LL res=1;
25     for(; b >= 1;){
26         if(b&1)
27             res=(res*a)%c;
28         a=(a*a)%c;
29     }
30     return res;
31 }
32 LL BSGS(LL a, LL b, LL p){
33     a%=p, b%=p;
34     if(a==0)
35         return b==0?1:-1;
36     if(b==1)
37         return 0;
38     map<LL, LL> tb;
39     LL sq=ceil(sqrt(p-1));
40     LL inv=fpow(a, p-sq-1, p);
41     tb[1]=sq;
42     for(LL i=1, tmp=1; i<sq; ++i){
43         tmp=(tmp*a)%p;
44         if(!tb.count(tmp))
45             tb[tmp]=i;
46     }
47     for(LL i=0; i<sq; ++i){
48         if(tb.count(b)){
49             LL res=tb[b];
50             return i*sq+(res==sq?0:res);
51         }
52         b=(b*inv)%p;
53     }
54     return -1;
55 }
56 int main(){
57     IOS; //輸入優化
58     while(cin>>P>>B>>N){
59         LL ans=BSGS(B, N, P);
60         if(ans!=-1)
61             cout<<"no solution\n";
62         else
63             cout<<ans<<"\n";
64     }
65 }

```

1.8 高斯消去

• 計算 $AX = B$

• 傳入：

增廣矩陣 $M = [A|B]$

equ = 有幾個 equation

var = 有幾個 variable

• 回傳： $X = (x_0, \dots, x_{n-1})$ 的解集

• ! 無法判斷無解或無限多組解!

```

1 using DBL = double;
2 using mat = vector<vector<DBL>>;
3
4 vector<DBL> Gauss(mat& M, int equ, int var) {
5     auto dcmp = [](DBL a, DBL b=0.0) {
6         return (a > b) - (a < b);
7     };
8
9     for(int r=0, c=0; r<equ && c<var; ) {
10         int mx = r; // 找絕對值最大的 M[i][c]
11         for(int i=r+1; i<equ; i++) {
12             if(dcmp(abs(M[i][c]), abs(M[mx][c]))>=1)
13                 mx = i;
14         }
15         if(mx != r) swap(M[mx], M[r]);
16
17         if(dcmp(M[r][c]) == 0) {
18             c++;
19             continue;
20         }
21
22         for(int i=r+1; i<equ; i++) {
23             if(dcmp(M[i][c]) == 0) continue;
24             DBL t = M[i][c] / M[r][c];
25             for(int j=c; j<M[c].size(); j++) {
26                 M[i][j] -= t * M[r][j];
27             }
28         }
29         r++, c++;
30     }
31
32     vector<DBL> X(var);
33     for(int i=var-1; i>=0; i--) {
34         X[i] = M[i][var];
35         for(int j=var-1; j>i; j--) {
36             X[i] -= M[i][j] * X[j];
37         }
38         X[i] /= M[i][i];
39     }
40     return X;
41 }

```

2 字串

2.1 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=l;
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=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 }

```

2.2 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 = ababababababababab
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 }

```

2.3 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     }
16 }

```

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;
45         d1=(y1-x1)/time;
46         d2=(y2-x2)/time;
47         double L=0,R=1e8,m1,m2,f1,f2;
48         double ans = x1.dist(x2);
49         while(abs(L-R)>1e-10){
50             m1=(L+R)/2;
51             m2=(m1+R)/2;
52             f1=((d1*m1)+x1).dist((d2*m1)+x2);
53             f2=((d1*m2)+x1).dist((d2*m2)+x2);
54             ans = min(ans,min(f1,f2));
55             if(f1<f2) R=m2;
56             else L=m1;
57         }
58         cout<<"Case "<<ti<<": ";
59         cout << fixed << setprecision(4) <<
60             sqrt(ans) << '\n';
61     }
62 }

```

3.2 差分

```

1 用途：在區間 [l, r] 加上一個數字v。
2 b[l] += v; (b[0~l] 加上v)
3 b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v) )
4 給的 a[] 是前綴和數列，建構 b[]，
5 因為 a[i] = b[0] + b[1] + b[2] + ... + b[i]，
6 所以 b[i] = a[i] - a[i-1]。

```

```

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

```

3.3 greedy

```

1 刪數字問題
2 //problem
3 給定一個數字 N(≤10^100)，需要刪除 K 個數字，
4 請問刪除 K 個數字後最小的數字為何？
5 //solution
6 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i
7 位數，
8 扣除高位數的影響較扣除低位數的大。
9 //code
10 int main(){
11     string s;
12     int k;
13     cin>>s>>k;
14     for(int i=0;i<k;++i){
15         if((int)s.size()==0) break;
16         int pos =(int)s.size()-1;
17         for(int j=0;j<(int)s.size()-1;++j){
18             if(s[j]>s[j+1]){
19                 pos=j;
20                 break;
21             }
22         }
23         s.erase(pos,1);
24     }
25     while((int)s.size()>0&&s[0]=='0')
26         s.erase(0,1);
27     if((int)s.size()) cout<<s<<'\n';
28     else cout<<<0<<'\n';
29 }
30 最小區間覆蓋長度
31 //problem
32 給定 n 條線段區間為 [Li,Ri]，
33 請問最少要選幾個區間才能完全覆蓋 [0,S]？
34 //solution
35 先將所有區間依照左界由小到大排序，
36 對於當前區間 [Li,Ri]，要從左界 >Ri 的所有區間中，
37 找到有著最大的右界的區間，連接當前區間。
38 //problem
39 長度 n 的直線中有數個加熱器，
40 在 x 的加熱器可以讓 [x-r,x+r] 內的物品加熱，
41 問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
42 //solution
43 對於最左邊沒加熱的點a，選擇最遠可以加熱a的加熱器，
44 更新已加熱範圍，重複上述動作繼續尋找加熱器。
45 //code
46 int main(){
47     int n, r;
48     int a[1005];
49     cin>>n>>r;
50     for(int i=1;i<=n;++i) cin>>a[i];
51     int i=1,ans=0;
52     while(i<=n){
53         int R=min(i+r-1,n),L=max(i-r+1,0)

```

```

54     int nextR=-1;
55     for(int j=R;j>=L;--j){
56         if(a[j]){
57             nextR=j;
58             break;
59         }
60     }
61     if(nextR!=-1){
62         ans=-1;
63         break;
64     }
65     ++ans;
66     i=nextR+r;
67 }
68 cout<<ans<<'\\n';
69 }
70 最多不重疊區間
71 //problem
72 給你 n 條線段區間為 [Li,Ri],
73 請問最多可以選擇幾條不重疊的線段(頭尾可相連)?
74 //solution
75 依照右界由小到大排序,
76 每次取到一個不重疊的線段, 答案 +1。
77 //code
78 struct Line{
79     int L,R;
80     bool operator<(const Line &rhs)const{
81         return R<rhs.R;
82     }
83 };
84 int main(){
85     int t;
86     cin>>t;
87     Line a[30];
88     while(t--){
89         int n=0;
90         while(cin>>a[n].L>>a[n].R,a[n].L||a[n].R){
91             ++n;
92         }
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 maxLi 最小。
111 //solution
112 按照到期時間從早到晚處理。
113 //code
114 struct Work{
115     int t, d;
116     bool operator<(const Work &rhs)const{
117         return d<rhs.d;
118     }
119 };
120 int main(){
121     int n;
122     Work a[10000];
123     cin>>n;
124     for(int i=0;i<n;++i)
125         cin>>a[i].t>>a[i].d;
126     sort(a,a+n);
127     int maxL=0,sumT=0;
128     for(int i=0;i<n;++i){
129         sumT+=a[i].t;
130         maxL=max(maxL,sumT-a[i].d);
131     }
132     cout<<maxL<<'\\n';
133 }
134 }
135 最少延遲數量問題
136 //problem
137 給定 N 個工作, 每個工作的需要處理時長為 Ti,
138 期限是 Di, 求一種工作排序使得逾期工作數量最小。
139 //solution
140 期限越早到期的工作越先做。
141 將工作依照到期時間從早到晚排序,
142 依序放入工作列表中, 如果發現有工作預期,
143 就從目前選擇的工作中, 移除耗時最長的工作。
144 上述方法為 Moore-Hodgson s Algorithm。
145 //problem
146 給定烏龜的重量和可承受重量, 問最多可以疊幾隻烏龜?
147 //solution
148 和最少延遲數量問題是相同的問題, 只要將題敘做轉換。
149 工作處理時長 → 烏龜重量
150 工作期限 → 烏龜可承受重量
151 多少工作不延期 → 可以疊幾隻烏龜
152 //code
153 struct Work{
154     int t, d;
155     bool operator<(const Work &rhs)const{
156         return d<rhs.d;
157     }
158 };
159 int main(){
160     int n=0;
161     Work a[10000];
162     priority_queue<int> pq;
163     while(cin>>a[n].t>>a[n].d){
164         ++n;
165     }
166     sort(a,a+n);
167     int sumT=0,ans=n;
168     for(int i=0;i<n;++i){
169         pq.push(a[i].t);
170         sumT+=a[i].t;
171         if(a[i].d<sumT){
172             int x=pq.top();
173             pq.pop();
174             sumT-=x;
175             --ans;
176         }
177     }
178     cout<<ans<<'\\n';
179 }
180 }
181 任務調度問題
182 //problem
183 給定 N 項工作, 每項工作的需要處理時長為 Ti,
184 期限是 Di, 如果第 i 項工作延遲需要受到 pi
185     單位懲罰,
186 請問最少會受到多少單位懲罰。
187 //solution
188 依照懲罰由大到小排序,
189 每項工作依序嘗試可不可以放在
190     Di-Ti+1,Di-Ti,...,1,0,
191 如果有空閒就放進去, 否則延後執行。
192 //problem
193 給定 N 項工作, 每項工作的需要處理時長為 Ti,
194 期限是 Di, 如果第 i 項工作在期限內完成會獲得 ai
195     單位獎勵,
196 請問最多會獲得多少單位獎勵。
197 //solution
198 和上題相似, 這題變成依照獎勵由大到小排序。
199 //code
200 struct Work{
201     int d,p;
202     bool operator<(const Work &rhs)const{
203         return p>rhs.p;
204     }
205 };
206 int main(){
207     int n;
208     Work a[100005];
209     bitset<100005> ok;
210     while(cin>>n){
211         ok.reset();
212         for(int i=0;i<n;++i)
213             cin>>a[i].d>>a[i].p;
214         sort(a,a+n);
215         int ans=0;
216         for(int i=0;i<n;++i){
217             int j=a[i].d;
218             while(j--){
219                 if(!ok[j]){
220                     ans+=a[i].p;
221                     ok[j]=true;
222                     break;
223                 }
224             }
225         }
226         cout<<ans<<'\\n';
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 }

```

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     }
38     return level[T];
39 }
40 int dfs(int cur, int lim) {
41     if(cur==T || lim==0) return lim;
42     int result = 0;
43     for(int& i=dfs_idx[cur]; i<G[cur].size()
44         && lim; i++) {
45         Edge& e = E[G[cur][i]];
46         if(level[e.s]+1 != level[e.t])
47             continue;
48         int flow = dfs(e.t, min(lim,
49             e.cap-e.flow));
50         if(flow <= 0) continue;
51         e.flow += flow;
52         result += flow;
53         E[G[cur][i]^1].flow -= flow;
54         lim -= flow;
55     }
56     return result;
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
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 }

```



```

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 }

```

```

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 }

```

3.5 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);

```

3.6 SCC Kosaraju

```

1 //做兩次dfs, O(V + E)
2 //g 是原圖, g2 是反圖
3 //s是dfs離開的節點
4 void dfs1(int u) {
5     vis[u] = true;
6     for (int v: g[u])
7         if (!vis[v]) dfs1(v);
8     s.push_back(u);
9 }
10 void dfs2(int u) {
11     group[u] = sccCnt;
12     for (int v: g2[u])
13         if (!group[v]) dfs2(v);
14 }
15 }
16 void kosaraju() {
17     sccCnt = 0;
18     for (int i = 1; i <= n; ++i)
19         if (!vis[i]) dfs1(i);
20     for (int i = n; i >= 1; --i)
21         if (!group[s[i]]) {
22             ++sccCnt;
23             dfs2(s[i]);
24         }
25 }
26 }

```

3.7 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 }
32 int main() {

```

```

32 char input[105];
33 char* token;
34 while (scanf("%d", &N) != EOF && N) {
35     G.assign(105, vector<int>());
36     memset(visited, false,
37           sizeof(visited));
38     memset(low, 0, sizeof(low));
39     memset(dfn, 0, sizeof(dfn));
40     timer = 0;
41     res = 0;
42     while (fgetc(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             v = atoi(token);
53             G[u].emplace_back(v);
54             G[v].emplace_back(u);
55         }
56     }
57     tarjan(1, -1);
58     printf("%d\n", res);
59 }
60 return 0;
61 }

```

3.8 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                 idx[cur]=i;
41                 if(cur == root) break;
42                 vis[cur] = i;
43                 cur = pre[cur];

```

```

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

```

3.9 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)) ++res;
26     }
27     cout << n - res << '\n';
28 }

```

3.10 Blossom Algorithm

```

1  const int maxn = 500 + 10;
2
3  struct Edge { int s, t; };
4
5  int n;
6  int base[maxn], match[maxn], p[maxn], inq[maxn];
7  bool vis[maxn], flower[maxn];
8  vector<Edge> G[maxn];
9  queue<int> q;
10
11 int lca(int a, int b) {
12     memset(vis, 0, sizeof(vis));
13     while(1) {
14         a = base[a];
15         vis[a] = true;

```

```

16         if(match[a] == -1) break;
17         a = p[match[a]];
18     }
19     while(1) {
20         b = base[b];
21         if(vis[b]) return b;
22         b = p[match[b]];
23     }
24     return -1;
25 }
26
27 void set_path(int x, int father) {
28     int tmp;
29     while(x != father) {
30         tmp = match[x];
31         flower[base[x]] = flower[base[tmp]] = 1;
32         tmp = p[tmp];
33         if(base[tmp] != father) p[tmp] = match[x];
34         x = tmp;
35     }
36 }
37
38 void blossom(int x, int y) {
39     memset(flower, 0, sizeof(flower));
40     int father = lca(x, y);
41     set_path(x, father);
42     set_path(y, father);
43     if(base[x] != father) p[x] = y;
44     if(base[y] != father) p[y] = x;
45     for(int i=1; i<=n; i++) {
46         if(!flower[base[i]]) continue;
47         base[i] = father;
48         if(!inq[i]) {
49             q.push(i);
50             inq[i] = true;
51         }
52     }
53 }
54
55 bool bfs(int root) {
56     int cur, y, nxt;
57     q = queue<int>();
58     q.push(root);
59     memset(inq, 0, sizeof(inq));
60     memset(p, -1, sizeof(p));
61     for(int i=1; i<=n; i++) base[i] = i;
62
63     while(!q.empty()) {
64         cur = q.front();
65         q.pop();
66         inq[cur] = false;
67
68         for(auto e : G[cur]) {
69             if(base[e.s] == base[e.t]) continue;
70             if(match[e.s] == e.t) continue;
71             if(e.t == root ||
72                (~match[e.t] && ~p[match[e.t]])) {
73                 blossom(cur, e.t);
74             } else if(p[e.t] == -1) {
75                 p[e.t] = cur;
76                 if(match[e.t] == -1) {
77                     cur = e.t;
78                     while(cur != -1) {
79                         y = p[cur];
80                         nxt = match[y];
81                         match[cur] = y;
82                         match[y] = cur;
83                         cur = nxt;
84                     }
85                     return true;
86                 } else {
87                     q.push(match[e.t]);
88                     inq[match[e.t]] = true;
89                 }
90             }
91         }
92     }
93     return false;

```

```

94 }
95
96 int maxMatch() {
97     int res = 0;
98     memset(match, -1, sizeof(match));
99     for(int i=1; i<=n; i++) {
100         if(match[i]==-1 && bfs(i)) res++;
101     }
102     return res;
103 }

```

3.11 Astar

```

1  /*A*求k短路
2      $f(x) = g(x) + h(x)$ 
3      $g(x)$  是實際cost,  $h(x)$  是估計cost
4     在此 $h(x)$ 用所有點到終點的最短距離, 則當用Astar找點
5     當該點 $cnt[u] == k$ 時即得到該點的第k短路
6  */
7  #define maxn 105
8  struct Edge {
9      int u, v, w;
10 };
11 struct Item_pqH {
12     int u, w;
13     bool operator < (const Item_pqH& other) const {
14         return this->w > other.w;
15     }
16 };
17 struct Item_astar {
18     int u, g, f;
19     bool operator < (const Item_astar& other) const {
20         return this->f > other.f;
21     }
22 };
23 vector<vector<Edge>> G;
24 //反向圖, 用於建 $h(u)$ 
25 vector<vector<Edge>> invertG;
26 int h[maxn];
27 bool visited[maxn];
28 int cnt[maxn];
29 //用反向圖去求出每一點到終點的最短距離, 並以此當作 $h(u)$ 
30 void dijkstra(int s, int t) {
31     memset(visited, 0, sizeof(visited));
32     priority_queue<Item_pqH> pq;
33     pq.push({s, 0});
34     h[s] = 0;
35     while (!pq.empty()) {
36         Item_pqH curr = pq.top();
37         pq.pop();
38         visited[curr.u] = true;
39         for (Edge& edge: invertG[curr.u]) {
40             if (!visited[edge.v]) {
41                 if (h[edge.v] > h[curr.u] + edge.w) {
42                     h[edge.v] = h[curr.u] + edge.w;
43                     pq.push({edge.v, h[edge.v]});
44                 }
45             }
46         }
47     }
48 }
49 int Astar(int s, int t, int k) {
50     memset(cnt, 0, sizeof(cnt));
51     priority_queue<Item_astar> pq;
52     pq.push({s, 0, h[s]});
53     while (!pq.empty()) {
54         Item_astar curr = pq.top();
55         pq.pop();
56         ++cnt[curr.u];
57         //終點出現k次, 此時即可得k短路
58         if (cnt[t] == k)
59             return curr.g;

```

```

60     for (Edge& edge: G[curr.u]) {
61         if (cnt[edge.v] < k) {
62             pq.push({edge.v, curr.g +
63                 edge.w, curr.g + edge.w
64                 + h[edge.v]});
65         }
66     }
67 }
68 int main() {
69     int n, m;
70     while (scanf("%d %d", &n, &m) && (n != 0
71         && m != 0)) {
72         G.assign(n + 5, vector<Edge>());
73         invertG.assign(n + 5, vector<Edge>());
74         int s, t, k;
75         scanf("%d %d %d", &s, &t, &k);
76         int u, v, w;
77         for (int i = 0; i < m; ++i) {
78             scanf("%d %d %d", &u, &v, &w);
79             G[u].emplace_back(Edge{u, v, w});
80             invertG[v].emplace_back(Edge{v,
81                 u, w});
82         }
83         memset(h, 0x3f, sizeof(h));
84         dijkstra(t, s);
85         printf("%d\n", Astar(s, t, k));
86     }
87     return 0;
88 }

```

3.12 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 // O(n)
5 int getWinner(int n, int k) {
6     int winner = 0;
7     for (int i = 1; i <= n; ++i)
8         winner = (winner + k) % i;
9     return winner;
10 }
11
12 int main() {
13     int n;
14     while (scanf("%d", &n) != EOF && n){
15         --n;
16         for (int k = 1; k <= n; ++k){
17             if (getWinner(n, k) == 11){
18                 printf("%d\n", k);
19                 break;
20             }
21         }
22     }
23     return 0;
24 }
25
26 // O(k log(n))
27 int josephus(int n, int k) {
28     if (n == 1) return 0;
29     if (k == 1) return n - 1;
30     if (k > n) return (josephus(n-1,k)+k)%n;
31     int res = josephus(n - n / k, k);
32     res -= n % k;
33     if (res < 0)
34         res += n; // mod n
35     else
36         res += res / (k - 1); // 还原位置
37     return res;
38 }
39 }

```

3.13 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         for (int i = 0; i < n; ++i) {
42             if (S[i]) Lx[i] -= diff;
43             if (T[i]) Ly[i] += diff;
44         }
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)

```

```

74             printf(" %d", Lx[i]);
75         else
76             printf(" %d", Lx[i]);
77         res += Lx[i];
78     }
79     puts("");
80     for (int i = 0; i < n; ++i) {
81         if (i != 0)
82             printf(" %d", Ly[i]);
83         else
84             printf(" %d", Ly[i]);
85         res += Ly[i];
86     }
87     puts("");
88     printf("%d\n", res);
89 }
90 return 0;
91 }

```

3.14 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]
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     // 最後發現不能跳了, 此時 x 的第 2^0 =
54     // 1 個祖先 (或說 y 的第 2^0 =
55     // 1 的祖先) 即為 x、y 的 lca
56     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 }

```

3.15 LCA 樹壓平 RMQ

```

1 //樹壓平求LCA RMQ(sparse table
   O(nlogn)建立, O(1)查詢, 求任意兩點距離,
2 //如果用笛卡兒樹可以壓到O(n)建立, O(1)查詢
3 //理論上可以過, 但遇到直鏈的case dfs深度會stack
  overflow
4 #define maxn 100005
5 struct Edge {
6     int u, v, w;
7 };
8 int dep[maxn], pos[maxn];
9 long long dis[maxn];
10 int st[maxn * 2][32]; //sparse table
11 int realLCA[maxn * 2][32];
   //最小深度對應的節點, 及真正的LCA
12 int Log[maxn]; //取代std::log2
13 int tp; // timestamp
14 vector<vector<Edge>> G; // tree
15 void calLog() {
16     Log[1] = 0;
17     Log[2] = 1;
18     for (int i = 3; i < maxn; ++i)
19         Log[i] = Log[i / 2] + 1;
20 }
21 void buildST() {
22     for (int j = 0; Log[tp]; ++j) {
23         for (int i = 0; i + (1 << j) - 1 < tp;
           ++i) {
24             if (st[i - 1][j] < st[i - 1][j + (1 <<
               i - 1)]) {
25                 st[i][j] = st[i - 1][j];
26                 realLCA[i][j] = realLCA[i - 1][j];
27             }
28             else {
29                 st[i][j] = st[i - 1][j + (1 << i -
                   1)];
30                 realLCA[i][j] = realLCA[i - 1][j + (1
                   << i - 1)];
31             }
32         }
33     }
34 } // O(nlogn)
35 int query(int l, int r) { // [l, r] min
   depth即為lca的深度
36     int k = Log[r - l + 1];
37     if (st[l][k] < st[r - (1 << k) + 1][k])
38         return realLCA[l][k];
39     else
40         return realLCA[r - (1 << k) + 1][k];
41 }
42 void dfs(int u, int p) { //euler tour
43     pos[u] = tp;

```

```

44     st[tp][0] = dep[u];
45     realLCA[tp][0] = dep[u];
46     ++tp;
47     for (int i = 0; i < G[u].size(); ++i) {
48         Edge& edge = G[u][i];
49         if (edge.v == p) continue;
50         dep[edge.v] = dep[u] + 1;
51         dis[edge.v] = dis[edge.u] + edge.w;
52         dfs(edge.v, u);
53         st[tp++][0] = dep[u];
54     }
55 }
56 long long getDis(int u, int v) {
57     if (pos[u] > pos[v])
58         swap(u, v);
59     int lca = query(pos[u], pos[v]);
60     return dis[u] + dis[v] - 2 *
       dis[query(pos[u], pos[v])];
61 }
62 int main() {
63     int n, q;
64     calLog();
65     while (~scanf("%d", &n) && n) {
66         int v, w;
67         G.assign(n + 5, vector<Edge>());
68         tp = 0;
69         for (int i = 1; i <= n - 1; ++i) {
70             scanf("%d %d", &v, &w);
71             G[i].push_back({i, v, w});
72             G[v].push_back({v, i, w});
73         }
74         dfs(0, -1);
75         buildST();
76         scanf("%d", &q);
77         int u;
78         while (q--) {
79             scanf("%d %d", &u, &v);
80             printf("%lld%c", getDis(u, v),
               (q) ? ' ' : '\n');
81         }
82     }
83     return 0;
84 }

```

3.16 LCA 樹鍊剖分

```

1 #define maxn 5005
2 //LCA, 用來練習樹鍊剖分
3 //題意: 給定樹, 找兩點的中點,
4 //若中點不存在(路徑為even), 就是中間的兩個點
5 int dfn[maxn];
6 int parent[maxn];
7 int depth[maxn];
8 int subtreeSize[maxn];
9 //樹鍊的頂點
10 int top[maxn];
11 //將dfn轉成node編碼
12 int dfnToNode[maxn];
13 //重兒子
14 int hson[maxn];
15 int dfsTime = 1;
16 //tree
17 vector<vector<int>> G;
18 //處理parent、depth、subtreeSize、dfnToNode
19 void dfs1(int u, int p) {
20     parent[u] = p;
21     hson[u] = -1;
22     subtreeSize[u] = 1;
23     for (int v: G[u]) {
24         if (v != p) {
25             depth[v] = depth[u] + 1;
26             dfs1(v, u);
27             subtreeSize[u] += subtreeSize[v];
28             if (hson[u] == -1 ||
               subtreeSize[hson[u]] <
               subtreeSize[v]) {
29                 hson[u] = v;

```

```

30     }
31 }
32 }
33 }
34 //實際剖分 <- 參數t是top的意思
35 //t初始應為root本身
36 void dfs2(int u, int t) {
37     top[u] = t;
38     dfn[u] = dfsTime;
39     dfnToNode[dfn[u]] = u;
40     ++dfsTime;
41     //葉子點 -> 沒有重兒子
42     if (hson[u] == -1)
43         return;
44     //優先對重兒子dfs, 才能保證同一重鍊dfn連續
45     dfs2(hson[u], t);
46     for (int v: G[u]) {
47         if (v != parent[u] && v != hson[u])
48             dfs2(v, v);
49     }
50 }
51 //不斷跳鍊, 當跳到同一條鍊時, 深度小的即為LCA
52 //跳鍊時優先鍊頂深度大的跳
53 int LCA(int u, int v) {
54     while (top[u] != top[v]) {
55         if (depth[top[u]] > depth[top[v]])
56             u = parent[top[u]];
57         else
58             v = parent[top[v]];
59     }
60     return (depth[u] > depth[v]) ? v : u;
61 }
62 int getK_parent(int u, int k) {
63     while (k-- && (u != -1))
64         u = parent[u];
65     return u;
66 }
67 int main() {
68     int n;
69     while (scanf("%d", &n) && n) {
70         dfsTime = 1;
71         G.assign(n + 5, vector<int>());
72         int u, v;
73         for (int i = 1; i < n; ++i) {
74             scanf("%d %d", &u, &v);
75             G[u].emplace_back(v);
76             G[v].emplace_back(u);
77         }
78         dfs1(1, -1);
79         dfs2(1, 1);
80         int q;
81         scanf("%d", &q);
82         for (int i = 0; i < q; ++i) {
83             scanf("%d %d", &u, &v);
84             //先得到LCA
85             int lca = LCA(u, v);
86             //計算路徑長(經過的邊)
87             int dis = depth[u] + depth[v] - 2
               * depth[lca];
88             //讓v比u深或等於
89             if (depth[u] > depth[v])
90                 swap(u, v);
91             if (u == v) {
92                 printf("The fleas meet at
                   %d.\n", u);
93             }
94             else if (dis % 2 == 0) {
95                 //路徑長是even -> 有中點
96                 printf("The fleas meet at
                   %d.\n", getK_parent(v,
                   dis / 2));
97             }
98             else {
99                 //路徑長是odd -> 沒有中點
100                 if (depth[u] == depth[v]) {
101                     int x = getK_parent(u, dis
                       / 2);

```

```

102         int y = getK_parent(v, dis / 2);
103         if (x > y)
104             swap(x, y);
105         printf("The fleas jump forever between %d and %d.\n", x, y);
106     }
107     else {
108         //技巧: 讓深的點v往上dis / 2步 = y,
109         //這個點的parent設為x
110         //此時的x、y就是答案要的中點兩點
111         //主要是往下不好找, 所以改用深的點
112         int y = getK_parent(v, dis / 2);
113         int x = getK_parent(y, 1);
114         if (x > y)
115             swap(x, y);
116         printf("The fleas jump forever between %d and %d.\n", x, y);
117     }
118 }
119 }
120 }
121 return 0;
122 }

```

3.17 MCMF

```

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

```

```

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

```

3.18 莫隊

```

1 /*利用prefix前綴XOR和
2 如果要求[x, y]的XOR和只要回答prefix[y] ^ prefix[x - 1]即可在O(1)回答
3 同時維護cnt[i]代表[x, y]XOR和 == i的個數
4 如此我們知道[l, r]可以快速知道[l - 1, r], [l + 1, r], [l, r - 1], [l, r + 1]的答案
5 就符合Mo's algorithm的思維O(N * sqrt(n))
6 每次轉移為O(1), 具體轉移方法在下面*/
7 #define maxn 100005
8 //在此prefix[i]是[1, i]的XOR和
9 int prefix[maxn];
10 //log_2(1000000) = 19.931568569324174087221916576937...
11 //所以開到1 << 20
12 //cnt[i]代表的是有符合nums[x, y] such that nums[x] ^ nums[x + 1] ^ ... ^ nums[y] == i
13 //的個數
14 long long cnt[1 << 20];
15 //塊大小 -> sqrt(n)
16 int sqrtQ;
17 struct Query {
18     int l, r, id;
19     bool operator < (const Query& other) const {
20         if (this->l / sqrtQ != other.l / sqrtQ)
21             return this->l < other.l;
22         //奇偶排序(優化)
23         if (this->l / sqrtQ & 1)
24             return this->r < other.r;
25         return this->r > other.r;
26     }
27 };
28 Query queries[maxn];
29 long long ans[maxn];
30 long long res = 0;
31 int k;
32 void add(int x) {
33     res += cnt[k ^ prefix[x]];
34     ++cnt[prefix[x]];
35 }
36 void sub(int x) {
37     --cnt[prefix[x]];
38     res -= cnt[k ^ prefix[x]];
39 }
40 int main() {
41     int n, m;
42     scanf("%d %d %d", &n, &m, &k);
43     sqrtQ = sqrt(n);
44     for (int i = 1; i <= n; ++i) {
45         scanf("%d", &prefix[i]);
46         prefix[i] ^= prefix[i - 1];
47     }
48     for (int i = 1; i <= m; ++i) {
49         scanf("%d %d", &queries[i].l, &queries[i].r);
50         //減1是因為prefix[i]是[1, i]的前綴XOR和, 所以題目問[l, r]我們要回答[l - 1, r]的答案
51         --queries[i].l;
52         queries[i].id = i;
53     }
54     sort(queries + 1, queries + m + 1);
55     int l = 1, r = 0;
56     for (int i = 1; i <= m; ++i) {
57         while (l < queries[i].l)
58             sub(l);
59         ++l;
60     }
61     while (l > queries[i].l) {
62         --l;
63         add(l);
64     }
65     while (r < queries[i].r) {
66         ++r;

```

```

67         add(r);
68     }
69     while (r > queries[i].r) {
70         sub(r);
71         --r;
72     }
73     ans[queries[i].id] = res;
74 }
75 for (int i = 1; i <= m; ++i){
76     printf("%lld\n", ans[i]);
77 }
78 return 0;
79 }

```

3.19 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]) {

```

```

57        result[idx] = row[i];
58        for(int j=R[i]; j!=i; j=R[j])
59            remove(col[j]);
60        if(dfs(idx+1)) return true;
61        for(int j=L[i]; j!=i; j=L[j])
62            recover(col[j]);
63    }
64    recover(c);
65    return false;
66 }
67 void dfs(int idx=0) { // 判斷最小 dfs
68     depth 版
69     if(R[0] == 0) {
70         resSize = min(resSize, idx); //
71         注意init值
72         return;
73     }
74     int c = R[0];
75     for(int i=R[0]; i; i=R[i]) {
76         if(colSize[i] < colSize[c]) c = i;
77     }
78     remove(c);
79     for(int i=D[c]; i!=c; i=D[i]) {
80         for(int j=R[i]; j!=i; j=R[j])
81             remove(col[j]);
82         dfs(idx+1);
83         for(int j=L[i]; j!=i; j=L[j])
84             recover(col[j]);
85     }
86     recover(c);
87 }

```

4 DataStructure

4.1 BIT

```

1 template <class T> class BIT {
2 private:
3     int size;
4     vector<T> bit;
5     vector<T> arr;
6
7 public:
8     BIT(int sz=0): size(sz), bit(sz+1),
9         arr(sz) {}
10
11     /** Sets the value at index idx to val. */
12     void set(int idx, T val) {
13         add(idx, val - arr[idx]);
14     }
15
16     /** Adds val to the element at index idx. */
17     void add(int idx, T val) {
18         arr[idx] += val;
19         for (++idx; idx<=size; idx+=(idx & -idx))
20             bit[idx] += val;
21     }
22
23     /** @return The sum of all values in [0,
24         idx]. */
25     T pre_sum(int idx) {
26         T total = 0;
27         for (++idx; idx>0; idx-= (idx & -idx))
28             total += bit[idx];
29         return total;
30     }
31 };

```

4.2 ChthollyTree

```

1 //重點：要求輸入資料隨機，否則可能被卡時間
2 struct Node {
3     long long l, r;
4     mutable long long val;
5     Node(long long l, long long r, long long
6         val) : l(l), r(r), val(val){}
7     bool operator < (const Node& other)
8         const{
9         return this->l < other.l;
10    }
11    set<Node> chthollyTree;
12    //將[l, r] 拆成 [l, pos - 1], [pos, r]
13    set<Node>::iterator split(long long pos) {
14        //找第一個左端點大於等於pos的區間
15        set<Node>::iterator it =
16            chthollyTree.lower_bound(Node(pos,
17                0, 0));
18        //運氣很好直接找到左端點是pos的區間
19        if (it != chthollyTree.end() && it->l ==
20            pos)
21            return it;
22        //到這邊代表找到的是第一個左端點大於pos的區間
23        //it - 1即可找到左端點等於pos的區間
24        //（不會是別的，因為沒有重疊的區間）
25        --it;
26        long long l = it->l, r = it->r;
27        long long val = it->val;
28        chthollyTree.erase(it);
29        chthollyTree.insert(Node(l, pos - 1,
30            val));
31        //回傳左端點是pos的區間iterator
32        return chthollyTree.insert(Node(pos, r,
33            val)).first;
34    }
35 }
36 //區間賦值

```

```

31 void assign(long long l, long long r, long
    long val) {
32     //<注意>
33     //end與begin的順序不能調換，因為end的split可能會改
34     //因為end可以在原本begin的區間中
35     set<Node>::iterator end = split(r + 1),
        begin = split(l);
36     //begin到end全部刪掉
37     chthollyTree.erase(begin, end);
38     //填回去 [l, r] 的區間
39     chthollyTree.insert(Node(l, r, val));
40 }
41 //區間加值(直接一個個區間去加)
42 void add(long long l, long long r, long long
    val) {
43     set<Node>::iterator end = split(r + 1);
44     set<Node>::iterator begin = split(l);
45     for (set<Node>::iterator it = begin; it
        != end; ++it)
46         it->val += val;
47 }
48 //查詢區間第k小 -> 直接把每個區間丟去vector排序
49 long long getKthSmallest(long long l, long
    long r, long long k) {
50     set<Node>::iterator end = split(r + 1);
51     set<Node>::iterator begin = split(l);
52     //pair -> first: val, second: 區間長度
53     vector<pair<long long, long long>> vec;
54     for (set<Node>::iterator it = begin; it
        != end; ++it) {
55         vec.push_back({it->val, it->r - it->l
            + 1});
56     }
57     sort(vec.begin(), vec.end());
58     for (const pair<long long, long long>&
        p: vec) {
59         k -= p.second;
60         if (k <= 0)
61             return p.first;
62     }
63     //不應該跑到這
64     return -1;
65 }
66 //快速幂
67 long long qpow(long long x, long long n,
    long long mod) {
68     long long res = 1;
69     x %= mod;
70     while (n)
71     {
72         if (n & 1)
73             res = res * x % mod;
74         n >>= 1;
75         x = x * x % mod;
76     }
77     return res;
78 }
79 //區間n次方和
80 long long sumOfPow(long long l, long long r,
    long long n, long long mod) {
81     long long total = 0;
82     set<Node>::iterator end = split(r + 1);
83     set<Node>::iterator begin = split(l);
84     for (set<Node>::iterator it = begin; it
        != end; ++it)
85     {
86         total = (total + qpow(it->val, n,
            mod) * (it->r - it->l + 1)) %
            mod;
87     }
88     return total;
89 }

```

4.3 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 qry(int ql, int qr, int l, int r, int i) {
22     // [ql,qr]是查詢區間，[l,r]是當前節點包含的區間
23     if (ql <= l && r <= qr)
24         return st[i];
25     int mid = l + ((r - l) >> 1);
26     if (tag[i]) {
27         //如果當前懶標有值則更新左右節點
28         st[i * 2] += tag[i] * (mid - l + 1);
29         st[i * 2 + 1] += tag[i] * (r - mid);
30         tag[i * 2] += tag[i];
31         tag[i * 2 + 1] += tag[i];
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(
42     int ql, int qr, int l, int r, int i, int c) {
43     // [ql,qr]是查詢區間，[l,r]是當前節點包含的區間
44     // c是變化量
45     if (ql <= l && r <= qr) {
46         st[i] += (r - l + 1) * c;
47         //求和，此需乘上區間長度
48         tag[i] += c;
49         return;
50     }
51     int mid = l + ((r - l) >> 1);
52     if (tag[i] && l != r) {
53         //如果當前懶標有值則更新左右節點
54         st[i * 2] += tag[i] * (mid - l + 1);
55         st[i * 2 + 1] += tag[i] * (r - mid);
56         tag[i * 2] += tag[i]; //下傳懶標至左節點
57         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
58         tag[i] = 0;
59     }
60     if (ql <= mid) update(ql, qr, l, mid, i
        * 2, c);
61     if (qr > mid) update(ql, qr, mid + 1, r,
        i * 2 + 1, c);
62     st[i] = pull(i * 2, i * 2 + 1);
63     //如果是直接改值而不是加值，query與update中的tag與st
64     //改值從+=改成=

```

4.4 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] =
            minST[xIndex][index] = val;
9         return;
10     }
11     maxST[xIndex][index] =
        max(maxST[xIndex * 2][index],
            maxST[xIndex * 2 + 1][index]);
12     minST[xIndex][index] =
        min(minST[xIndex * 2][index],
            minST[xIndex * 2 + 1][index]);
13 }
14 else {
15     int mid = (l + r) / 2;
16     if (yPos <= mid)
17         modifyY(index * 2, l, mid, val,
            yPos, xIndex, xIsLeaf);
18     else
19         modifyY(index * 2 + 1, mid + 1,
            r, val, yPos, xIndex,
            xIsLeaf);
20 }
21 maxST[xIndex][index] =
        max(maxST[xIndex][index * 2],
            maxST[xIndex][index * 2 + 1]);
22 minST[xIndex][index] =
        min(minST[xIndex][index * 2],
            minST[xIndex][index * 2 + 1]);
23 }
24 }
25 void modifyX(int index, int l, int r, int
    val, int xPos, int yPos) {
26     if (l == r) {
27         modifyY(1, 1, N, val, yPos, index,
            true);
28     }
29     else {
30         int mid = (l + r) / 2;
31         if (xPos <= mid)
32             modifyX(index * 2, l, mid, val,
                xPos, yPos);
33         else
34             modifyX(index * 2 + 1, mid + 1,
                r, val, xPos, yPos);
35         modifyY(1, 1, N, val, yPos, index,
            false);
36     }
37 }
38 void queryY(int index, int l, int r, int
    yql, int yqr, int xIndex, int& vmax,
    int& vmin) {
39     if (yql <= l && r <= yqr) {
40         vmax = max(vmax,
            maxST[xIndex][index]);
41         vmin = min(vmin,
            minST[xIndex][index]);
42     }
43     else
44     {
45         int mid = (l + r) / 2;
46         if (yql <= mid)
47             queryY(index * 2, l, mid, yql,
                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, 1, 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 }

```

4.5 權值線段樹

```

1 //權值線段樹 + 離散化 解決區間第k小問題
2 //其他網路上的解法: 2個heap, Treap, AVL tree
3 #define maxn 30005
4 int nums[maxn];
5 int getArr[maxn];
6 int id[maxn];
7 int st[maxn << 2];
8 void update(int index, int l, int r, int qx)
9 {
10     if (l == r)
11     {
12         ++st[index];
13         return;
14     }
15     int mid = (l + r) / 2;
16     if (qx <= mid)
17         update(index * 2, l, mid, qx);
18     else
19         update(index * 2 + 1, mid + 1, r, qx);
20     st[index] = st[index * 2] + st[index * 2
        + 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 }
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             {
65                 printf("%d\n", query(1, 1,
                    stSize, k));
66                 ++k;
67                 ++getCount;
68             }
69             else {
70                 update(1, 1, stSize,
                    nums[addCount + 1]);
71                 ++addCount;
72             }
73         }
74         return 0;
75 }

```

4.6 Trie

```

1 const int maxc = 26; // 單字字符數
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     }
49     return cur->cnt || prefix;
50 }
51
52 // 字典裡有 word 的前綴為 prefix
53 bool startsWith(string prefix) {
54     return search(prefix, true);
55 }
56 };

```

4.7 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)

```

4.8 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 // 和上面同理
30 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 公式

1. Circle and Line

點 $P(x_0, y_0)$ 到直線 $L: ax + by + c = 0$ 的距離

$$d(P, L) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

兩平行直線

$L_1: ax + by + c_1 = 0$ 與 $L_2: ax + by + c_2 = 0$

的距離

$$d(L_1, L_2) = \frac{|c_1 - c_2|}{\sqrt{a^2 + b^2}}$$

2. Triangle

設三角形頂點為 $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$

點 A, B, C 的對邊長分別為 a, b, c

三角形面積為 Δ

重心為 (G_x, G_y) ，內心為 (I_x, I_y) ，

外心為 (O_x, O_y) 和垂心為 (H_x, H_y)

$$\Delta = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$$G_x = \frac{x_1 + x_2 + x_3}{3}, G_y = \frac{y_1 + y_2 + y_3}{3}$$

$$I_x = \frac{ax_1 + bx_2 + cx_3}{a + b + c}, I_y = \frac{ay_1 + by_2 + cy_3}{a + b + c}$$

$$O_x = \frac{\begin{vmatrix} x_1^2 + y_1^2 & x_2^2 + y_2^2 & x_3^2 + y_3^2 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}}{4\Delta}, O_y = \frac{\begin{vmatrix} x_1^2 + y_1^2 & x_2^2 + y_2^2 & x_3^2 + y_3^2 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}}{4\Delta}$$

$$H_x = -\frac{\begin{vmatrix} x_2x_3 + y_2y_3 & x_1x_3 + y_1y_3 & x_1x_2 + y_1y_2 \\ x_2 & x_3 & x_1 \\ y_2 & y_3 & y_1 \end{vmatrix}}{2\Delta}$$

$$H_y = -\frac{\begin{vmatrix} x_1 & x_2x_3 + y_2y_3 & x_1x_3 + y_1y_3 & x_1x_2 + y_1y_2 \\ x_2 & x_3 & x_1 \\ y_2 & y_3 & y_1 \end{vmatrix}}{2\Delta}$$

任意三角形，重心、外心、垂心共線

$$G_x = \frac{2}{3}O_x + \frac{1}{3}H_x, G_y = \frac{2}{3}O_y + \frac{1}{3}H_y$$

5.2 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 using Point = Vector;
15 using Polygon = vector<Point>;
16
17 Vector operator+(Vector a, Vector b) {
18     return Vector(a.x+b.x, a.y+b.y); }
19 Vector operator-(Vector a, Vector b) {
20     return Vector(a.x-b.x, a.y-b.y); }
21 Vector operator*(Vector a, DBL b) {
22     return Vector(a.x*b, a.y*b); }
23 Vector operator/(Vector a, DBL b) {
24     return Vector(a.x/b, a.y/b); }
25
26 TP dot(Vector a, Vector b) {
27     return a.x*b.x + a.y*b.y;
28 }
29 TP cross(Vector a, Vector b) {

```

```

30 return a.x*b.y - a.y*b.x;
31 }
32 DBL Vector::length() {
33     return sqrt(dot(*this, *this));
34 }
35 DBL dis(Point a, Point b) {
36     return sqrt(dot(a-b, a-b));
37 }
38 Vector unit_normal_vector(Vector v) {
39     DBL len = v.length();
40     return Vector(-v.y/len, v.x/len);
41 }
42
43 struct Line {
44     Point p;
45     Vector v;
46     DBL ang;
47     Line(Point _p={}, Vector _v={}) {
48         p = _p;
49         v = _v;
50         ang = atan2(v.y, v.x);
51     }
52     bool operator<(const Line& l) const {
53         return ang < l.ang;
54     }
55 };
56
57 struct Segment {
58     Point s, e;
59     Segment(): s({0, 0}), e({0, 0}) {}
60     Segment(Point s, Point e): s(s), e(e) {}
61     DBL length() { return dis(s, e); }
62 };
63
64 struct Circle {
65     Point o;
66     DBL r;
67     Circle(): o({0, 0}), r(0) {}
68     Circle(Point o, DBL r=0): o(o), r(r) {}
69     Circle(Point a, Point b) { // ab 直徑
70         o = (a + b) / 2;
71         r = dis(o, a);
72     }
73     Circle(Point a, Point b, Point c) {
74         Vector u = b-a, v = c-a;
75         DBL c1=dot(u, a+b)/2, c2=dot(v, a+c)/2;
76         DBL dx=c1*v.y-c2*u.y, dy=u.x*c2-v.x*c1;
77         o = Point(dx, dy) / cross(u, v);
78         r = dis(o, a);
79     }
80     bool cover(Point p) {
81         return dis(o, p) <= r;
82     }
83 };

```

5.3 Polygon

```

1 // 判斷點 (point) 是否在凸包 (p) 內
2 bool pointInConvex(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 }

```

5.4 Intersection

```

1 // 除 intersection(Line a, Line b) 之外，
2 // 皆尚未丟 online judge
3
4 int dcmp(DBL a, DBL b=0.0) {
5     return (a > b) - (a < b);
6 }
7
8 bool hasIntersection(Point p, Segment s) {
9     return dcmp(cross(p-s.s, s.s-s.e))==0&&
10        dcmp(dot(p.x-s.s.x, p.x-s.e.x))<=0&&
11        dcmp(dot(p.y-s.s.y, p.y-s.e.y))<=0;
12 }
13
14 bool hasIntersection(Point p, Line l) {
15     return dcmp(cross(p-l.p, l.v)) == 0;
16 }
17
18 DBL dis(Line l, Point p) {
19     DBL t = cross(p, l.v) + cross(l.v, l.p);
20     return abs(t) / sqrt(dot(l.v, l.v));
21 }
22
23 Point intersection(Line a, Line b) {
24     Vector u = a.p - b.p;
25     DBL t = 1.0*cross(b.v, u)/cross(a.v, b.v);
26     return a.p + a.v*t;
27 }
28
29 // 返回 p 在 l 上的垂足(投影點)
30 Point getPedal(Line l, Point p) {
31     DBL len = dot(p-l.p, l.v) / dot(l.v, l.v);
32     return l.p + l.v * len;
33 }

```

5.5 最小圓覆蓋

```

1 vector<Point> p(3); // 在圖上的點
2 Circle MEC(vector<Point>& v, int n, int d=0){
3     Circle mec;
4     if(d == 1) mec = Circle(p[0]);
5     if(d == 2) mec = Circle(p[0], p[1]);
6     if(d == 3) return Circle(p[0], p[1], p[2]);
7     for(int i=0; i<n; i++) {
8         if(mec.cover(v[i])) continue;
9         p[d] = v[i];
10        mec = MEC(v, i, d+1);
11    }
12    return mec;
13 }

```

5.6 旋轉卡尺

```

1 // 回傳凸包內最遠兩點的距離
2 int longest_distance(Polygon& p) {
3     auto test = [&](Line l, Point a, Point b) {
4         return cross(l.v, a-l.p)<=cross(l.v, b-l.p);
5     };
6     if(p.size() <= 2) {
7         return cross(p[0]-p[1], p[0]-p[1]);
8     }
9     int mx = 0;
10    for(int i=0, j=1, n=p.size(); i<n; i++) {

```

```

11        Line l(p[i], p[(i+1)%n] - p[i]);
12        for(; test(l, p[j], p[(j+1)%n]); j=(j+1)%n);
13        mx = max({
14            mx,
15            dot(p[(i+1)%n]-p[j], p[(i+1)%n]-p[j]),
16            dot(p[i]-p[j], p[i]-p[j])
17        });
18    }
19    return mx;
20 }

```

5.7 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
- 注意計算 d2 的型態範圍限制

```

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 }

```

5.8 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 // Return: 能形成半平面交的凸包邊界點
8 Polygon halfplaneIntersect(vector<Line>& nar){
9     sort(nar.begin(), nar.end());
10    // DBL 跟 0 比較，沒浮點數不用

```

```

11 auto dblcmp=[](DBL v){return (v>0)-(v<0)};
12 // p 是否在 l 的左半平面
13 auto lft = [&](Point p, Line l) {
14     return dblcmp(cross(l.v, p-l.p)) > 0;
15 };
16
17 int ql = 0, qr = 0;
18 Line L[maxn] = {nar[0]};
19 Point P[maxn];
20
21 for(int i=1; i<nar.size(); i++) {
22     for(; ql<qr&&!lft(P[qr-1], nar[i]); qr--);
23     for(; ql<qr&&!lft(P[ql], nar[i]); ql++);
24     L[++qr] = nar[i];
25     if(dblcmp(cross(L[qr].v, L[qr-1].v))==0) {
26         if(lft(nar[i].p, L[--qr])) L[qr]=nar[i];
27     }
28     if(ql < qr)
29         P[qr-1] = intersection(L[qr-1], L[qr]);
30 }
31 for(; ql<qr && !lft(P[qr-1], L[ql]); qr--);
32 if(qr-ql <= 1) return {};
33 P[qr] = intersection(L[qr], L[ql]);
34 return Polygon(P+ql, P+qr+1);
35 }

```

6 DP

6.1 以價值為主的背包

```

1 /*w 變得太大所以一般的01背包解法變得不可能
2 觀察題目w變成10^9
3 而v_i變成10^3
4 N不變10^2
5 試著湊湊看dp狀態
6 dp[maxn][maxv]是可接受的複雜度
7 剩下的是轉移式，轉移式變成
8 dp[i][j] = w ->
9     當目前只考慮到第i個商品時，達到獲利j時最少的weight
10    = w
11 所以答案是dp[n][1 ~ maxv]找價值最大且裝的下的*/
12 #define maxn 105
13 #define maxv 100005
14 long long dp[maxn][maxv];
15 long long weight[maxn];
16 long long v[maxn];
17 int main() {
18     int n;
19     long long w;
20     scanf("%d %lld", &n, &w);
21     for (int i = 1; i <= n; ++i) {
22         scanf("%lld %lld", &weight[i], &v[i]);
23     }
24     memset(dp, 0x3f, sizeof(dp));
25     dp[0][0] = 0;
26     for (int i = 1; i <= n; ++i) {
27         for (int j = 0; j <= maxv; ++j) {
28             if (j - v[i] >= 0)
29                 dp[i][j] = dp[i-1][j-v[i]] + weight[i];
30             dp[i][j] = min(dp[i-1][j], dp[i][j]);
31         }
32     }
33     long long res = 0;
34     for (int j = maxv-1; j >= 0; --j) {
35         if (dp[n][j] <= w) {
36             res = j;
37             break;
38         }
39     }
40     printf("%lld\n", res);
41     return 0;
42 }

```

6.2 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];

```

6.3 Barcode

```

1 int N, K, M;
2 long long dp[55][55];
3 // n -> 目前剩多少units
4 // k -> 目前剩多少bars

```

```

5 // m -> 1 bar最多多少units
6 long long dfs(int n, int k) {
7     if (k == 1) {
8         return (n <= M);
9     }
10    if (dp[n][k] != -1)
11        return dp[n][k];
12    long long result = 0;
13    for (int i = 1; i < min(M+1, n); ++i)
14        { // < min(M+1, n)是因為n不能==0
15            result += dfs(n-i, k-1);
16        }
17    return dp[n][k] = result;
18 }
19 int main() {
20     while (scanf("%d %d %d", &N, &K, &M) !=
21             EOF) {
22         memset(dp, -1, sizeof(dp));
23         printf("%lld\n", dfs(N, K));
24     }
25     return 0;
26 }

```

6.4 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 }

```

6.5 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

```

6.6 RangeDP

```

1 //區間dp
2 int dp[55][55];
3 // dp[i][j] -> [i,j] 切割區間中最小的cost
4 int cuts[55];
5 int solve(int i, int j) {
6     if (dp[i][j] != -1)
7         return dp[i][j];
8     //代表沒有其他切法，只能是cuts[j] - cuts[i]

```

```

9   if (i == j - 1)
10      return dp[i][j] = 0;
11   int cost = 0x3f3f3f3f;
12   for (int m = i + 1; m < j; ++m) {
13       //枚舉區間中間切點
14       cost = min(cost, solve(i, m) +
15                 solve(m, j) + cuts[j] - cuts[i]);
16   }
17   return dp[i][j] = cost;
18 }
19 int main() {
20     int l, n;
21     while (scanf("%d", &l) != EOF && l) {
22         scanf("%d", &n);
23         for (int i = 1; i <= n; ++i)
24             scanf("%d", &cuts[i]);
25         cuts[0] = 0;
26         cuts[n + 1] = 1;
27         memset(dp, -1, sizeof(dp));
28         printf("ans = %d.\n", solve(0, n + 1));
29     }
30     return 0;
31 }

```

6.7 stringDP

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

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

Longest Palindromic Subsequence

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

6.8 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 }

```

6.9 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 }

```

6.10 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 }

```