

## Contents

1	Ubuntu	1
1.1	GDB 參數	1
1.2	GDB 指令	1
2	字串	1
2.1	KMP	1
2.2	Z Algorithm	1
2.3	最長迴文子字串	1
3	math	2
3.1	公式	2
3.2	Rational	2
3.3	質數與因數	2
3.4	Pisano Period	3
3.5	矩陣快速幂	3
3.6	歐拉函數	3
3.7	乘法逆元、組合數	3
3.8	大步小步	4
3.9	高斯消去	4
4	algorithm	5
4.1	greedy	5
4.2	三分搜	6
4.3	dinic	6
4.4	JosephusProblem	6
4.5	SCC Kosaraju	6
4.6	SCC Tarjan	7
4.7	ArticulationPoints Tarjan	7
4.8	最小樹狀圖	7
4.9	KM	8
4.10	二分圖最大匹配	8
4.11	莫隊	8
4.12	Blossom Algorithm	9
4.13	差分	9
4.14	Astar	9
4.15	Dancing Links	10
4.16	MCMF	10
4.17	LCA 倍增法	11
4.18	LCA 樹壓平 RMQ	11
4.19	LCA 樹鍊剖分	12
5	DataStructure	13
5.1	BIT	13
5.2	帶權併查集	13
5.3	線段樹 1D	13
5.4	線段樹 2D	13
5.5	權值線段樹	14
5.6	ChthollyTree	14
5.7	單調隊列	15
5.8	Trie	15
5.9	AC Trie	15
6	Geometry	16
6.1	公式	16
6.2	Template	16
6.3	最小圓覆蓋	16
6.4	Intersection	16
6.5	Polygon	16
6.6	旋轉卡尺	17
6.7	凸包	17
6.8	半平面相交	17
7	DP	18
7.1	以價值為主的背包	18
7.2	Barcode	18
7.3	RangeDP	18
7.4	抽屜	18
7.5	Deque 最大差距	18
7.6	LCS 和 LIS	19
7.7	stringDP	19
7.8	樹 DP 有幾個 path 長度為 k	19
7.9	TreeDP reroot	19
7.10	WeightedLIS	19
8	DP List	20

## 1 Ubuntu

### 1.1 GDB 參數

`[-tui]` - 在終端機顯示檔案  
`[-q]` - 在初始設定不顯示版本

### 1.2 GDB 指令

command	功能
[b]	在當前這行放中斷點
[bt]	印出堆疊
[b fn]	在函式 fn 的開頭放中斷點
[b N]	在第 N 行放中斷點
[b +N]	從目前這行屬往下 N 行都放中斷點
[c]	繼續執行程式直到下一個中斷點或錯誤
[call fn]	呼叫函式 fn
[d]	刪除所有中斷點
[d N]	刪除編號 N 的中斷點
[display x]	像 [p], 但每一次下一步都會印出 x
[f]	執行程式直到跑完當前函式
[info break]	列出所有中斷點
[l]	印出 10 行程式碼
[l N]	印出包含第 N 行的程式碼
[l fn]	印出包含函式 fn 的程式碼
[n]	像 [s], 但不進入函數
[p var]	印出 var
[q]	結束 gdb
[r]	執行程式
[return]	從當前函式 return
[r < file1]	像 [r], 且以 file1 作為輸入
[s]	執行下一行
[s N]	執行下 N 行
[set x=y]	將 x 設定為 y
[u]	網上一層堆疊
[undisplay x]	取消 display x
[watch x=3]	執行程式直到符合條件時暫停

## 2 字串

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

### 2.2 Z Algorithm

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

### 2.3 最長迴文子字串

```
1 #include<bits/stdc++.h>
2 #define T(x) ((x)%2 ? s[(x)/2] : '.')
3 using namespace std;
4
5 string s;
6 int n;
7
8 int ex(int l,int r){
9     int i=0;
10    while(l-i>=0&&r+i<n&&T(l-i)==T(r+i)) i++;
11    return i;
12 }
13
14 int main(){
15     cin>>s;
16     n=2*s.size()+1;
17     int mx=0;
18     int center=0;
19     vector<int> r(n);
20     int ans=1;
21     r[0]=1;
22     for(int i=1;i<n;i++){
23         int ii=center-(i-center);
24         int len=mx-i+1;
25         if(i>mx){
26             r[i]=ex(i,i);
27             center=i;
28             mx=i+r[i]-1;
29         }
30         else if(r[ii]==len){
31             r[i]=len+ex(i-len,i+len);
32             center=i;
33             mx=i+r[i]-1;
34         }
35         else r[i]=min(r[ii],len);
36         ans=max(ans,r[i]);
37     }
38     cout<<ans-1<<"\n";
39     return 0;
40 }
```

### 3 math

#### 3.1 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. Catlan 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

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

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

也可用高斯消去法找  $deg(p+1)$  的多項式，例：

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

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

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

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

##### 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

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

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

##### 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

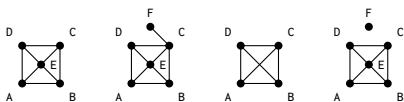
給定頂點座標均是整點（或正方形格子點）的簡單多邊形，  
 其面積  $A$  和內部格點數目  $i$ 、邊上格點數目  $b$  的關係為

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

##### 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

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

### 3.2 Rational

```

1 const char sep = '/'; // 分數的分隔符
2 bool div0; // 要記得適時歸零
3 using ll = long long;
4
5 struct Rational {
6     ll p, q;
7     Rational(ll a=0, ll b=1) {
8         p = a, q = b;
9         reduce();
10    }
11    Rational(string s) {
12        if(s.find(sep) == string::npos) {
13            p = stoll(s);
14            q = 1;
15        } else {
16            p = stoll(s.substr(0, s.find(sep)));
17            q = stoll(s.substr(s.find(sep)+1));
18        }
19        reduce();
20    }
21    void reduce() {
22        ll t = abs(__gcd(p, q));
23        if(t == 0) {
24            div0 = true;
25            return;
26        }
27        p /= t, q /= t;
28        if(q < 0) p = -p, q = -q;
29        return;
30    }
31    string toString() {
32        if(q == 0) {
33            div0 = true;
34            return "INVALID";
35        }
36        if(p%q == 0) return to_string(p/q);
37        return to_string(p) + sep + to_string(q);
38    }
39    friend istream& operator>>(
40        istream& i, Rational& r) {
41        string s;
42        i >> s;
43        r = Rational(s);
44        return i;
45    }
46    friend ostream& operator<<(
47        ostream& o, Rational r) {
48        o << r.toString();
49        return o;
50    }
51 };
52
53 Rational operator+(Rational x, Rational y) {
54     ll t = abs(__gcd(x.q, y.q));
55     if(t == 0) return Rational(0, 0);
56     return Rational(
57         y.q/t*x.p + x.q/t*y.p, x.q/t*y.q);
58 }
59 Rational operator-(Rational x, Rational y) {
60     return x + Rational(-y.p, y.q);
61 }
62 Rational operator*(Rational x, Rational y) {
63     return Rational(x.p*y.p, x.q*y.q);
64 }
65 Rational operator/(Rational x, Rational y) {
66     return x * Rational(y.q, y.p);
67 }

```

### 3.3 質數與因數

```

1 歐拉篩O(n)
2 #define MAXN 47000 //sqrt(2^31)=46,340...
3 bool isPrime[MAXN];
4 int p[MAXN];
5 int pSize=0;
6 void getPrimes(){
7     memset(isPrime, true, sizeof(isPrime));
8     isPrime[0]=isPrime[1]=false;
9     for(int i=2; i<MAXN; i++){
10        if(isPrime[i]) p[pSize++]=i;
11        for(int j=0; j<pSize&&i*p[j]<=MAXN; ++j){
12            isPrime[i*p[j]]=false;
13            if(i%p[j]==0) break;
14        }
15    }
16 }
17
18 最大公因數 O(log(min(a,b)))
19 int GCD(int a, int b){
20     if(b == 0) return a;
21     return GCD(b, a%b);
22 }
23
24 質因數分解
25 void primeFactorization(int n){
26     for(int i=0; i<p.size(); ++i) {
27         if(p[i]*p[i] > n) break;
28         if(n % p[i]) continue;
29         cout << p[i] << ' ';
30         while(n%p[i] == 0) n /= p[i];
31     }
32     if(n != 1) cout << n << ' ';
33     cout << '\n';
34 }
35
36 擴展歐幾里得算法 ax + by = GCD(a, b)
37 int ext_euc(int a, int b, int &x, int &y) {
38     if(b == 0){
39         x = 1, y = 0;
40         return a;
41     }
42     int d = ext_euc(b, a%b, y, x);
43     y -= a/b*x;
44     return d;
45 }
46 int main(){
47     int a, b, x, y;
48     cin >> a >> b;
49     ext_euc(a, b, x, y);
50     cout << x << ' ' << y << endl;
51     return 0;
52 }
53
54 歌德巴赫猜想
55 解：把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
56 #define N 2000000
57 int ox[N], p[N], pr;
58 void PrimeTable(){
59     ox[0] = ox[1] = 1;
60     pr = 0;
61     for(int i=2; i<N; i++){
62         if(!ox[i]) p[pr++] = i;
63         for(int j=0; i*p[j]<N&&j<pr; j++){
64             ox[i*p[j]] = 1;
65         }
66     }
67 }
68
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";
101 }

```

### 3.4 Pisano Period

```

1 #include <cstdio>
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 mod)
29 {
30     if (b == 0) return a;
31     long long res = 1;
32     while (b) {
33         if (b & 1)
34             res = ((a % mod) * (res % mod)) % mod;
35         a = ((a % mod) * (a % mod)) % mod;
36         b >>= 1;
37     }
38     return res;
39 }
40
41 int main()
42 {
43     int t;
44     unsigned long long a, b;
45     int n;
46

```

```

47 //注意：這裡沒算mod 1的循環長度，
48 //因為mod 1都等於0，沒有週期
49 for (int i = 2; i < maxn; ++i)
50 {
51     fib[i].emplace_back(0);
52     fib[i].emplace_back(1);
53     for (int j = 2; j < maxn * maxn; ++j)
54     {
55         fib[i].emplace_back(
56             (fib[i][j-1]%i+fib[i][j-2]%i)%i
57         );
58         if (fib[i][j-1]==0&&fib[i][j]==1)
59         {
60             period[i] = j - 1;
61             break;
62         }
63     }
64 }
65
66 scanf("%d", &t);
67
68 while (t--)
69 {
70     scanf("%llu %llu %d", &a, &b, &n);
71     if (a == 0)
72         puts("0");
73     else if (n == 1) //當mod 1時任何數都是0，
74         puts("0");
75     //所以直接輸出0，避免我們沒算
76     //fib[1][i]的問題(Runtime
77     error)
78     printf("%d\n",
79         fib[n][qpow(a % period[n], b,
80             period[n])]);
81 }
82 return 0;
83 }

```

### 3.5 矩陣快速幂

```

1 using ll = long long;
2 using mat = vector<vector<ll>>>;
3 const int mod = 1e9 + 7;
4
5 mat operator*(mat A, mat B) {
6     mat res(A.size(), 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 矩陣
21 mat mpow(mat& M, int n) {
22     if(n <= 1) return n ? M : I;
23     mat v = mpow(M, n>>1);
24     return (n & 1) ? v*v*M : v*v;
25 }
26
27 // 迴圈版本
28 mat mpow(mat M, int n) {
29     mat res(M.size(), vector<ll>(M[0].size()));
30     for(int i=0; i<res.size(); i++)
31         res[i][i] = 1;
32     for(; n>>=1) {
33         if(n & 1) res = res * M;
34         M = M * M;
35     }
36     return res;
37 }

```

### 3.6 歐拉函數

```

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

```

### 3.7 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>>=1) {
22             if(b & 1) res = res * a % mod;
23             a = a * a % mod;
24         }
25         return res;
26     };
27
28     for(int x=2; x<maxn; x++) {
29         fact[x] = (ll)x * fact[x-1] % mod;
30         invFact[x] = qpow(fact[x], mod-2);
31     }
32 }
33
34 // C(a, b) % mod
35 int comb(int a, int b) {
36     if(a < b) return 0;
37     ll x = fact[a];
38     ll y = (ll)invFact[b] * invFact[a-b] % mod;
39     return x * y % mod;
40 }

```

### 3.8 大步小步

### 3.9 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 }
```

```
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
5    $B^0, B^P, B^{2P}, \dots$ ，
6 也就是說如果有解則  $L < N$ ，枚舉  $0, 1, 2, L-1$ 
7 能得到結果，但會超時。
8 將 L 拆成  $mx+y$ ，只要分別枚舉 x,y 就能得到答案，
9 設  $m=\sqrt{P}$  能保證最多枚舉  $2\sqrt{P}$  次。
10  $B^{mx+y} \equiv N \pmod P$ 
11  $B^{mx} B^y \equiv N \pmod P$ 
12 先求出  $B^0, B^1, B^2, \dots, B^{m-1}$ ，
13 再枚舉  $N(B^{(-m)}), N(B^{(-m)})^2, \dots$  查看是否有對應的  $B^y$ 。
14 這種算法稱為大步小步演算法，
15 大步指的是枚舉 x (一次跨 m 步)，
16 小步指的是枚舉 y (一次跨 1 步)。
17 複雜度分析
18 利用 map/unorder_map 存放
19    $B^0, B^1, B^2, \dots, B^{m-1}$ ，
20 枚舉 x 查詢 map/unorder_map 是否有對應的  $B^y$ ，
21 存放和查詢最多  $2\sqrt{P}$  次，時間複雜度為
22    $O(\sqrt{P} \log \sqrt{P}) / O(\sqrt{P})$ 。
23
24 using LL = long long;
25 LL B, N, P;
26 LL fpow(LL a, LL b, LL c){
27     LL res=1;
28     for(;b>=1;){
29         if(b&1)
30             res=(res*a)%c;
31         a=(a*a)%c;
32         b>>=1;
33     }
34     return res;
35 }
36 LL BSGS(LL a, LL b, LL p){
37     a%=p, b%=p;
38     if(a==0)
39         return b==0?1:-1;
40     if(b==1)
41         return 0;
42     map<LL, LL> tb;
43     LL sq=ceil(sqrt(p-1));
44     LL inv=fpow(a, p-sq-1, p);
45     tb[1]=sq;
46     for(LL i=1, tmp=1; i<sq; ++i){
47         tmp=(tmp*a)%p;
48         if(!tb.count(tmp))
49             tb[tmp]=i;
50     }
51     for(LL i=0; i<sq; ++i){
52         if(tb.count(b)){
53             LL res=tb[b];
54             return i*sq+(res==sq?0:res);
55         }
56         b=(b*inv)%p;
57     }
58     return -1;
59 }
60 int main(){
61     IOS; //輸入優化
62     while(cin>>P>>B>>N){
63         LL ans=BSGS(B,N,P);
64         if(ans!=-1)
65             cout<<"no solution\n";
66         else
67             cout<<ans<<"\n";
68     }
69 }
```

## 4 algorithm

### 4.1 greedy

```

1 刪數字問題
2 //problem
3 給定一個數字  $N(\leq 10^{100})$ ，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  $\max Li$  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 最少延遲數量問題
135 //problem
136 給定  $N$  個工作，每個工作的需要處理時長為  $Ti$ ，
137 期限是  $Di$ ，求一種工作排序使得逾期工作數量最小。
138 //solution
139 期限越早到期的工作越先做。
140 將工作依照到期時間從早到晚排序，
141 依序放入工作列表中，如果發現有工作預期，
142 就從目前選擇的工作中，移除耗時最長的工作。
143 上述方法為 Moore-Hodgson's Algorithm。
144 //problem
145 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？
146 //solution
147 和最少延遲數量問題是相同的問題，只要將題敘做轉換。
148 工作處理時長  $\rightarrow$  烏龜重量
149 工作期限  $\rightarrow$  烏龜可承受重量
150 多少工作不延期  $\rightarrow$  可以疊幾隻烏龜
151 //code

```

```

151 struct Work{
152     int t, d;
153     bool operator<(const Work &rhs)const{
154         return d<rhs.d;
155     }
156 };
157 int main(){
158     int n=0;
159     Work a[10000];
160     priority_queue<int> pq;
161     while(cin>>a[n].t>>a[n].d)
162         ++n;
163     sort(a,a+n);
164     int sumT=0,ans=n;
165     for(int i=0;i<n;++i){
166         pq.push(a[i].t);
167         sumT+=a[i].t;
168         if(a[i].d<sumT){
169             int x=pq.top();
170             pq.pop();
171             sumT-=x;
172             --ans;
173         }
174     }
175     cout<<ans<<'\n';
176 }
177 }
178 任務調度問題
179 //problem
180 給定  $N$  項工作，每項工作的需要處理時長為  $Ti$ ，
181 期限是  $Di$ ，如果第  $i$  項工作延遲需要受到  $pi$ 
182 單位懲罰，
183 請問最少會受到多少單位懲罰。
184 //solution
185 依照懲罰由大到小排序，
186 每項工作依序嘗試可不可以放在
187  $Di-Ti+1, Di-Ti, \dots, 1, 0$ ，
188 如果有空間就放進去，否則延後執行。
189 //problem
190 給定  $N$  項工作，每項工作的需要處理時長為  $Ti$ ，
191 期限是  $Di$ ，如果第  $i$  項工作在期限內完成會獲得  $ai$ 
192 單位獎勵，
193 請問最多會獲得多少單位獎勵。
194 //solution
195 和上題相似，這題變成依照獎勵由大到小排序。
196 //code
197 struct Work{
198     int d,p;
199     bool operator<(const Work &rhs)const{
200         return p>rhs.p;
201     }
202 };
203 int main(){
204     int n;
205     Work a[100005];
206     bitset<100005> ok;
207     while(cin>>n){
208         ok.reset();
209         for(int i=0;i<n;++i)
210             cin>>a[i].d>>a[i].p;
211         sort(a,a+n);
212         int ans=0;
213         for(int i=0;i<n;++i){
214             int j=a[i].d;
215             while(j--){
216                 if(!ok[j]){
217                     ans+=a[i].p;
218                     ok[j]=true;
219                     break;
220                 }
221             }
222         }
223     }
224     cout<<ans<<'\n';
225 }

```



## 4.2 三分搜

```

1  題意
2  給定兩射線方向和速度，問兩射線最近距離。
3  題解
4  假設  $F(t)$  為兩射線在時間  $t$  的距離， $F(t)$ 
    為二次函數，
5  可用三分查找二次函數最小值。
6  struct Point{
7      double x, y, z;
8      Point() {}
9      Point(double _x, double _y, double _z):
10         x(_x), y(_y), z(_z){}
11     friend istream& operator>>(istream& is,
12         Point& p) {
13         is >> p.x >> p.y >> p.z;
14         return is;
15     }
16     Point operator+(const Point &rhs) const{
17         return Point(x+rhs.x, y+rhs.y, z+rhs.z);
18     }
19     Point operator-(const Point &rhs) const{
20         return Point(x-rhs.x, y-rhs.y, z-rhs.z);
21     }
22     Point operator*(const double &d) const{
23         return Point(x*d, y*d, z*d);
24     }
25     Point operator/(const double &d) const{
26         return Point(x/d, y/d, z/d);
27     }
28     double dist(const Point &rhs) const{
29         double res = 0;
30         res += (x-rhs.x)*(x-rhs.x);
31         res += (y-rhs.y)*(y-rhs.y);
32         res += (z-rhs.z)*(z-rhs.z);
33         return res;
34     };
35 int main(){
36     IOS; //輸入優化
37     int T;
38     cin>>T;
39     for(int ti=1; ti<=T; ++ti){
40         double time;
41         Point x1, y1, d1, x2, y2, d2;
42         cin>>time>>x1>>y1>>x2>>y2;
43         d1=(y1-x1)/time;
44         d2=(y2-x2)/time;
45         double L=0, R=1e8, m1, m2, f1, f2;
46         double ans = x1.dist(x2);
47         while(abs(L-R)>1e-10){
48             m1=(L+R)/2;
49             m2=(m1+R)/2;
50             f1=((d1*m1)+x1).dist((d2*m1)+x2);
51             f2=((d1*m2)+x1).dist((d2*m2)+x2);
52             ans = min(ans, min(f1, f2));
53             if(f1<f2) R=m2;
54             else L=m1;
55         }
56         cout<<"Case "<<ti<<": ";
57         cout << fixed << setprecision(4) <<
            sqrt(ans) << '\n';
58     }
59 }

```

## 4.3 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 int dinic() { //  $O((V^2)E)$ 
59     int result = 0;
60     while(bfs()) {
61         memset(dfs_idx, 0, sizeof(dfs_idx));
62         result += dfs(S, inf);
63     }
64     return result;
65 }

```

## 4.4 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
5  //  $O(n)$ 
6  int getWinner(int n, int k) {
7      int winner = 0;
8      for (int i = 1; i <= n; ++i)
9          winner = (winner + k) % i;
10     return winner;
11 }
12
13 int main() {
14     int n;
15     while (scanf("%d", &n) != EOF && n){
16         --n;
17         for (int k = 1; k <= n; ++k){
18             if (getWinner(n, k) == 11){
19                 printf("%d\n", k);
20                 break;
21             }
22         }
23     }
24     return 0;
25 }
26
27 //  $O(k \log(n))$ 
28 int josephus(int n, int k) {
29     if (n == 1) return 0;
30     if (k == 1) return n - 1;
31     if (k > n) return (josephus(n-1, k)+k)%n;
32     int res = josephus(n - n / k, k);
33     res -= n % k;
34     if (res < 0)
35         res += n; // mod n
36     else
37         res += res / (k - 1); // 还原位置
38     return res;
39 }

```

## 4.5 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
11 void dfs2(int u) {
12     group[u] = sccCnt;
13     for (int v : g2[u])
14         if (!group[v]) dfs2(v);
15 }
16
17 void kosaraju() {
18     sccCnt = 0;
19     for (int i = 1; i <= n; ++i)
20         if (!vis[i]) dfs1(i);
21     for (int i = n; i >= 1; --i)
22         if (!group[s[i]]) {
23             ++sccCnt;
24             dfs2(s[i]);
25         }
26 }

```

## 4.6 SCC Tarjan

## 4.7 ArticulationPoints Tarjan

## 4.8 最小樹狀圖

```

1 //單純考SCC，每個SCC中找成本最小的蓋，如果有多個一樣小
2 //的要數出來，因為題目要方法數
3 //注意以下程式有縮點，但沒存起來，
4 //存法就是開一個array -> ID[u] = SCCID
5 #define maxn 100005
6 #define MOD 1000000007
7 long long cost[maxn];
8 vector<vector<int>>> G;
9 int SCC = 0;
10 stack<int> sk;
11 int dfn[maxn];
12 int low[maxn];
13 bool inStack[maxn];
14 int dfsTime = 1;
15 long long totalCost = 0;
16 long long ways = 1;
17 void dfs(int u) {
18     dfn[u] = low[u] = dfsTime;
19     ++dfsTime;
20     sk.push(u);
21     inStack[u] = true;
22     for (int v: G[u]) {
23         if (dfn[v] == 0) {
24             dfs(v);
25             low[u] = min(low[u], low[v]);
26         }
27         else if (inStack[v]) {
28             //屬於同個SCC且是我的back edge
29             low[u] = min(low[u], dfn[v]);
30         }
31     }
32     //如果是SCC
33     if (dfn[u] == low[u]) {
34         long long minCost = 0x3f3f3f3f;
35         int currWays = 0;
36         ++SCC;
37         while (1) {
38             int v = sk.top();
39             inStack[v] = 0;
40             sk.pop();
41             if (minCost > cost[v]) {
42                 minCost = cost[v];
43                 currWays = 1;
44             }
45             else if (minCost == cost[v]) {
46                 ++currWays;
47             }
48             if (v == u)
49                 break;
50         }
51         totalCost += minCost;
52         ways = (ways * currWays) % MOD;
53     }
54 }
55 int main() {
56     int n;
57     scanf("%d", &n);
58     for (int i = 1; i <= n; ++i)
59         scanf("%lld", &cost[i]);
60     G.assign(n + 5, vector<int>());
61     int m;
62     scanf("%d", &m);
63     int u, v;
64     for (int i = 0; i < m; ++i) {
65         scanf("%d %d", &u, &v);
66         G[u].emplace_back(v);
67     }
68     for (int i = 1; i <= n; ++i) {
69         if (dfn[i] == 0)
70             dfs(i);
71     }
72     printf("%lld %lld\n", totalCost, ways %
73         MOD);
74     return 0;
75 }

```

```

vector<vector<int>>> G;
2 int N, timer;
3 bool visited[105];
4 int dfn[105]; // 第一次visit的時間
5 int low[105];
6 //最小能回到的父節點
7 //(不能是自己的parent)的visTime
8 int res;
9 //求割點數量
10 void tarjan(int u, int parent) {
11     int child = 0;
12     bool isCut = false;
13     visited[u] = true;
14     dfn[u] = low[u] = ++timer;
15     for (int v: G[u]) {
16         if (!visited[v]) {
17             ++child;
18             tarjan(v, u);
19             low[u] = min(low[u], low[v]);
20             if (parent != -1 && low[v] >=
21                 dfn[u])
22                 isCut = true;
23         }
24         else if (v != parent)
25             low[u] = min(low[u], dfn[v]);
26     }
27     //If u is root of DFS
28     //tree->有兩個以上的children
29     if (parent == -1 && child >= 2)
30         isCut = true;
31     if (isCut) ++res;
32 }
33 int main() {
34     char input[105];
35     char* token;
36     while (scanf("%d", &N) != EOF && N) {
37         G.assign(105, vector<int>());
38         memset(visited, false,
39             sizeof(visited));
40         memset(low, 0, sizeof(low));
41         memset(dfn, 0, sizeof(dfn));
42         timer = 0;
43         res = 0;
44         getchar(); // for \n
45         while (fgets(input, 105, stdin)) {
46             if (input[0] == '\0')
47                 break;
48             int size = strlen(input);
49             input[size - 1] = '\0';
50             --size;
51             token = strtok(input, " ");
52             int u = atoi(token);
53             int v;
54             while (token = strtok(NULL, " "))
55                 v = atoi(token);
56             G[u].emplace_back(v);
57             G[v].emplace_back(u);
58         }
59         tarjan(1, -1);
60         printf("%d\n", res);
61     }
62     return 0;
63 }

```

```

1 const int maxn = 60 + 10;
2 const int inf = 0x3f3f3f3f;
3 struct Edge {
4     int s, t, cap, cost;
5 }; // cap 為頻寬 (optional)
6 int n, m, c;
7 int inEdge[maxn], idx[maxn], pre[maxn],
8     vis[maxn];
9 // 對於每個點，選擇對它入度最小的那條邊
10 // 找環，如果沒有則 return;
11 // 進行縮環並更新其他點到環的距離。
12 int dirMST(vector<Edge> edges, int low) {
13     int result = 0, root = 0, N = n;
14     while(true) {
15         memset(inEdge, 0x3f, sizeof(inEdge));
16         // 找所有點的 in edge 放進 inEdge
17         // optional: low 為最小 cap 限制
18         for(const Edge& e : edges) {
19             if(e.cap < low) continue;
20             if(e.s!=e.t &&
21                 e.cost<inEdge[e.t]) {
22                 inEdge[e.t] = e.cost;
23                 pre[e.t] = e.s;
24             }
25         }
26         for(int i=0; i<N; i++) {
27             if(i!=root && inEdge[i]==inf)
28                 return -1; //除了root 還有點沒有 in
29                 edge
30         }
31         int seq = inEdge[root] = 0;
32         memset(idx, -1, sizeof(idx));
33         memset(vis, -1, sizeof(vis));
34         // 找所有的 cycle，一起編號為 seq
35         for(int i=0; i<N; i++) {
36             result += inEdge[i];
37             int cur = i;
38             while(vis[cur] != i &&
39                 idx[cur] == -1) {
40                 if(cur == root) break;
41                 vis[cur] = i;
42                 cur = pre[cur];
43             }
44             if(cur != root && idx[cur] == -1) {
45                 for(int j=pre[cur]; j!=cur;
46                     j=pre[j])
47                     idx[j] = seq;
48                 idx[cur] = seq++;
49             }
50         }
51         if(seq == 0) return result; // 沒有
52         cycle
53         for(int i=0; i<N; i++)
54             // 沒有被縮點的點
55             if(idx[i] == -1) idx[i] = seq++;
56         // 縮點並重新編號
57         for(Edge& e : edges) {
58             if(idx[e.s] != idx[e.t])
59                 e.cost -= inEdge[e.t];
60             e.s = idx[e.s];
61             e.t = idx[e.t];
62         }
63         N = seq;
64         root = idx[root];
65     }
66 }

```

## 4.9 KM

```

1 #define maxn 505
2 int W[maxn][maxn];
3 int Lx[maxn], Ly[maxn];
4 bool S[maxn], T[maxn];
5 //L[i] = j -> S_i配給T_j, -1 for 還沒匹配
6 int L[maxn];
7 int n;
8 bool match(int i) {
9     S[i] = true;
10    for (int j = 0; j < n; ++j) {
11        // KM重點
12        // Lx + Ly >= selected_edge(x, y)
13        // 要想辦法降低Lx + Ly
14        // 所以選Lx + Ly == selected_edge(x, y)
15        if (Lx[i] + Ly[j] == W[i][j] &&
16            !T[j]) {
17            T[j] = true;
18            if ((L[j] == -1) || match(L[j])) {
19                L[j] = i;
20                return true;
21            }
22        }
23    }
24    return false;
25 }
26 //修改二分圖上的交錯路徑上點的權重
27 //此舉是在通過調整vertex labeling看看
28 //能不能產生出新的增廣路
29 //(KM的增廣路要求Lx[i] + Ly[j] == W[i][j])
30 //在這裡優先從最小的diff調看，才能保證最大權重匹配
31 void update() {
32     int diff = 0x3f3f3f3f;
33     for (int i = 0; i < n; ++i) {
34         if (S[i]) {
35             for (int j = 0; j < n; ++j) {
36                 if (!T[j])
37                     diff = min(diff, Lx[i] +
38                               Ly[j] - W[i][j]);
39             }
40         }
41         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)) break;
59             else update(); //去調整vertex
60                             //labeling以增加增廣路徑
61         }
62     }
63 }
64 int main() {
65     while (scanf("%d", &n) != EOF) {
66         for (int i = 0; i < n; ++i)
67             for (int j = 0; j < n; ++j)
68                 scanf("%d", &W[i][j]);
69         KM();
70         int res = 0;
71         for (int i = 0; i < n; ++i) {
72             if (i != 0)
73                 printf(" %d", Lx[i]);
74             else
75                 printf("%d", Lx[i]);
76             res += Lx[i];
77         }
78     }
79 }

```

```

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

```

## 4.10 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 }

```

## 4.11 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] ^
3    prefix[x - 1]即可在O(1)回答
4    同時維護cnt[i]代表[x, y]XOR和 == i的個數
5    如此我們知道[l, r]可以快速知道[l - 1, r], [l
6    + 1, r], [l, r - 1], [l, r + 1]的答案
7    就符合Mo's algorithm的思維O(N * sqrt(n))
8    每次轉移為O(1)，具體轉移方法在下面*/
9 #define maxn 100005
10 //在此prefix[i]是[l, i]的XOR和
11 int prefix[maxn];
12 //log_2(1000000) =
13     19.931568569324174087221916576937...
14 //所以開到1 << 20
15 //cnt[i]代表的是有符合nums[x, y] such that
16     nums[x] ^ nums[x + 1] ^ .. ^ nums[y] ==
17     i
18 //的個數
19 long long cnt[1 << 20];
20 //塊大小 -> sqrt(n)
21 int sqrtQ;
22 struct Query {
23     int l, r, id;
24     bool operator < (const Query& other)
25     const {
26         if (this->l / sqrtQ != other.l /
27             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,
50             &queries[i].r);
51         //減1是因為prefix[i]是[1,
52             i]的前綴XOR和，所以題目問[l,
53             r]我們要回答[l - 1, r]的答案
54         --queries[i].l;
55         queries[i].id = i;
56     }
57     sort(queries + 1, queries + m + 1);
58     int l = 1, r = 0;
59     for (int i = 1; i <= m; ++i) {
60         while (l < queries[i].l) {
61             sub(l);
62             ++l;
63         }
64         while (l > queries[i].l) {
65             --l;
66             add(l);
67         }
68         while (r < queries[i].r) {
69             ++r;
70             add(r);
71         }
72         while (r > queries[i].r) {
73             sub(r);
74             --r;
75         }
76         ans[queries[i].id] = res;
77     }
78     for (int i = 1; i <= m; ++i) {
79         printf("%lld\n", ans[i]);
80     }
81     return 0;
82 }

```



## 4.12 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 }

```

## 4.13 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 }

```

## 4.14 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 { int u, v, w; };
9 struct Item_pqH {
10     int u, w;
11     bool operator < (const Item_pqH& other) const {
12         return this->w > other.w;
13     }
14 };
15 struct Item_astar {
16     int u, g, f;
17     bool operator < (const Item_astar& other) const {
18         return this->f > other.f;
19     }
20 };
21 vector<vector<Edge>> G;
22 //反向圖，用於建h(u)
23 vector<vector<Edge>> invertG;
24 int h[maxn];
25 bool visited[maxn];
26 int cnt[maxn];
27 //用反向圖去求出一點到終點的最短距離，並以此當作h(u)
28 void dijkstra(int s, int t) {
29     memset(visited, 0, sizeof(visited));
30     priority_queue<Item_pqH> pq;
31     pq.push({s, 0});
32     h[s] = 0;
33     while (!pq.empty()) {
34         Item_pqH curr = pq.top();
35         pq.pop();
36         visited[curr.u] = true;
37         for (Edge& edge: invertG[curr.u]) {
38             if (!visited[edge.v]) {
39                 if (h[edge.v] > h[curr.u] + edge.w) {
40                     h[edge.v] = h[curr.u] + edge.w;
41                     pq.push({edge.v, h[edge.v]});
42                 }
43             }
44         }
45     }
46 }
47 int Astar(int s, int t, int k) {
48     memset(cnt, 0, sizeof(cnt));
49     priority_queue<Item_astar> pq;
50     pq.push({s, 0, h[s]});
51     while (!pq.empty()) {
52         Item_astar curr = pq.top();
53         pq.pop();
54         ++cnt[curr.u];
55         //終點出現k次，此時即可得k短路
56         if (cnt[t] == k)
57             return curr.g;
58         for (Edge& edge: G[curr.u]) {
59             if (cnt[edge.v] < k) {
60                 pq.push({edge.v, curr.g + edge.w, curr.g + edge.w + h[edge.v]});
61             }
62         }
63     }
64     return -1;
65 }
66 int main() {
67     int n, m;
68     while (scanf("%d %d", &n, &m) && (n != 0 && m != 0)) {

```

## 4.15 Dancing Links

```

69 G.assign(n + 5, vector<Edge>());
70 invertG.assign(n + 5, vector<Edge>());
71 int s, t, k;
72 scanf("%d %d %d", &s, &t, &k);
73 int u, v, w;
74 for (int i = 0; i < m; ++i) {
75     scanf("%d %d %d", &u, &v, &w);
76     G[u].emplace_back(Edge{u, v, w});
77     invertG[v].emplace_back(Edge{v,
78         u, w});
79 }
80 memset(h, 0x3f, sizeof(h));
81 dijkstra(t, s);
82 printf("%d\n", Astar(s, t, k));
83 }
84 return 0;
}

4.15 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;
13        resSize = -1;
14        memset(rowHead, 0, sizeof(rowHead));
15        memset(colSize, 0, sizeof(colSize));
16    }
17    void insert(int r, int c) {
18        row[++seq] = r, col[seq] = c,
19        ++colSize[c];
20        U[seq] = c, D[seq] = D[c], U[D[c]] = seq,
21        D[c] = seq;
22        if (rowHead[r]) {
23            L[seq] = rowHead[r],
24            R[seq] = R[rowHead[r]];
25            L[R[rowHead[r]]] = seq,
26            R[rowHead[r]] = seq;
27        } else {
28            rowHead[r] = L[seq] = R[seq] =
29            seq;
30        }
31    }
32    void remove(int c) {
33        L[R[c]] = L[c], R[L[c]] = R[c];
34        for (int i = D[c]; i != c; i = D[i]) {
35            for (int j = R[i]; j != i; j = R[j]) {
36                U[D[j]] = U[j];
37                D[U[j]] = D[j];
38                --colSize[col[j]];
39            }
40        }
41    }
42    void recover(int c) {
43        for (int i = U[c]; i != c; i = U[i]) {
44            for (int j = L[i]; j != i; j = L[j]) {
45                U[D[j]] = D[U[j]] = j;
46                ++colSize[col[j]];
47            }
48        }
49        L[R[c]] = R[L[c]] = c;
50    }
51    bool dfs(int idx = 0) { // 判斷其中一解版
52        if (R[0] == 0) {
53            resSize = idx;
54            return true;
55        }
56        int c = R[0];
57        for (int i = R[0]; i; i = R[i]) {
58            if (colSize[i] < colSize[c]) c = i;
59        }
60        remove(c);
61        for (int i = D[c]; i != c; i = D[i]) {
62            for (int j = R[i]; j != i; j = R[j]) {
63                U[D[j]] = U[j];
64                D[U[j]] = D[j];
65                --colSize[col[j]];
66            }
67        }
68        dfs(idx + 1);
69        recover(c);
70    }
71    void dfs(int idx = 0) { // 判斷最小 dfs
72        // depth 版
73        if (R[0] == 0) {
74            resSize = min(resSize, idx); //
75            // 注意init值
76            return;
77        }
78        int c = R[0];
79        for (int i = R[0]; i; i = R[i]) {
80            if (colSize[i] < colSize[c]) c = i;
81        }
82        remove(c);
83        for (int i = D[c]; i != c; i = D[i]) {
84            for (int j = R[i]; j != i; j = R[j]) {
85                U[D[j]] = U[j];
86                D[U[j]] = D[j];
87                --colSize[col[j]];
88            }
89        }
90        dfs(idx + 1);
91        recover(c);
92    }
93    void dfs(int idx = 0) { // 判斷最小 dfs
94        // depth 版
95        if (R[0] == 0) {
96            resSize = min(resSize, idx); //
97            // 注意init值
98            return;
99        }
100        int c = R[0];
101        for (int i = R[0]; i; i = R[i]) {
102            if (colSize[i] < colSize[c]) c = i;
103        }
104        remove(c);
105        for (int i = D[c]; i != c; i = D[i]) {
106            for (int j = R[i]; j != i; j = R[j]) {
107                U[D[j]] = U[j];
108                D[U[j]] = D[j];
109                --colSize[col[j]];
110            }
111        }
112        dfs(idx + 1);
113        recover(c);
114    }
115    void dfs(int idx = 0) { // 判斷最小 dfs
116        // depth 版
117        if (R[0] == 0) {
118            resSize = min(resSize, idx); //
119            // 注意init值
120            return;
121        }
122        int c = R[0];
123        for (int i = R[0]; i; i = R[i]) {
124            if (colSize[i] < colSize[c]) c = i;
125        }
126        remove(c);
127        for (int i = D[c]; i != c; i = D[i]) {
128            for (int j = R[i]; j != i; j = R[j]) {
129                U[D[j]] = U[j];
130                D[U[j]] = D[j];
131                --colSize[col[j]];
132            }
133        }
134        dfs(idx + 1);
135        recover(c);
136    }
137    void dfs(int idx = 0) { // 判斷最小 dfs
138        // depth 版
139        if (R[0] == 0) {
140            resSize = min(resSize, idx); //
141            // 注意init值
142            return;
143        }
144        int c = R[0];
145        for (int i = R[0]; i; i = R[i]) {
146            if (colSize[i] < colSize[c]) c = i;
147        }
148        remove(c);
149        for (int i = D[c]; i != c; i = D[i]) {
150            for (int j = R[i]; j != i; j = R[j]) {
151                U[D[j]] = U[j];
152                D[U[j]] = D[j];
153                --colSize[col[j]];
154            }
155        }
156        dfs(idx + 1);
157        recover(c);
158    }
159    void dfs(int idx = 0) { // 判斷最小 dfs
160        // depth 版
161        if (R[0] == 0) {
162            resSize = min(resSize, idx); //
163            // 注意init值
164            return;
165        }
166        int c = R[0];
167        for (int i = R[0]; i; i = R[i]) {
168            if (colSize[i] < colSize[c]) c = i;
169        }
170        remove(c);
171        for (int i = D[c]; i != c; i = D[i]) {
172            for (int j = R[i]; j != i; j = R[j]) {
173                U[D[j]] = U[j];
174                D[U[j]] = D[j];
175                --colSize[col[j]];
176            }
177        }
178        dfs(idx + 1);
179        recover(c);
180    }
181    void dfs(int idx = 0) { // 判斷最小 dfs
182        // depth 版
183        if (R[0] == 0) {
184            resSize = min(resSize, idx); //
185            // 注意init值
186            return;
187        }
188        int c = R[0];
189        for (int i = R[0]; i; i = R[i]) {
190            if (colSize[i] < colSize[c]) c = i;
191        }
192        remove(c);
193        for (int i = D[c]; i != c; i = D[i]) {
194            for (int j = R[i]; j != i; j = R[j]) {
195                U[D[j]] = U[j];
196                D[U[j]] = D[j];
197                --colSize[col[j]];
198            }
199        }
200        dfs(idx + 1);
201        recover(c);
202    }
203    void dfs(int idx = 0) { // 判斷最小 dfs
204        // depth 版
205        if (R[0] == 0) {
206            resSize = min(resSize, idx); //
207            // 注意init值
208            return;
209        }
210        int c = R[0];
211        for (int i = R[0]; i; i = R[i]) {
212            if (colSize[i] < colSize[c]) c = i;
213        }
214        remove(c);
215        for (int i = D[c]; i != c; i = D[i]) {
216            for (int j = R[i]; j != i; j = R[j]) {
217                U[D[j]] = U[j];
218                D[U[j]] = D[j];
219                --colSize[col[j]];
220            }
221        }
222        dfs(idx + 1);
223        recover(c);
224    }
225    void dfs(int idx = 0) { // 判斷最小 dfs
226        // depth 版
227        if (R[0] == 0) {
228            resSize = min(resSize, idx); //
229            // 注意init值
230            return;
231        }
232        int c = R[0];
233        for (int i = R[0]; i; i = R[i]) {
234            if (colSize[i] < colSize[c]) c = i;
235        }
236        remove(c);
237        for (int i = D[c]; i != c; i = D[i]) {
238            for (int j = R[i]; j != i; j = R[j]) {
239                U[D[j]] = U[j];
240                D[U[j]] = D[j];
241                --colSize[col[j]];
242            }
243        }
244        dfs(idx + 1);
245        recover(c);
246    }
247    void dfs(int idx = 0) { // 判斷最小 dfs
248        // depth 版
249        if (R[0] == 0) {
250            resSize = min(resSize, idx); //
251            // 注意init值
252            return;
253        }
254        int c = R[0];
255        for (int i = R[0]; i; i = R[i]) {
256            if (colSize[i] < colSize[c]) c = i;
257        }
258        remove(c);
259        for (int i = D[c]; i != c; i = D[i]) {
260            for (int j = R[i]; j != i; j = R[j]) {
261                U[D[j]] = U[j];
262                D[U[j]] = D[j];
263                --colSize[col[j]];
264            }
265        }
266        dfs(idx + 1);
267        recover(c);
268    }
269    void dfs(int idx = 0) { // 判斷最小 dfs
270        // depth 版
271        if (R[0] == 0) {
272            resSize = min(resSize, idx); //
273            // 注意init值
274            return;
275        }
276        int c = R[0];
277        for (int i = R[0]; i; i = R[i]) {
278            if (colSize[i] < colSize[c]) c = i;
279        }
280        remove(c);
281        for (int i = D[c]; i != c; i = D[i]) {
282            for (int j = R[i]; j != i; j = R[j]) {
283                U[D[j]] = U[j];
284                D[U[j]] = D[j];
285                --colSize[col[j]];
286            }
287        }
288        dfs(idx + 1);
289        recover(c);
290    }
291    void dfs(int idx = 0) { // 判斷最小 dfs
292        // depth 版
293        if (R[0] == 0) {
294            resSize = min(resSize, idx); //
295            // 注意init值
296            return;
297        }
298        int c = R[0];
299        for (int i = R[0]; i; i = R[i]) {
300            if (colSize[i] < colSize[c]) c = i;
301        }
302        remove(c);
303        for (int i = D[c]; i != c; i = D[i]) {
304            for (int j = R[i]; j != i; j = R[j]) {
305                U[D[j]] = U[j];
306                D[U[j]] = D[j];
307                --colSize[col[j]];
308            }
309        }
310        dfs(idx + 1);
311        recover(c);
312    }
313    void dfs(int idx = 0) { // 判斷最小 dfs
314        // depth 版
315        if (R[0] == 0) {
316            resSize = min(resSize, idx); //
317            // 注意init值
318            return;
319        }
320        int c = R[0];
321        for (int i = R[0]; i; i = R[i]) {
322            if (colSize[i] < colSize[c]) c = i;
323        }
324        remove(c);
325        for (int i = D[c]; i != c; i = D[i]) {
326            for (int j = R[i]; j != i; j = R[j]) {
327                U[D[j]] = U[j];
328                D[U[j]] = D[j];
329                --colSize[col[j]];
330            }
331        }
332        dfs(idx + 1);
333        recover(c);
334    }
335    void dfs(int idx = 0) { // 判斷最小 dfs
336        // depth 版
337        if (R[0] == 0) {
338            resSize = min(resSize, idx); //
339            // 注意init值
340            return;
341        }
342        int c = R[0];
343        for (int i = R[0]; i; i = R[i]) {
344            if (colSize[i] < colSize[c]) c = i;
345        }
346        remove(c);
347        for (int i = D[c]; i != c; i = D[i]) {
348            for (int j = R[i]; j != i; j = R[j]) {
349                U[D[j]] = U[j];
350                D[U[j]] = D[j];
351                --colSize[col[j]];
352            }
353        }
354        dfs(idx + 1);
355        recover(c);
356    }
357    void dfs(int idx = 0) { // 判斷最小 dfs
358        // depth 版
359        if (R[0] == 0) {
360            resSize = min(resSize, idx); //
361            // 注意init值
362            return;
363        }
364        int c = R[0];
365        for (int i = R[0]; i; i = R[i]) {
366            if (colSize[i] < colSize[c]) c = i;
367        }
368        remove(c);
369        for (int i = D[c]; i != c; i = D[i]) {
370            for (int j = R[i]; j != i; j = R[j]) {
371                U[D[j]] = U[j];
372                D[U[j]] = D[j];
373                --colSize[col[j]];
374            }
375        }
376        dfs(idx + 1);
377        recover(c);
378    }
379    void dfs(int idx = 0) { // 判斷最小 dfs
380        // depth 版
381        if (R[0] == 0) {
382            resSize = min(resSize, idx); //
383            // 注意init值
384            return;
385        }
386        int c = R[0];
387        for (int i = R[0]; i; i = R[i]) {
388            if (colSize[i] < colSize[c]) c = i;
389        }
390        remove(c);
391        for (int i = D[c]; i != c; i = D[i]) {
392            for (int j = R[i]; j != i; j = R[j]) {
393                U[D[j]] = U[j];
394                D[U[j]] = D[j];
395                --colSize[col[j]];
396            }
397        }
398        dfs(idx + 1);
399        recover(c);
400    }
401    void dfs(int idx = 0) { // 判斷最小 dfs
402        // depth 版
403        if (R[0] == 0) {
404            resSize = min(resSize, idx); //
405            // 注意init值
406            return;
407        }
408        int c = R[0];
409        for (int i = R[0]; i; i = R[i]) {
410            if (colSize[i] < colSize[c]) c = i;
411        }
412        remove(c);
413        for (int i = D[c]; i != c; i = D[i]) {
414            for (int j = R[i]; j != i; j = R[j]) {
415                U[D[j]] = U[j];
416                D[U[j]] = D[j];
417                --colSize[col[j]];
418            }
419        }
420        dfs(idx + 1);
421        recover(c);
422    }
423    void dfs(int idx = 0) { // 判斷最小 dfs
424        // depth 版
425        if (R[0] == 0) {
426            resSize = min(resSize, idx); //
427            // 注意init值
428            return;
429        }
430        int c = R[0];
431        for (int i = R[0]; i; i = R[i]) {
432            if (colSize[i] < colSize[c]) c = i;
433        }
434        remove(c);
435        for (int i = D[c]; i != c; i = D[i]) {
436            for (int j = R[i]; j != i; j = R[j]) {
437                U[D[j]] = U[j];
438                D[U[j]] = D[j];
439                --colSize[col[j]];
440            }
441        }
442        dfs(idx + 1);
443        recover(c);
444    }
445    void dfs(int idx = 0) { // 判斷最小 dfs
446        // depth 版
447        if (R[0] == 0) {
448            resSize = min(resSize, idx); //
449            // 注意init值
450            return;
451        }
452        int c = R[0];
453        for (int i = R[0]; i; i = R[i]) {
454            if (colSize[i] < colSize[c]) c = i;
455        }
456        remove(c);
457        for (int i = D[c]; i != c; i = D[i]) {
458            for (int j = R[i]; j != i; j = R[j]) {
459                U[D[j]] = U[j];
460                D[U[j]] = D[j];
461                --colSize[col[j]];
462            }
463        }
464        dfs(idx + 1);
465        recover(c);
466    }
467    void dfs(int idx = 0) { // 判斷最小 dfs
468        // depth 版
469        if (R[0] == 0) {
470            resSize = min(resSize, idx); //
471            // 注意init值
472            return;
473        }
474        int c = R[0];
475        for (int i = R[0]; i; i = R[i]) {
476            if (colSize[i] < colSize[c]) c = i;
477        }
478        remove(c);
479        for (int i = D[c]; i != c; i = D[i]) {
480            for (int j = R[i]; j != i; j = R[j]) {
481                U[D[j]] = U[j];
482                D[U[j]] = D[j];
483                --colSize[col[j]];
484            }
485        }
486        dfs(idx + 1);
487        recover(c);
488    }
489    void dfs(int idx = 0) { // 判斷最小 dfs
490        // depth 版
491        if (R[0] == 0) {
492            resSize = min(resSize, idx); //
493            // 注意init值
494            return;
495        }
496        int c = R[0];
497        for (int i = R[0]; i; i = R[i]) {
498            if (colSize[i] < colSize[c]) c = i;
499        }
500        remove(c);
501        for (int i = D[c]; i != c; i = D[i]) {
502            for (int j = R[i]; j != i; j = R[j]) {
503                U[D[j]] = U[j];
504                D[U[j]] = D[j];
505                --colSize[col[j]];
506            }
507        }
508        dfs(idx + 1);
509        recover(c);
510    }
511    void dfs(int idx = 0) { // 判斷最小 dfs
512        // depth 版
513        if (R[0] == 0) {
514            resSize = min(resSize, idx); //
515            // 注意init值
516            return;
517        }
518        int c = R[0];
519        for (int i = R[0]; i; i = R[i]) {
520            if (colSize[i] < colSize[c]) c = i;
521        }
522        remove(c);
523        for (int i = D[c]; i != c; i = D[i]) {
524            for (int j = R[i]; j != i; j = R[j]) {
525                U[D[j]] = U[j];
526                D[U[j]] = D[j];
527                --colSize[col[j]];
528            }
529        }
530        dfs(idx + 1);
531        recover(c);
532    }
533    void dfs(int idx = 0) { // 判斷最小 dfs
534        // depth 版
535        if (R[0] == 0) {
536            resSize = min(resSize, idx); //
537            // 注意init值
538            return;
539        }
540        int c = R[0];
541        for (int i = R[0]; i; i = R[i]) {
542            if (colSize[i] < colSize[c]) c = i;
543        }
544        remove(c);
545        for (int i = D[c]; i != c; i = D[i]) {
546            for (int j = R[i]; j != i; j = R[j]) {
547                U[D[j]] = U[j];
548                D[U[j]] = D[j];
549                --colSize[col[j]];
550            }
551        }
552        dfs(idx + 1);
553        recover(c);
554    }
555    void dfs(int idx = 0) { // 判斷最小 dfs
556        // depth 版
557        if (R[0] == 0) {
558            resSize = min(resSize, idx); //
559            // 注意init值
560            return;
561        }
562        int c = R[0];
563        for (int i = R[0]; i; i = R[i]) {
564            if (colSize[i] < colSize[c]) c = i;
565        }
566        remove(c);
567        for (int i = D[c]; i != c; i = D[i]) {
568            for (int j = R[i]; j != i; j = R[j]) {
569                U[D[j]] = U[j];
570                D[U[j]] = D[j];
571                --colSize[col[j]];
572            }
573        }
574        dfs(idx + 1);
575        recover(c);
576    }
577    void dfs(int idx = 0) { // 判斷最小 dfs
578        // depth 版
579        if (R[0] == 0) {
580            resSize = min(resSize, idx); //
581            // 注意init值
582            return;
583        }
584        int c = R[0];
585        for (int i = R[0]; i; i = R[i]) {
586            if (colSize[i] < colSize[c]) c = i;
587        }
588        remove(c);
589        for (int i = D[c]; i != c; i = D[i]) {
590            for (int j = R[i]; j != i; j = R[j]) {
591                U[D[j]] = U[j];
592                D[U[j]] = D[j];
593                --colSize[col[j]];
594            }
595        }
596        dfs(idx + 1);
597        recover(c);
598    }
599    void dfs(int idx = 0) { // 判斷最小 dfs
600        // depth 版
601        if (R[0] == 0) {
602            resSize = min(resSize, idx); //
603            // 注意init值
604            return;
605        }
606        int c = R[0];
607        for (int i = R[0]; i; i = R[i]) {
608            if (colSize[i] < colSize[c]) c = i;
609        }
610        remove(c);
611        for (int i = D[c]; i != c; i = D[i]) {
612            for (int j = R[i]; j != i; j = R[j]) {
613                U[D[j]] = U[j];
614                D[U[j]] = D[j];
615                --colSize[col[j]];
616            }
617        }
618        dfs(idx + 1);
619        recover(c);
620    }
621    void dfs(int idx = 0) { // 判斷最小 dfs
622        // depth 版
623        if (R[0] == 0) {
624            resSize = min(resSize, idx); //
625            // 注意init值
626            return;
627        }
628        int c = R[0];
629        for (int i = R[0]; i; i = R[i]) {
630            if (colSize[i] < colSize[c]) c = i;
631        }
632        remove(c);
633        for (int i = D[c]; i != c; i = D[i]) {
634            for (int j = R[i]; j != i; j = R[j]) {
635                U[D[j]] = U[j];
636                D[U[j]] = D[j];
637                --colSize[col[j]];
638            }
639        }
640        dfs(idx + 1);
641        recover(c);
642    }
643    void dfs(int idx = 0) { // 判斷最小 dfs
644        // depth 版
645        if (R[0] == 0) {
646            resSize = min(resSize, idx); //
647            // 注意init值
648            return;
649        }
650        int c = R[0];
651        for (int i = R[0]; i; i = R[i]) {
652            if (colSize[i] < colSize[c]) c = i;
653        }
654        remove(c);
655        for (int i = D[c]; i != c; i = D[i]) {
656            for (int j = R[i]; j != i; j = R[j]) {
657                U[D[j]] = U[j];
658                D[U[j]] = D[j];
659                --colSize[col[j]];
660            }
661        }
662        dfs(idx + 1);
663        recover(c);
664    }
665    void dfs(int idx = 0) { // 判斷最小 dfs
666        // depth 版
667        if (R[0] == 0) {
668            resSize = min(resSize, idx); //
669            // 注意init值
670            return;
671        }
672        int c = R[0];
673        for (int i = R[0]; i; i = R[i]) {
674            if (colSize[i] < colSize[c]) c = i;
675        }
676        remove(c);
677        for (int i = D[c]; i != c; i = D[i]) {
678            for (int j = R[i]; j != i; j = R[j]) {
679                U[D[j]] = U[j];
680                D[U[j]] = D[j];
681                --colSize[col[j]];
682            }
683        }
684        dfs(idx + 1);
685        recover(c);
686    }
687    void dfs(int idx = 0) { // 判斷最小 dfs
688        // depth 版
689        if (R[0] == 0) {
690            resSize = min(resSize, idx); //
691            // 注意init值
692            return;
693        }
694        int c = R[0];
695        for (int i = R[0]; i; i = R[i]) {
696            if (colSize[i] < colSize[c]) c = i;
697        }
698        remove(c);
699        for (int i = D[c]; i != c; i = D[i]) {
700            for (int j = R[i]; j != i; j = R[j]) {
701                U[D[j]] = U[j];
702                D[U[j]] = D[j];
703                --colSize[col[j]];
704            }
705        }
706        dfs(idx + 1);
707        recover(c);
708    }
709    void dfs(int idx = 0) { // 判斷最小 dfs
710        // depth 版
711        if (R[0] == 0) {
712            resSize = min(resSize, idx); //
713            // 注意init值
714            return;
715        }
716        int c = R[0];
717        for (int i = R[0]; i; i = R[i]) {
718            if (colSize[i] < colSize[c]) c = i;
719        }
720        remove(c);
721        for (int i = D[c]; i != c; i = D[i]) {
722            for (int j = R[i]; j != i; j = R[j]) {
723                U[D[j]] = U[j];
724                D[U[j]] = D[j];
725                --colSize[col[j]];
726            }
727        }
728        dfs(idx + 1);
729        recover(c);
730    }
731    void dfs(int idx = 0) { // 判斷最小 dfs
732        // depth 版
733        if (R[0] == 0) {
734            resSize = min(resSize, idx); //
735            // 注意init值
736            return;
737        }
738        int c = R[0];
739        for (int i = R[0]; i; i = R[i]) {
740            if (colSize[i] < colSize[c]) c = i;
741        }
742        remove(c);
743        for (int i = D[c]; i != c; i = D[i]) {
744            for (int j = R[i]; j != i; j = R[j]) {
745                U[D[j]] = U[j];
746                D[U[j]] = D[j];
747                --colSize[col[j]];
748            }
749        }
750        dfs(idx + 1);
751        recover(c);
752    }
753    void dfs(int idx = 0) { // 判斷最小 dfs
754        // depth 版
755        if (R[0] == 0) {
756            resSize = min(resSize, idx); //
757            // 注意init值
758            return;
759        }
760        int c = R[0];
761        for (int i = R[0]; i; i = R[i]) {
762            if (colSize[i] < colSize[c]) c = i;
763        }
764        remove(c);
765        for (int i = D[c]; i != c; i = D[i]) {
766            for (int j = R[i]; j != i; j = R[j]) {
767                U[D[j]] = U[j];
768                D[U[j]] = D[j];
769                --colSize[col[j]];
770            }
771        }
772        dfs(idx + 1);
773        recover(c);
774    }
775    void dfs(int idx = 0) { // 判斷最小 dfs
776        // depth 版
777        if (R[0] == 0) {
778            resSize = min(resSize, idx); //
779            // 注意init值
780            return;
781        }
782        int c = R[0];
783        for (int i = R[0]; i; i = R[i]) {
784            if (colSize[i] < colSize[c]) c = i;
785        }
786        remove(c);
787        for (int i = D[c]; i != c; i = D[i]) {
788            for (int j = R[i]; j != i; j = R[j]) {
789                U[D[j]] = U[j];
790                D[U[j]] = D[j];
791                --colSize[col[j]];
792            }
793        }
794        dfs(idx + 1);
795        recover(c);
796    }
797    void dfs(int idx = 0) { // 判斷最小 dfs
798        // depth 版
799        if (R[0] == 0) {
800            resSize = min(resSize, idx); //
801            // 注意init值
802            return;
803        }
804        int c = R[0];
805        for (int i = R[0]; i; i = R[i]) {
806            if (colSize[i] < colSize[c]) c = i;
807        }
808        remove(c);
809        for (int i = D[c]; i != c; i = D[i]) {
810            for (int j = R[i]; j != i; j = R[j]) {
811                U[D[j]] = U[j];
812                D[U[j]] = D[j];
813                --colSize[col[j]];
814            }
815        }
816        dfs(idx + 1);
817        recover(c);
818    }
819    void dfs(int idx = 0) { // 判斷最小 dfs
820        // depth 版
821        if (R[0] == 0) {
822            resSize = min(resSize, idx); //
823            // 注意init值
824            return;
825        }
826        int c = R[0];
827        for (int i = R[0]; i; i = R[i]) {
828            if (colSize[i] < colSize[c]) c = i;
829        }
830        remove(c);
831        for (int i = D[c]; i != c; i = D[i]) {
832            for (int j = R[i]; j != i; j = R[j]) {
833                U[D[j]] = U[j];
834                D[U[j]] = D[j];
835                --colSize[col[j]];
836            }
837        }
838        dfs(idx + 1);
839        recover(c);
840    }
841    void dfs(int idx = 0) { // 判斷最小 dfs
842        // depth 版
843        if (R[0] == 0) {
844            resSize = min(resSize, idx); //
845            // 注意init值
846            return;
847        }
848        int c = R[0];
849        for (int i = R[0]; i; i = R[i]) {
850            if (colSize[i] < colSize[c]) c = i;
851        }
852        remove(c);
853        for (int i = D[c]; i != c; i = D[i]) {
854            for (int j = R[i]; j != i; j = R[j]) {
855                U[D[j]] = U[j];
856                D[U[j]] = D[j];
857                --colSize[col[j]];
858            }
859        }
860        dfs(idx + 1);
861        recover(c);
862    }
863    void dfs(int idx = 0) { // 判斷最小 dfs
864        // depth 版
865        if (R[0] == 0) {
866            resSize = min(resSize, idx); //
867            // 注意init值
868            return;
869        }
870        int c = R[0];
871        for (int i = R[0]; i; i = R[i]) {
872            if (colSize[i] < colSize[c]) c = i;
873        }
874        remove(c);
875        for (int i = D[c]; i != c; i = D[i]) {
876            for (int j = R[i]; j != i; j = R[j]) {
877                U[D[j]] = U[j];
878                D[U[j]] = D[j];
879                --colSize[col[j]];
880            }
881        }
882        dfs(idx + 1);
883        recover(c);
884    }
885    void dfs(int idx = 0) { // 判斷最小 dfs
886        // depth 版
887        if (R[0] == 0) {
888            resSize = min(resSize, idx); //
889            // 注意init值
890            return;
891        }
892        int c = R[0];
893        for (int i = R[0]; i; i = R[i]) {
894            if (colSize[i] < colSize[c]) c = i;
895        }
896        remove(c);
897        for (int i = D[c]; i != c; i = D[i]) {
898            for (int j = R[i]; j != i; j = R[j]) {
899                U[D[j]] = U[j];
900                D[U[j]] = D[j];
901                --colSize[col[j]];
902            }
903        }
904        dfs(idx + 1);
905        recover(c);
906    }
907    void dfs(int idx = 0) { // 判斷最小 dfs
908        // depth 版
909        if (R[0] == 0) {
910            resSize = min(resSize, idx); //
911            // 注意init值
912            return;
913        }
914        int c = R[0];
915        for (int i = R[0]; i; i = R[i]) {
916            if (colSize[i] < colSize[c]) c = i;
917        }
918        remove(c);
919        for (int i = D[c]; i != c; i = D[i]) {
920            for (int j = R[i]; j != i; j = R[j]) {
921                U[D[j]] = U[j];
922                D[U[j]] = D[j];
923                --colSize[col[j]];
924            }
925        }
926        dfs(idx + 1);
927        recover(c);
928    }
929    void dfs(int idx = 0) { // 判斷最小 dfs
930        // depth 版
931        if (R[0] == 0) {
932            resSize = min(resSize, idx); //
933            // 注意init值
934            return;
935        }
936        int c = R[0];
937        for (int i = R[0]; i; i = R[i]) {
938            if (colSize[i] < colSize[c]) c = i;
939        }
940        remove(c);
941        for (int i = D[c]; i != c; i = D[i]) {
942            for (int j = R[i]; j != i; j = R[j]) {
943                U[D[j]] = U[j];
944                D[U[j]] = D[j];
945                --colSize[col[j]];
946            }
947        }
948        dfs(idx + 1);
949        recover(c);
950    }
951    void dfs(int idx = 0) { // 判斷最小 dfs
952        // depth 版
953        if (R[0] == 0) {
954            resSize = min(resSize, idx); //
955            // 注意init值
956            return;
957        }
958        int c = R[0];
959        for (int i = R[0]; i; i = R[i]) {
960            if (colSize[i] < colSize[c]) c = i;
961        }
962        remove(c);
963        for (int i = D[c]; i != c; i = D[i]) {
964            for (int j = R[i]; j != i; j = R[j]) {
965                U[D[j]] = U[j];
966                D[U[j]] = D[j];
967                --colSize[col[j]];
968            }
969        }
970        dfs(idx + 1);
971        recover(c);
972    }
973    void dfs(int idx = 0) { // 判斷最小 dfs
9
```

## 4.17 LCA 倍增法

```

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,
85                 sellPrice, sellMax,
86                 inventoryMonth;
87             scanf("%d %d %d %d %d",
88                 &produceCost, &produceMax,
89                 &sellPrice, &sellMax,
90                 &inventoryMonth);
91             addEdge(s, i, produceMax,
92                 produceCost);
93             addEdge(M + i, t, sellMax,
94                 -sellPrice);
95             for (int j = 0; j <=
96                 inventoryMonth; ++j) {
97                 if (i + j <= M)
98                     addEdge(i, M + i + j, INF,
99                         I * j);
100             }
101         }
102         printf("Case %d: %lld\n", Case,
103             -MCMF());
104     }
105     return 0;
106 }

```

```

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 { int u, v, w; };
5 vector<vector<Edge>> G; // tree
6 int fa[maxn][31]; //fa[u][i] -> u的第 $2^i$ 個祖先
7 long long dis[maxn][31];
8 int dep[maxn]; //深度
9 void dfs(int u, int p) { //預處理fa
10     fa[u][0] = p; //因為u的第 $2^0 = 1$ 的祖先就是p
11     dep[u] = dep[p] + 1;
12     //第 $2^i$ 的祖先是(第 $2^{i-1}$ 個祖先)的
13     //第 $2^{i-1}$ 的祖先
14     //ex: 第8個祖先是 (第4個祖先) 的第4個祖先
15     for (int i = 1; i < 31; ++i) {
16         fa[u][i] = fa[fa[u][i-1]][i-1];
17         dis[u][i] = dis[fa[u][i-1]][i-1]
18             + dis[u][i-1];
19     }
20     //遍歷子節點
21     for (Edge& edge: G[u]) {
22         if (edge.v == p) continue;
23         dis[edge.v][0] = edge.w;
24         dfs(edge.v, u);
25     }
26 long long lca(int x, int y) {
27     //此函數是找lca同時計算x、y的距離 -> dis(x,
28     //lca) + dis(lca, y)
29     //讓y比x深
30     if (dep[x] > dep[y])
31         swap(x, y);
32     int deltaDep = dep[y] - dep[x];
33     long long res = 0;
34     //讓y與x在同一個深度
35     for (int i = 0; deltaDep != 0; ++i,
36         deltaDep >>= 1)
37         if (deltaDep & 1)
38             res += dis[y][i], y = fa[y][i];
39     if (y == x) //x = y -> x、y彼此是彼此的祖先
40         return res;
41     //往上找，一起跳，但x、y不能重疊
42     for (int i = 30; i >= 0 && y != x; --i) {
43         if (fa[x][i] != fa[y][i]) {
44             res += dis[x][i] + dis[y][i];
45             x = fa[x][i];
46             y = fa[y][i];
47         }
48     }
49     //最後發現不能跳了，此時x的第 $2^0 = 1$ 
50     //個祖先(或說y的第 $2^0 = 1$ 的祖先)即為x、y的lca
51     res += dis[x][0] + dis[y][0];
52     return res;
53 }
54 int main() {
55     int n, q;
56     while (~scanf("%d", &n) && n) {
57         int v, w;
58         G.assign(n + 5, vector<Edge>());
59         for (int i = 1; i <= n - 1; ++i) {
60             scanf("%d %d", &v, &w);
61             G[i + 1].push_back({i + 1, v + 1, w});
62             G[v + 1].push_back({v + 1, i + 1, w});
63         }
64         dfs(1, 0);
65         scanf("%d", &q);
66         int u;
67         while (q--) {
68             scanf("%d %d", &u, &v);
69             printf("%lldc", lca(u + 1, v +
70                 1), (q ? ' ': '\n'));
71         }
72     }
73 }

```

## 4.18 LCA 樹壓平 RMQ

```

1 //樹壓平求LCA RMQ(sparse table
2 // $O(n\log n)$ 建立， $O(1)$ 查詢)，求任意兩點距離，
3 //如果用笛卡兒樹可以壓到 $O(n)$ 建立， $O(1)$ 查詢
4 //理論上可以過，但遇到直鏈的case dfs深度會stack
5 //overflow
6 #define maxn 100005
7 struct Edge {
8     int u, v, w;
9 };
10 int dep[maxn], pos[maxn];
11 long long dis[maxn];
12 int st[maxn * 2][32]; //sparse table
13 int realLCA[maxn * 2][32];
14 //最小深度對應的節點，及真正的LCA
15 int Log[maxn]; //取代std::log2
16 int tp; // timestamp
17 vector<vector<Edge>> G; // tree
18 void calLog() {
19     Log[1] = 0;
20     Log[2] = 1;
21     for (int i = 3; i < maxn; ++i)
22         Log[i] = Log[i / 2] + 1;
23 }
24 void buildST() {
25     for (int j = 0; Log[tp]; ++j) {
26         for (int i = 0; i + (1 << j) - 1 < tp;
27             ++i) {
28             if (st[i - 1][j] < st[i - 1][j + (1 <<
29                 i - 1)]) {
30                 st[i][j] = st[i - 1][j];
31                 realLCA[i][j] = realLCA[i - 1][j];
32             }
33             else {
34                 st[i][j] = st[i - 1][j + (1 << i -
35                     1)];
36                 realLCA[i][j] = realLCA[i - 1][j + (1
37                     << i - 1)];
38             }
39         }
40     }
41 }
42 //  $O(n\log n)$ 
43 int query(int l, int r) { // [l, r] min
44     depth即為lca的深度
45     int k = Log[r - l + 1];
46     if (st[l][k] < st[r - (1 << k) + 1][k])
47         return realLCA[l][k];
48     else
49         return realLCA[r - (1 << k) + 1][k];
50 }
51 void dfs(int u, int p) { //euler tour
52     pos[u] = tp;
53     st[tp][0] = dep[u];
54     realLCA[tp][0] = dep[u];
55     ++tp;
56     for (int i = 0; i < G[u].size(); ++i) {
57         Edge& edge = G[u][i];
58         if (edge.v == p) continue;
59         dep[edge.v] = dep[u] + 1;
60         dis[edge.v] = dis[edge.u] + edge.w;
61         dfs(edge.v, u);
62         st[tp++][0] = dep[u];
63     }
64 }
65 long long getDis(int u, int v) {
66     if (pos[u] > pos[v])
67         swap(u, v);
68     int lca = query(pos[u], pos[v]);
69     return dis[u] + dis[v] - 2 *
70         dis[query(pos[u], pos[v])];
71 }
72 int main() {
73     int n, q;
74     calLog();
75     while (~scanf("%d", &n) && n) {
76         int v, w;
77         G.assign(n + 5, vector<Edge>());
78     }
79 }

```

```

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("%lldc", getDis(u, v),
81             (q) ? ' ' : '\n');
82     }
83     return 0;
84 }

```

## 4.19 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, 找任兩點的中點,
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 ||
29                 subtreeSize[hson[u]] <
30                 subtreeSize[v]) {
31                 hson[u] = v;
32             }
33         }
34     }
35 }
36 //實際剖分 <- 參數t是top的意思
37 //t初始應為root本身
38 void dfs2(int u, int t) {
39     top[u] = t;
40     dfn[u] = dfsTime;
41     dfnToNode[dfsTime] = u;
42     ++dfsTime;
43     //葉子點 -> 沒有重兒子
44     if (hson[u] == -1)
45         return;
46     //優先對重兒子dfs, 才能保證同一重鍊dfn連續
47     dfs2(hson[u], t);
48     for (int v: G[u]) {
49         if (v != parent[u] && v != hson[u])
50             dfs2(v, v);
51     }
52 }
53 //不斷跳鍊, 當跳到同一條鍊時, 深度小的即為LCA
54 //跳鍊時優先鍊頂深度大的跳
55 int LCA(int u, int v) {
56     while (top[u] != top[v]) {
57

```

```

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
88                 * depth[lca];
89             //讓v比u深或等於
90             if (depth[u] > depth[v])
91                 swap(u, v);
92             if (u == v) {
93                 printf("The fleas meet at
94                     %d.\n", u);
95             }
96             else if (dis % 2 == 0) {
97                 //路徑長是even -> 有中點
98                 printf("The fleas meet at
99                     %d.\n", getK_parent(v,
100                         dis / 2));
101             }
102             else {
103                 //路徑長是odd -> 沒有中點
104                 if (depth[u] == depth[v]) {
105                     int x = getK_parent(u, dis
106                         / 2);
107                     int y = getK_parent(v, dis
108                         / 2);
109                     if (x > y) swap(x, y);
110                     printf("The fleas jump
111                         forever between %d
112                         and %d.\n", x, y);
113                 }
114                 else {
115                     //技巧: 讓深的點v往上dis /
116                     2步 = y,
117                     //這個點的parent設為x
118                     //此時的x、y就是答案要的中點兩點
119                     //主要是往下不好找, 所以改用深的點用parent往上
120                     int y = getK_parent(v, dis
121                         / 2);
122                     int x = getK_parent(y, 1);
123                     if (x > y) swap(x, y);
124                     printf("The fleas jump
125                         forever between %d
126                         and %d.\n", x, y);
127                 }
128             }
129         }
130     }
131     return 0;
132 }

```

## 5 DataStructure

### 5.1 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):
9         size(sz), bit(sz+1), 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     /** The sum of all values in [0, idx]. */
24     T pre_sum(int idx) {
25         T total = 0;
26         for (++idx; idx>0; idx--=(idx & -idx))
27             total += bit[idx];
28         return total;
29     }
30 };

```

### 5.2 帶權併查集

- $val[x]$  為  $x$  到  $p[x]$  的距離 (隨題目變化更改)
- $merge(u, v, w)$   
 $u \xrightarrow{w} v$   
 $pu = pv$  時,  $val[v] - val[u] \neq w$  代表有誤
- 若  $[l, r]$  的總和為  $w$ , 則應呼叫  $merge(l-1, r, w)$

```

1 const int maxn = 2e5 + 10;
2
3 int p[maxn], val[maxn];
4
5 int findP(int x) {
6     if(p[x] == -1) return x;
7     int par = findP(p[x]);
8     val[x] += val[p[x]]; //依題目更新val[x]
9     return p[x] = par;
10 }
11
12 void merge(int u, int v, int w) {
13     int pu = findP(u);
14     int pv = findP(v);
15     if(pu == pv) {
16         // 理論上 val[v]-val[u] == w
17         // 依題目判斷 error 的條件
18         return;
19     }
20     val[pv] = val[u] - val[v] + w;
21     p[pv] = pu;
22 }

```

### 5.3 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 }
64 //如果是直接改值而不是加值, query與update中的tag與st
65 //改值從+=改成=

```

### 5.4 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
6     val, int yPos, int xIndex, bool
7     xIsLeaf) {
8     if (l == r) {
9         if (xIsLeaf) {
10             maxST[xIndex][index] =
11                 minST[xIndex][index] = val;
12             return;
13         }
14         maxST[xIndex][index] =
15             max(maxST[xIndex * 2][index],
16                 maxST[xIndex * 2 + 1][index]);
17         minST[xIndex][index] =
18             min(minST[xIndex * 2][index],
19                 minST[xIndex * 2 + 1][index]);
20     }
21     else {
22         int mid = (l + r) / 2;
23         if (yPos <= mid)
24             modifyY(index * 2, l, mid, val,
25                 yPos, xIndex, xIsLeaf);
26         else
27             modifyY(index * 2 + 1, mid + 1,
28                 r, val, yPos, xIndex,
29                 xIsLeaf);
30         maxST[xIndex][index] =
31             max(maxST[xIndex][index * 2],
32                 maxST[xIndex][index * 2 + 1]);
33         minST[xIndex][index] =
34             min(minST[xIndex][index * 2],
35                 minST[xIndex][index * 2 + 1]);
36     }
37 }
38 void modifyX(int index, int l, int r, int
39     val, int xPos, int yPos) {
40     if (l == r) {
41         modifyY(1, 1, N, val, yPos, index,
42             true);
43     }
44     else {
45         int mid = (l + r) / 2;
46         if (xPos <= mid)
47             modifyX(index * 2, l, mid, val,
48                 xPos, yPos);
49         else
50             modifyX(index * 2 + 1, mid + 1,
51                 r, val, xPos, yPos);
52         modifyY(1, 1, N, val, yPos, index,
53             false);
54     }
55 }
56 void queryY(int index, int l, int r, int
57     yql, int yqr, int xIndex, int& vmax,
58     int& vmin) {
59     if (yql <= l && r <= yqr) {
60         vmax = max(vmax,
61             maxST[xIndex][index]);
62         vmin = min(vmin,
63             minST[xIndex][index]);
64     }
65     else
66     {
67         int mid = (l + r) / 2;
68         if (yql <= mid)
69             queryY(index * 2, l, mid, yql,
70                 yqr, xIndex, vmax, vmin);
71         if (mid < yqr)
72             queryY(index * 2 + 1, mid + 1, r,
73                 yql, yqr, xIndex, vmax,
74                 vmin);
75     }
76 }

```



## 5.5 權值線段樹

## 5.6 ChthollyTree

```

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

```

```

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

```

```

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
        val)
6         : l(l), r(r), val(val){}
7     bool operator<(const Node& other) const {
8         return this->l < other.l;
9     }
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 =
        chthollyTree.lower_bound(Node(pos,
            0, 0));
16     //運氣很好直接找到左端點是pos的區間
17     if (it != chthollyTree.end() && it->l ==
        pos)
18         return it;
19     //到這邊代表找到的是第一個左端點大於pos的區間
20     //it - 1即可找到左端點等於pos的區間
21     //(不會是別的, 因為沒有重疊的區間)
22     --it;
23     long long l = it->l, r = it->r;
24     long long val = it->val;
25     chthollyTree.erase(it);
26     chthollyTree.insert(Node(l, pos-1, val));
27     //回傳左端點是pos的區間iterator
28     return chthollyTree.insert(Node(pos, r,
        val)).first;
29 }
30 //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可能會改變
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) return p.first;
61     }
62     //不應該跑到這
63     return -1;
}

```

## 5.8 Trie

## 5.9 AC Trie

## 5.7 單調隊列

```

64 }
65 //快速幂
66 long long qpow(long long x, long long n,
67 long long mod) {
68     long long res = 1;
69     while (n) {
70         if (n & 1) res = res * x % mod;
71         n >>= 1;
72         x = x * x % mod;
73     }
74     return res;
75 }
76 //區間n次方和
77 long long sumOfPow(long long l, long long r,
78 long long n, long long mod) {
79     long long total = 0;
80     set<Node>::iterator end = split(r + 1);
81     set<Node>::iterator begin = split(l);
82     for (set<Node>::iterator it = begin; it
83         != end; ++it) {
84         total = (total + qpow(it->val, n, mod) *
85             (it->r - it->l + 1)) % mod;
86     }
87     return total;
88 }

```

5.7 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 給出一個長度為 n 的數組，  
6 輸出每 k 個連續的數中的最大值和最小值。  
7  
8 #define maxn 1000100  
9 int q[maxn], a[maxn];  
10 int n, k;  
11 //得到這個隊列裡的最小值，直接找到最後的就行了  
12 void getmin() {  
13 int head=0, tail=0;  
14 for(int i=1; i<=n; i++) {  
15 while(head<=tail && a[q[tail]]>=a[i])  
16 tail--;  
17 q[++tail]=i;  
18 }  
19 for(int i=k; i<=n; i++) {  
20 while(head<=tail && a[q[tail]]>=a[i])  
21 tail--;  
22 q[++tail]=i;  
23 while(q[head]<=i-k) head++;  
24 cout<<a[q[head]]<<" ";  
25 }  
26 cout<<endl;  
27 }  
28 //和上面同理  
29 void getmax() {  
30 int head=0, tail=0;  
31 for(int i=1; i<=n; i++) {  
32 while(head<=tail && a[q[tail]]<=a[i]) tail--;  
33 q[++tail]=i;  
34 }  
35 for(int i=k; i<=n; i++) {  
36 while(head<=tail && a[q[tail]]<=a[i]) tail--;  
37 q[++tail]=i;  
38 while(q[head]<=i-k) head++;  
39 cout<<a[q[head]]<<" ";  
40 }  
41 cout<<endl;  
42 }  
43 int main(){  
44 cin>>n>>k; //每k個連續的數  
45 for(int i=1; i<=n; i++) cin>>a[i];  
46 getmin();  
47 getmax();  
48 }

```

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             cur = cur->child[c];
49         }
50         return cur->cnt || prefix;
51     }
52
53     // 字典裡有 word 的前綴為 prefix
54     bool startsWith(string prefix) {
55         return search(prefix, true);
56     }
57 };

```

```

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)

```

## 6 Geometry

### 6.1 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  $\Delta$

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  $(H_x, H_y)$

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

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

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

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

$$H_x = -\frac{\begin{vmatrix} x_2x_3 + y_2y_3 & y_1 & 1 \\ x_1x_3 + y_1y_3 & y_2 & 1 \\ x_1x_2 + y_1y_2 & y_3 & 1 \end{vmatrix}}{2\Delta}$$

$$H_y = -\frac{\begin{vmatrix} x_1 & x_2x_3 + y_2y_3 & 1 \\ x_2 & x_1x_3 + y_1y_3 & 1 \\ x_3 & x_1x_2 + y_1y_2 & 1 \end{vmatrix}}{2\Delta}$$

任意三角形，重心、外心、垂心共線

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

### 6.2 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 };
```

### 6.3 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 }
```

### 6.4 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 }
```

### 6.5 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 dbldcmp
4     auto dbldcmp=[](DBL v){return (v>0)-(v<0)};
5     // 不包含線上，改 '>=' 為 '>'
6     auto test = [&](Point& p0, Point& p1) {
7         return dbldcmp(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 }
```

## 6.6 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 }

```

## 6.7 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:
  - ang, 該點與基準點的 atan2 值
  - d2, 該點與基準點的 (距離)<sup>2</sup>
- 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 }

```

## 6.8 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，沒符號數不用
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 }

```

## 7 DP

### 7.1 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 ->
    當目前只考慮到第i個商品時，達到獲利j時最少的weight
    = w
9 所以答案是dp[n][1 ~ maxv]找價值最大且裝的下的*/
10 #define maxn 105
11 #define maxv 100005
12 long long dp[maxn][maxv];
13 long long weight[maxn];
14 long long v[maxn];
15 int main() {
16     int n;
17     long long w;
18     scanf("%d %lld", &n, &w);
19     for (int i = 1; i <= n; ++i) {
20         scanf("%lld %lld", &weight[i], &v[i]);
21     }
22     memset(dp, 0x3f, sizeof(dp));
23     dp[0][0] = 0;
24     for (int i = 1; i <= n; ++i) {
25         for (int j = 0; j <= maxv; ++j) {
26             if (j - v[i] >= 0)
27                 dp[i][j] = dp[i - 1][j - v[i]] + weight[i];
28             dp[i][j] = min(dp[i - 1][j], dp[i][j]);
29         }
30     }
31     long long res = 0;
32     for (int j = maxv - 1; j >= 0; --j) {
33         if (dp[n][j] <= w) {
34             res = j;
35             break;
36         }
37     }
38     printf("%lld\n", res);
39     return 0;
40 }

```

### 7.2 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) != EOF) {
21         memset(dp, -1, sizeof(dp));
22         printf("%lld\n", dfs(N, K));
23     }
24 }

```

### 7.3 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) + solve(m, j) + cuts[j] - cuts[i]);
15    }
16    return dp[i][j] = cost;
17 }
18 int main() {
19     int l, n;
20     while (scanf("%d", &l) != EOF && l) {
21         scanf("%d", &n);
22         for (int i = 1; i <= n; ++i)
23             scanf("%d", &cuts[i]);
24         cuts[0] = 0;
25         cuts[n + 1] = 1;
26         memset(dp, -1, sizeof(dp));
27         printf("ans = %d.\n", solve(0, n + 1));
28     }
29     return 0;
30 }
31 }

```

### 7.4 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];

```

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

```

### 7.6 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

```

### 7.7 stringDP

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

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

Longest Palindromic Subsequence

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

### 7.8 樹 DP 有幾個 path 長度為 k

```

1 #define maxn 50005
2 #define maxx 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;
26        for (int x = 0; x <= k - 2; ++x) {
27            cnt += dp[v][x] * (dp[u][k - x - 1] - dp[v][k - x - 2]);
28        }
29    }
30    res += cnt / 2;
31 }
32 int main() {
33     ...
34     dfs(1, -1);
35     printf("%lld\n", res);
36     return 0;
37 }
38 }

```



## 7.9 TreeDP reroot

## 7.10 WeightedLIS

```

1  /*re-root dp on tree  $O(n + n + n) \rightarrow O(n)*$ 
2  class Solution {
3  public:
4      vector<int> sumOfDistancesInTree(int n,
5          vector<vector<int>>& edges) {
6          this->res.assign(n, 0);
7          G.assign(n + 5, vector<int>());
8          for (vector<int>& edge: edges) {
9              G[edge[0]].emplace_back(edge[1]);
10             G[edge[1]].emplace_back(edge[0]);
11         }
12         memset(this->visited, 0,
13             sizeof(this->visited));
14         this->dfs(0);
15         memset(this->visited, 0,
16             sizeof(this->visited));
17         this->dfs3(0, n);
18         return this->res;
19     }
20 private:
21     vector<vector<int>> G;
22     bool visited[30005];
23     int subtreeSize[30005];
24     vector<int> res;
25     //求subtreeSize
26     int dfs(int u) {
27         this->visited[u] = true;
28         for (int v: this->G[u])
29             if (!this->visited[v])
30                 this->subtreeSize[u] +=
31                     this->dfs(v);
32         //自己
33         this->subtreeSize[u] += 1;
34         return this->subtreeSize[u];
35     }
36     //求res[0], 0到所有點的距離
37     int dfs2(int u, int dis) {
38         this->visited[u] = true;
39         int sum = 0;
40         for (int v: this->G[u])
41             if (!visited[v])
42                 sum += this->dfs2(v, dis + 1);
43         //要加上自己的距離
44         return sum + dis;
45     }
46     //算出所有的res
47     void dfs3(int u, int n) {
48         this->visited[u] = true;
49         for (int v: this->G[u]) {
50             if (!visited[v]) {
51                 this->res[v] = this->res[u] +
52                     n - 2 *
53                     this->subtreeSize[v];
54                 this->dfs3(v, n);
55             }
56         }
57     }
58 }
59 };

```

```

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 }

```

## 77

77					155							
78					156							
79	-----					157	-----					
80					158							
81					159							
82	-----					160	-----					
83					161							
84					162							
85	-----					163	-----					
86					164							
87					165							
88	-----					166	-----					
89					167							
90					168							
91	-----					169	-----					
92					170							
93					171							
94	-----					172	-----					
95					173							
96					174							
97	-----					175	-----					
98					176							
99					177							
100	-----					178	-----					
101					179							
102					180							
103	-----					181	-----					
104					182							
105					183							
106	-----					184	-----					
107					185							
108					186							
109	-----					187	-----					
110					188							
111					189							
112	-----					190	-----					
113					191							
114					192							
115	-----					193	-----					
116					194							
117					195							
118	-----					196	-----					
119					197							
120					198							
121	-----					199	-----					
122					200							
123					201							
124	-----					202	-----					
125					203							
126					204							
127	-----					205	-----					
128					206							
129					207							
130	-----					208	-----					
131					209							
132					210							
133	-----					211	-----					
134					212							
135					213							
136	-----					214	-----					
137					215							
138					216							
139	-----					217	-----					
140					218							
141					219							
142	-----					220	-----					
143					221							
144					222							
145	-----					223	-----					
146					224							
147					225							
148	-----					226	-----					
149					227							
150					228							
151	-----					229	-----					
152					230							
153					231							
154	-----					232	-----					

233						311							389						
234						312							390						
235	-----					313	-----					391	-----						
236						314							392						
237						315							393						
238	-----					316	-----					394	-----						
239						317							395						
240						318							396						
241	-----					319	-----					397	-----						
242						320							398						
243						321							399						
244	-----					322	-----					400	-----						
245						323							401						
246						324							402						
247	-----					325	-----					403	-----						
248						326							404						
249						327							405						
250	-----					328	-----					406	-----						
251						329							407						
252						330							408						
253	-----					331	-----					409	-----						
254						332							410						
255						333							411						
256	-----					334	-----					412	-----						
257						335							413						
258						336							414						
259	-----					337	-----					415	-----						
260						338							416						
261						339							417						
262	-----					340	-----					418	-----						
263						341							419						
264						342							420						
265	-----					343	-----					421	-----						
266						344							422						
267						345							423						
268	-----					346	-----					424	-----						
269						347							425						
270						348							426						
271	-----					349	-----					427	-----						
272						350							428						
273						351							429						
274	-----					352	-----					430	-----						
275						353							431						
276						354							432						
277	-----					355	-----					433	-----						
278						356							434						
279						357							435						
280	-----					358	-----					436	-----						
281						359							437						
282						360							438						
283	-----					361	-----					439	-----						
284						362							440						
285						363							441						
286	-----					364	-----					442	-----						
287						365							443						
288						366							444						
289	-----					367	-----					445	-----						
290						368							446						
291						369							447						
292	-----					370	-----					448	-----						
293						371							449						
294						372							450						
295	-----					373	-----					451	-----						
296						374							452						
297						375							453						
298	-----					376	-----					454	-----						
299						377							455						
300						378							456						
301	-----					379	-----					457	-----						
302						380							458						
303						381							459						
304	-----					382	-----					460	-----						
305						383							461						
306						384							462						
307	-----					385	-----					463	-----						
308						386							464						
309						387							465						
310	-----					388	-----					466	-----						

467						545							623							
468						546							624							
469	-----						547	-----						625	-----					
470						548							626							
471						549							627							
472	-----						550	-----						628	-----					
473						551							629							
474						552							630							
475	-----						553	-----						631	-----					
476						554							632							
477						555							633							
478	-----						556	-----						634	-----					
479						557							635							
480						558							636							
481	-----						559	-----						637	-----					
482						560							638							
483						561							639							
484	-----						562	-----						640	-----					
485						563							641							
486						564							642							
487	-----						565	-----						643	-----					
488						566							644							
489						567							645							
490	-----						568	-----						646	-----					
491						569							647							
492						570							648							
493	-----						571	-----						649	-----					
494						572							650							
495						573							651							
496	-----						574	-----						652	-----					
497						575							653							
498						576							654							
499	-----						577	-----						655	-----					
500						578							656							
501						579							657							
502	-----						580	-----						658	-----					
503						581							659							
504						582							660							
505	-----						583	-----						661	-----					
506						584							662							
507						585							663							
508	-----						586	-----						664	-----					
509						587							665							
510						588							666							
511	-----						589	-----						667	-----					
512						590							668							
513						591							669							
514	-----						592	-----						670	-----					
515						593							671							
516						594							672							
517	-----						595	-----						673	-----					
518						596							674							
519						597							675							
520	-----						598	-----						676	-----					
521						599							677							
522						600							678							
523	-----						601	-----						679	-----					
524						602							680							
525						603							681							
526	-----						604	-----						682	-----					
527						605							683							
528						606							684							
529	-----						607	-----						685	-----					
530						608							686							
531						609							687							
532	-----						610	-----						688	-----					
533						611							689							
534						612							690							
535	-----						613	-----						691	-----					
536						614							692							
537						615							693							
538	-----						616	-----						694	-----					
539						617							695							
540						618							696							
541	-----						619	-----						697	-----					
542						620							698							
543						621							699							
544	-----						622	-----						700	-----					

701						779							857							
702						780							858							
703	-----						781	-----						859	-----					
704						782							860							
705						783							861							
706	-----						784	-----						862	-----					
707						785							863							
708						786							864							
709	-----						787	-----						865	-----					
710						788							866							
711						789							867							
712	-----						790	-----						868	-----					
713						791							869							
714						792							870							
715	-----						793	-----						871	-----					
716						794							872							
717						795							873							
718	-----						796	-----						874	-----					
719						797							875							
720						798							876							
721	-----						799	-----						877	-----					
722						800							878							
723						801							879							
724	-----						802	-----						880	-----					
725						803							881							
726						804							882							
727	-----						805	-----						883	-----					
728						806							884							
729						807							885							
730	-----						808	-----						886	-----					
731						809							887							
732						810							888							
733	-----						811	-----						889	-----					
734						812							890							
735						813							891							
736	-----						814	-----						892	-----					
737						815							893							
738						816							894							
739	-----						817	-----						895	-----					
740						818							896							
741						819							897							
742	-----						820	-----						898	-----					
743						821							899							
744						822							900							
745	-----						823	-----						901	-----					
746						824							902							
747						825							903							
748	-----						826	-----						904	-----					
749						827							905							
750						828							906							
751	-----						829	-----						907	-----					
752						830							908							
753						831							909							
754	-----						832	-----						910	-----					
755						833							911							
756						834							912							
757	-----						835	-----						913	-----					
758						836							914							
759						837							915							
760	-----						838	-----						916	-----					
761						839							917							
762						840							918							
763	-----						841	-----						919	-----					
764						842							920							
765						843							921							
766	-----						844	-----						922	-----					
767						845							923							
768						846							924							
769	-----						847	-----						925	-----					
770						848							926							
771						849							927							
772	-----						850	-----						928	-----					
773						851							929							
774						852							930							
775	-----						853	-----						931	-----					
776						854							932							
777						855							933							
778	-----						856	-----						934	-----					



Jc11	FJCU																24												
935									1013									1091											
936									1014									1092											
937	-----									1015	-----									1093	-----								
938									1016									1094											
939									1017									1095											
940	-----									1018	-----									1096	-----								
941									1019									1097											
942									1020									1098											
943	-----									1021	-----									1099	-----								
944									1022									1100											
945									1023									1101											
946	-----									1024	-----									1102	-----								
947									1025									1103											
948									1026									1104											
949	-----									1027	-----									1105	-----								
950									1028									1106											
951									1029									1107											
952	-----									1030	-----									1108	-----								
953									1031									1109											
954									1032									1110											
955	-----									1033	-----									1111	-----								
956									1034									1112											
957									1035									1113											
958	-----									1036	-----									1114	-----								
959									1037									1115											
960									1038									1116											
961	-----									1039	-----									1117	-----								
962									1040									1118											
963									1041									1119											
964	-----									1042	-----									1120	-----								
965									1043									1121											
966									1044									1122											
967	-----									1045	-----									1123	-----								
968									1046									1124											
969									1047									1125											
970	-----									1048	-----									1126	-----								
971									1049									1127											
972									1050									1128											
973	-----									1051	-----									1129	-----								
974									1052									1130											
975									1053									1131											
976	-----									1054	-----									1132	-----								
977									1055									1133											
978									1056									1134											
979	-----									1057	-----									1135	-----								
980									1058									1136											
981									1059									1137											
982	-----									1060	-----									1138	-----								
983									1061									1139											
984									1062									1140											
985	-----									1063	-----									1141	-----								
986									1064									1142											
987									1065									1143											
988	-----									1066	-----									1144	-----								
989									1067									1145											
990									1068									1146											
991	-----									1069	-----									1147	-----								
992									1070									1148											
993									1071									1149											
994	-----									1072	-----									1150	-----								
995									1073									1151											
996									1074									1152											
997	-----									1075	-----									1153	-----								
998									1076									1154											
999									1077									1155											
1000	-----									1078	-----									1156	-----								
1001									1079									1157											
1002									1080									1158											
1003	-----									1081	-----									1159	-----								
1004									1082									1160											
1005									1083									1161											
1006	-----									1084	-----									1162	-----								
1007									1085									1163											
1008									1086									1164											
1009	-----									1087	-----									1165	-----								
1010									1088									1166											
1011									1089									1167											
1012	-----									1090	-----									1168	-----								

Jc11	FJCU										25									
1169						1247						1325								
1170						1248						1326								
1171	-----					1249	-----					1327	-----							
1172						1250						1328								
1173						1251						1329								
1174	-----					1252	-----					1330	-----							
1175						1253						1331								
1176						1254						1332								
1177	-----					1255	-----					1333	-----							
1178						1256						1334								
1179						1257						1335								
1180	-----					1258	-----					1336	-----							
1181						1259						1337								
1182						1260						1338								
1183	-----					1261	-----					1339	-----							
1184						1262						1340								
1185						1263						1341								
1186	-----					1264	-----					1342	-----							
1187						1265						1343								
1188						1266						1344								
1189	-----					1267	-----					1345	-----							
1190						1268						1346								
1191						1269						1347								
1192	-----					1270	-----					1348	-----							
1193						1271						1349								
1194						1272						1350								
1195	-----					1273	-----					1351	-----							
1196						1274						1352								
1197						1275						1353								
1198	-----					1276	-----					1354	-----							
1199						1277						1355								
1200						1278						1356								
1201	-----					1279	-----					1357	-----							
1202						1280						1358								
1203						1281						1359								
1204	-----					1282	-----					1360	-----							
1205						1283						1361								
1206						1284						1362								
1207	-----					1285	-----					1363	-----							
1208						1286						1364								
1209						1287						1365								
1210	-----					1288	-----					1366	-----							
1211						1289						1367								
1212						1290						1368								
1213	-----					1291	-----					1369	-----							
1214						1292						1370								
1215						1293						1371								
1216	-----					1294	-----					1372	-----							
1217						1295						1373								
1218						1296						1374								
1219	-----					1297	-----					1375	-----							
1220						1298						1376								
1221						1299						1377								
1222	-----					1300	-----					1378	-----							
1223						1301						1379								
1224						1302						1380								
1225	-----					1303	-----					1381	-----							
1226						1304						1382								
1227						1305						1383								
1228	-----					1306	-----					1384	-----							
1229						1307						1385								
1230						1308						1386								
1231	-----					1309	-----					1387	-----							
1232						1310						1388								
1233						1311						1389								
1234	-----					1312	-----					1390	-----							
1235						1313						1391								
1236						1314						1392								
1237	-----					1315	-----					1393	-----							
1238						1316						1394								
1239						1317						1395								
1240	-----					1318	-----					1396	-----							
1241						1319						1397								
1242						1320						1398								
1243	-----					1321	-----					1399	-----							
1244						1322						1400								
1245						1323						1401								
1246	-----					1324	-----					1402	-----							