

## Contents

1 字串	1
1.1 最長迴文子字串	1
1.2 KMP	1
2 math	1
2.1 SG	1
2.2 質數與因數	1
2.3 歐拉函數	2
2.4 大步小步	2
3 algorithm	2
3.1 三分搜	2
3.2 差分	2
3.3 greedy	3
3.4 dinic	4
3.5 SCC Tarjan	4
3.6 ArticulationPoints Tarjan	4
3.7 最小樹狀圖	5
3.8 JosephusProblem	6
3.9 KM	6
3.10 LCA 倍增法	6
3.11 MCMF	7
3.12 Dancing Links	8
4 DataStructure	8
4.1 線段樹 1D	8
4.2 線段樹 2D	8
4.3 權值線段樹	9
4.4 Trie	10
4.5 單調隊列	10
5 geometry	10
5.1 intersection	10
5.2 半平面相交	10
5.3 凸包	11
6 DP	12
6.1 抽屜	12
6.2 Deque 最大差距	12
6.3 LCS 和 LIS	12
6.4 RangeDP	12
6.5 stringDP	12
6.6 TreeDP 有幾個 path 長度為 k	12
6.7 TreeDP reroot	13
6.8 WeightedLIS	13

## 1 字串

### 1.1 最長迴文子字串

```

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

```

```

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

```

### 1.2 KMP

```

1 #define maxn 1000005
2 int nextArr[maxn];
3 void getNextArr(const string& str)
4 {
5     nextArr[0] = 0;
6     int prefixLen = 0;
7     for (int i = 1; i <
8         str.size(); ++i) {
9         prefixLen = nextArr[i - 1];
10        //如果不一樣就在之前算過的pre
11        while (prefixLen > 0 &&
12            str[prefixLen] !=
13            str[i])
14            prefixLen =
15                nextArr[prefixLen
16                    - 1];
17        //一樣就繼承之前的前後綴長度+1
18        if (str[prefixLen] ==
19            str[i])
20            ++prefixLen;
21        nextArr[i] = prefixLen;
22 }
23 for (int i = 0; i < str.size()
24     - 1; ++i) {
25     vis[nextArr[i]] = true;
26 }
27 }

```

## 2 math

### 2.1 SG

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

### 2.2 質數與因數

```

1 歐拉篩 O(n)
2 #define MAXN 47000
3 //sqrt(2^31)=46,340...
4 bool isPrime[MAXN];
5 int prime[MAXN];
6 int primeSize=0;
7 void getPrimes(){
8     memset(isPrime,true,sizeof(isPrime));
9     isPrime[0]=isPrime[1]=false;
10    for(int i=2;i<MAXN;i++){
11        if(isPrime[i])
12            prime[primeSize++]=i;
13        for(int
14            j=0;j<primeSize&&i*prime[j]
15            <MAXN;j++){
16            isPrime[i*prime[j]]=false;
17            if(i%prime[j]==0)
18                break;
19        }
20    }
21 }
22
23 最大公因數 O(log(min(a,b)))
24 int GCD(int a,int b){
25     if(b==0) return a;
26     return GCD(b,a%b);
27 }

```

### 質因數分解

```

24 void primeFactorization(int n){
25     for(int
26         i=0;i<(int)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 }

```

### 擴展歐幾里得算法

//ax+by=GCD(a,b)

```

36 int ext_euc(int a,int b,int &x,int
37     &y){
38     if(b==0){
39         x=1,y=0;
40         return a;
41     }
42     int d=ext_euc(b,a%b,y,x);
43     y-=a/b*x;
44     return d;
45 }
46
47 int main(){
48     int a,b,x,y;
49     cin>>a>>b;
50     ext_euc(a,b,x,y);
51     cout<<x<<' '<<y<<endl;
52     return 0;
53 }

```

### 歌德巴赫猜想

solution : 把偶數  $N$  ( $6 \leq N \leq 10^6$ ) 寫成兩個質數的和。

```

61 #define N 2000000
62 int ox[N],p[N],pr;
63 void PrimeTable(){
64     ox[0]=ox[1]=1;
65     pr=0;
66     for(int i=2;i<N;i++){
67         if(!ox[i]) p[pr++]=i;
68         for(int
69             j=0;i*p[j]<N&&j<pr;j++){
70             ox[i*p[j]]=1;
71         }
72     }
73
74 int main(){
75     PrimeTable();
76     int n;
77     while(cin>>n,n){
78         int x;
79         for(x=1;x+=2)
80             if(!ox[x]&&!ox[n-x])
81                 break;
82         printf("%d = %d +
83             %d\n",n,x,n-x);
84     }
85 }

```

problem : 給定整數  $N$ ，求  $N$

最少可以拆成多少個質數的和。

- 85 如果  $N$  是質數，則答案為 1。
- 86 如果  $N$  是偶數 (不包含 2)，則答案為 2 (強歌德巴赫猜想)。
- 87 如果  $N$  是奇數且  $N-2$  是質數，則答案為 2 (2+質數)。
- 88 其他狀況答案為 3 (弱歌德巴赫猜想)。

```

89 bool isPrime(int n){
90

```

```

91     for(int i=2;i<n;++i){
92         if(i*i>n) return true;
93         if(n%i==0) return false;
94     }
95     return true;
96 }
97
98 int main(){
99     int n;
100    cin>>n;
101    if(isPrime(n)) cout<<"1\n";
102    else if(n%2==0||isPrime(n-2))
103        cout<<"2\n";
104    else cout<<"3\n";

```

## 2.3 歐拉函數

```

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

```

## 2.4 大步小步

```

1  題意
2  給定 B,N,P，求出 L 滿足 B^L ≡ N(mod P)。
3
4  題解
5  餘數的循環節長度必定為 P
   的因數，因此
   B^0, B^P, B^1, B^(P+1), ...，
6  也就是說如果有解則
   L<N，枚舉 0,1,2,L-1
   能得到結果，但會超時。
7  將 L 拆成 mx+y，只要分別枚舉 x,y
   就能得到答案，
8  設 m=√P 能保證最多枚舉 2√P 次。
9  B^(mx+y) ≡ N(mod P)
10 B^y ≡ N(B^(-m))^x (mod P)
11 先求出 B^0, B^1, B^2, ..., B^(m-1)，
12 再枚舉 N(B^(-m)), N(B^(-m))^2, ...
   查看是否有對應的 B^y。
13 這種算法稱為大步小步演算法，
14 大步指的是枚舉 x（一次跨 m 步），
15 小步指的是枚舉 y（一次跨 1 步）。
16 複雜度分析
17 利用 map/unorder_map 存放
   B^0, B^1, B^2, ..., B^(m-1)，
18 枚舉 x 查詢 map/unorder_map
   是否有對應的 B^y，
19 存放和查詢最多 2√P
   次，時間複雜度為
   O(√P log √P) / O(√P)。
20
21 using LL = long long;
22 LL B, N, P;
23 LL fpow(LL a, LL b, LL c){
24     LL res=1;
25     for(;b>=1;){
26         if(b&1)
27             res=(res*a)%c;
28         a=(a*a)%c;

```

```

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

```

## 3 algorithm

««« HEAD ===== »»» cc6ba60f318928240554b362b91a42dc337ff3d4

### 3.1 三分搜

```

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

```

```

24     Point operator/(const double
25         &d) const{
26         return Point(x/d, y/d, z/d);
27     }
28     double dist(const Point &rhs)
29         const{
30         double res = 0;
31         res+=(x-rhs.x)*(x-rhs.x);
32         res+=(y-rhs.y)*(y-rhs.y);
33         res+=(z-rhs.z)*(z-rhs.z);
34         return res;
35     }
36 };
37 int main(){
38     IOS; //輸入優化
39     int T;
40     cin>>T;
41     for(int ti=1; ti<=T; ++ti){
42         double time;
43         Point x1, y1, d1, x2, y2, d2;
44         cin>>time>>x1>>y1>>x2>>y2;
45         d1=(y1-x1)/time;
46         d2=(y2-x2)/time;
47         double
48             L=0, R=1e8, m1, m2, f1, f2;
49         double ans = x1.dist(x2);
50         while(abs(L-R)>1e-10){
51             m1=(L+R)/2;
52             m2=(m1+R)/2;
53             f1=((d1*m1)+x1).dist((d2*m1)+x2);
54             f2=((d1*m2)+x1).dist((d2*m2)+x2);
55             ans =
56                 min(ans, min(f1, f2));
57             if(f1<f2) R=m2;
58             else L=m1;
59         }
60         cout<<"Case "<<ti<<": ";
61         cout<<fixed<<setprecision(4)<<sqrt(a
62         )
63     }
64 }

```

### 3.2 差分

```

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

```

## 3.3 greedy

```

貪心演算法的核心為，
採取在目前狀態下最好或最佳（即最有利）
貪心演算法雖然能獲得當前最佳解，
但不保證能獲得最後（全域）最佳解，
提出想法後可以先試圖尋找有沒有能推翻的
確認無誤再實作。
刪除數字問題
//problem
給定一個數字 N( $\leq 10^4$ )，需要刪除
K 個數字，
請問刪除 K 個數字後最小的數字為何？
//solution
刪除滿足第 i 位數大於第 i+1
位數的最左邊第 i 位數，
扣除高位數的影響較扣除低位數的大。
//code
int main(){
    string s;
    int k;
    cin>>s>>k;
    for(int i=0;i<k;++i){
        if((int)s.size()==0) break;
        int pos=(int)s.size()-1;
        for(int j=0;j<(int)s.size()-1;++j){
            if(s[j]>s[j+1]){
                pos=j;
                break;
            }
        }
        s.erase(pos,1);
    }
    while((int)s.size()>0&&s[0]=='0')
        s.erase(0,1);
    if((int)s.size())
        cout<<s<<'\n';
    else cout<<0<<'\n';
}
最小區間覆蓋長度
//problem
給定 n 條線段區間為 [Li,Ri]，
請問最少要選幾個區間才能完全覆蓋
[0,S]？
//solution
先將所有區間依照左界由小到大排序，
對於當前區間 [Li,Ri]，要從左界 >Ri
的所有區間中，
找到有著最大的右界的區間，連接當前區間
//problem
長度 n 的直線中有數個加熱器，
在 x 的加熱器可以讓 [x-r,x+r]
內的物品加熱，
問最少要幾個加熱器可以把 [0,n]
的範圍加熱。
//solution
對於最左邊沒加熱的點a，選擇最遠可以加熱
更新已加熱範圍，重複上述動作繼續尋找
//code
int main(){
    int n, r;
    int a[1005];
    cin>>n>>r;
    for(int i=1;i<=n;++i)
        cin>>a[i];
    int i=1,ans=0;
    while(i<=n){
        int
            R=min(i+r-1,n),L=max(i-r,0);
        int nextR=-1;
        for(int j=R;j>=L;--j){
            if(a[j]){
                nextR=j;
                break;
            }
        }
        i=nextR+1;
        ans++;
    }
    cout<<ans<<'\n';
}

```

```

}
}
if(nextR==-1){
    ans=-1;
    break;
}
++ans;
i=nextR+r;
}
cout<<ans<<'\n';
}
最多不重疊區間
//problem
給你 n 條線段區間為 [Li,Ri]，
請問最多可以選擇幾條不重疊的線段（頭尾互不重疊）
//solution
依照右界由小到大排序，
每次取到一個不重疊的線段，答案 +1。
//code
struct Line{
    int L,R;
    bool operator<(const Line
        &rhs)const{
        return R<rhs.R;
    }
};
int main(){
    int t;
    cin>>t;
    Line a[30];
    while(t--){
        int n=0;
        while(cin>>a[n].L>>a[n].R,a[n].L<=a[n].R){
            ++n;
        }
        sort(a,a+n);
        int ans=1,R=a[0].R;
        for(int i=1;i<n;i++){
            if(a[i].L>R){
                ++ans;
                R=a[i].R;
            }
        }
        cout<<ans<<'\n';
    }
}
最小化最大延遲問題
//problem
給定 N
項工作，每項工作的需要處理時長為
Ti，
期限是 Di，第 i 項工作延遲的時間為
Li=max(0,Fi-Di)，
原本Fi為第 i 項工作的完成時間，
求一種工作排序使 maxLi 最小。
//solution
按照到期時間從早到晚處理。
//code
struct Work{
    int t, d;
    bool operator<(const Work
        &rhs)const{
        return d<rhs.d;
    }
};
int main(){
    int n;
    Work a[10000];
    cin>>n;
    for(int i=0;i<n;++i)
        cin>>a[i].t>>a[i].d;
    sort(a,a+n);
    int maxL=0,sumT=0;
    for(int i=0;i<n;++i){
        sumT+=a[i].t;
        maxL=max(maxL,sumT-a[i].d);
    }
    cout<<maxL<<'\n';
}

```

```

}
最少延遲數量問題
//problem
給定 N
項工作，每個工作的需要處理時長為
Ti，
期限是
Di，求一種工作排序使得逾期工作數量最小。
//solution
期限越早到期的工作越先做。將工作依照到期時間
依序放入工作列表中，如果發現有工作預期，
就從目前選擇的工作中，移除耗時最長的工作。
上述方法為 Moore-Hodgson's
Algorithm。
//problem
給定烏龜的重量和可承受重量，問最多可以疊幾隻
//solution
和最少延遲數量問題是相同的問題，只要將題敘做
工作處理時長 → 烏龜重量
工作期限 → 烏龜可承受重量
多少工作不延期 → 可以疊幾隻烏龜
//code
struct Work{
    int t, d;
    bool operator<(const Work
        &rhs)const{
        return d<rhs.d;
    }
};
int main(){
    int n=0;
    Work a[10000];
    priority_queue<int> pq;
    while(cin>>a[n].t>>a[n].d){
        ++n;
    }
    sort(a,a+n);
    int sumT=0,ans=n;
    for(int i=0;i<n;++i){
        pq.push(a[i].t);
        sumT+=a[i].t;
        if(a[i].d<sumT){
            int x=pq.top();
            pq.pop();
            sumT-=x;
            --ans;
        }
    }
    cout<<ans<<'\n';
}
任務調度問題
//problem
給定 N
項工作，每項工作的需要處理時長為
Ti，
期限是 Di，如果第 i
項工作延遲需要受到 pi
單位懲罰，
請問最少會受到多少單位懲罰。
//solution
依照懲罰由大到小排序，
每項工作依序嘗試可不可以放在
Di-Ti+1,Di-Ti,...,1,0，
如果有空閒就放進去，否則延後執行。
//problem
給定 N
項工作，每項工作的需要處理時長為
Ti，
期限是 Di，如果第 i
項工作在期限內完成會獲得 ai
單位獎勵，
請問最多會獲得多少單位獎勵。
//solution
和上題相似，這題變成依照獎勵由大到小排序。
//code

```

```

201 struct Work{
202     int d,p;
203     bool operator< (const Work
204         &rhs) const{
205         return p>rhs.p;
206     };
207 int main(){
208     int n;
209     Work a[100005];
210     bitset<100005> ok;
211     while(cin>>n){
212         ok.reset();
213         for(int i=0;i<n;++i)
214             cin>>a[i].d>>a[i].p;
215         sort(a,a+n);
216         int ans=0;
217         for(int i=0;i<n;++i){
218             int j=a[i].d;
219             while(j--){
220                 if(!ok[j]){
221                     ans+=a[i].p;
222                     ok[j]=true;
223                     break;
224                 }
225             }
226             cout<<ans<<'\\n';
227         }
228     }

```

### 3.4 dinic

```

1 const int maxn = 1e5 + 10;
2 const int inf = 0x3f3f3f3f;
3 struct Edge {
4     int s, t, cap, flow;
5 };
6 int n, m, S, T;
7 int level[maxn], dfs_idx[maxn];
8 vector<Edge> E;
9 vector<vector<int>> G;
10 void init() {
11     S = 0;
12     T = n + m;
13     E.clear();
14     G.assign(maxn, vector<int>());
15 }
16 void addEdge(int s, int t, int
17     cap) {
18     E.push_back({s, t, cap, 0});
19     E.push_back({t, s, 0, 0});
20     G[s].push_back(E.size()-2);
21     G[t].push_back(E.size()-1);
22 }
23 bool bfs() {
24     queue<int> q({S});
25     memset(level, -1,
26         sizeof(level));
27     level[S] = 0;
28     while(!q.empty()) {
29         int cur = q.front();
30         q.pop();
31         for(int i : G[cur]) {
32             Edge e = E[i];
33             if(level[e.t]==-1 &&
34                 e.cap>e.flow) {
35                 level[e.t] =
36                     level[e.s] + 1;
37                 q.push(e.t);
38             }
39         }
40     }
41     return level[T];
42 }
43 int dfs(int cur, int lim) {

```

```

40     if(cur==T || lim==0) return
41         lim;
42     int result = 0;
43     for(int& i=dfs_idx[cur];
44         i<G[cur].size() && lim;
45         i++) {
46         Edge& e = E[G[cur][i]];
47         if(level[e.s]+1 !=
48             level[e.t]) continue;
49         int flow = dfs(e.t,
50             min(lim,
51                 e.cap-e.flow));
52         if(flow <= 0) continue;
53         e.flow += flow;
54         result += flow;
55         E[G[cur][i]^1].flow -=
56             flow;
57         lim -= flow;
58     }
59     return result;
60 }
61 int dinic() { // O((V^2)E)
62     int result = 0;
63     while(bfs()) {
64         memset(dfs_idx, 0,
65             sizeof(dfs_idx));
66         result += dfs(S, inf);
67     }
68     return result;
69 }

```

### 3.5 SCC Tarjan

```

1 //單純考 SCC，每個 SCC 中找成本最小的蓋，
2 //注意以下程式有縮點，但沒存起來，存法
3 -> ID[u] = SCCID
4 #define maxn 100005
5 #define MOD 1000000007
6 long long cost[maxn];
7 vector<vector<int>> G;
8 int SCC = 0;
9 stack<int> sk;
10 int dfn[maxn];
11 int low[maxn];
12 bool inStack[maxn];
13 int dfsTime = 1;
14 long long totalCost = 0;
15 long long ways = 1;
16 void dfs(int u) {
17     dfn[u] = low[u] = dfsTime;
18     ++dfsTime;
19     sk.push(u);
20     inStack[u] = true;
21     for (int v: G[u]) {
22         if (dfn[v] == 0) {
23             dfs(v);
24             low[u] = min(low[u],
25                 low[v]);
26         }
27         else if (inStack[v]) {
28             //屬於同個 SCC 且是我的 back
29             //edge
30             low[u] = min(low[u],
31                 dfn[v]);
32         }
33     }
34     //如果是 SCC
35     if (dfn[u] == low[u]) {
36         long long minCost =
37             0x3f3f3f3f;
38         int currWays = 0;
39         ++SCC;
40         while (1) {
41             int v = sk.top();
42             inStack[v] = 0;
43             sk.pop();

```

```

39     if (minCost > cost[v])
40     {
41         minCost = cost[v];
42         currWays = 1;
43     }
44     else if (minCost ==
45         cost[v]) {
46         ++currWays;
47     }
48     if (v == u)
49         break;
50 }
51 totalCost += minCost;
52 ways = (ways * currWays) %
53     MOD;
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",
73         totalCost, ways % MOD);
74     return 0;
75 }

```

### 3.6 ArticulationPoints Tarjan

```

1 vector<vector<int>> G;
2 int N, timer;
3 bool visited[105];
4 int dfn[105]; // 第一次 visit 的時間
5 int low[105];
6 //
7 // 最小能回到的父節點 (不能是自己的 parent) 的
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],
20                 low[v]);
21             if (parent != -1 &&
22                 low[v] >= dfn[u])
23                 isCut = true;
24         }
25         else if (v != parent)
26             low[u] = min(low[u],
27                 dfn[v]);
28     }
29     //If u is root of DFS
30     //tree->有兩個以上的 children
31     if (parent == -1 && child >= 2)
32         isCut = true;
33     if (isCut) ++res;

```

```

29 }
30 int main() {
31     char input[105];
32     char* token;
33     while (scanf("%d", &N) != EOF
34             && N) {
35         G.assign(105,
36                 vector<int>());
37         memset(visited, false,
38                 sizeof(visited));
39         memset(low, 0,
40                 sizeof(low));
41         memset(dfn, 0,
42                 sizeof(visited));
43         timer = 0;
44         res = 0;
45         getchar(); // for \n
46         while (fgets(input, 105,
47                     stdin)) {
48             if (input[0] == '0')
49                 break;
50             int size =
51                 strlen(input);
52             input[size - 1] = '\0';
53             --size;
54             token = strtok(input,
55                             " ");
56             int u = atoi(token);
57             int v;
58             while (token =
59                     strtok(NULL, " "))
60                 {
61                     v = atoi(token);
62                     G[u].emplace_back(v);
63                     G[v].emplace_back(u);
64                 }
65             tarjan(1, -1);
66             printf("%d\n", res);
67         }
68     }
69     return 0;
70 }

```

### 3.7 最小樹狀圖

```

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

```

```

31         inEdge[e.t] =
32             e.cost;
33         pre[e.t] = e.s;
34     }
35     for(int i=0; i<N; i++) {
36         if(i!=root &&
37             inEdge[i]==inf)
38             return
39             -1; //除了 root
40             還有點沒有 in
41             edge
42     }
43     int seq = inEdge[root] = 0;
44     memset(idx, -1,
45             sizeof(idx));
46     memset(vis, -1,
47             sizeof(vis));
48     // 找所有的
49     // cycle，一起編號為 seq
50     for(int i=0; i<N; i++) {
51         result += inEdge[i];
52         int cur = i;
53         while(vis[cur]!=i &&
54             idx[cur]==-1) {
55             if(cur == root)
56                 break;
57             vis[cur] = i;
58             cur = pre[cur];
59         }
60         if(cur!=root &&
61             idx[cur]==-1) {
62             for(int
63                 j=pre[cur];
64                 j!=cur;
65                 j=pre[j])
66                 idx[j] = seq;
67             idx[cur] = seq++;
68         }
69     }
70     if(seq == 0) return
71     result; // 沒有 cycle
72     for(int i=0; i<N; i++)
73         // 沒有被縮點的點
74         if(idx[i] == -1)
75             idx[i] = seq++;
76     // 縮點並重新編號
77     for(Edge& e : edges) {
78         if(idx[e.s] !=
79             idx[e.t])
80             e.cost -=
81             inEdge[e.t];
82         e.s = idx[e.s];
83         e.t = idx[e.t];
84     }
85     N = seq;
86     root = idx[root];
87 }
88 }
89 =====
90 O(m+nlog
91     n)時間內解決最小樹形圖問題的演算
92 27
93 typedef long long ll;
94 #define maxn 102
95 #define INF 0x3f3f3f3f
96 struct UnionFind {
97     int fa[maxn << 1];
98     UnionFind() { memset(fa, 0,
99         sizeof(fa)); }
100     void clear(int n) {
101         memset(fa + 1, 0, sizeof(int)
102             * n);
103     }
104     int find(int x) {
105         return fa[x] ? fa[x] =
106             find(fa[x]) : x;
107     }
108 }

```

```

75 int operator()(int x) { return
76     find(x); }
77 };
78 struct Edge {
79     int u, v, w, w0;
80 };
81 struct Heap {
82     Edge *e;
83     int rk, constant;
84     Heap *lch, *rch;
85     Heap(Edge *_e):
86         e(_e),rk(1),constant(0),lch(NULL),rch(NULL)
87     {}
88     void push() {
89         if (lch) lch->constant +=
90             constant;
91         if (rch) rch->constant +=
92             constant;
93         e->w += constant;
94         constant = 0;
95     }
96 };
97 Heap *merge(Heap *x, Heap *y) {
98     if (!x) return y;
99     if (!y) return x;
100     if (x->e->w + x->constant >
101         y->e->w + y->constant)
102         swap(x, y);
103     x->push();
104     x->rch = merge(x->rch, y);
105     if (!x->lch || x->lch->rk <
106         x->rch->rk)
107         swap(x->lch, x->rch);
108     if (x->rch)
109         x->rk = x->rch->rk + 1;
110     else
111         x->rk = 1;
112     return x;
113 }
114 Edge *extract(Heap *&x) {
115     Edge *r = x->e;
116     x->push();
117     x = merge(x->lch, x->rch);
118     return r;
119 }
120 vector<Edge> in[maxn];
121 int n, m, fa[maxn << 1], nxt[maxn
122     << 1];
123 Edge *ed[maxn << 1];
124 Heap *Q[maxn << 1];
125 UnionFind id;
126 void contract() {
127     bool mark[maxn << 1];
128     //將圖上的每一個節點與其相連的那些節點進行
129     // 縮點
130     for (int i = 1; i <= n; i++) {
131         queue<Heap *> q;
132         for (int j = 0; j <
133             in[i].size(); j++)
134             q.push(new Heap(&in[i][j]));
135         while (q.size() > 1) {
136             Heap *u = q.front();
137             q.pop();
138             Heap *v = q.front();
139             q.pop();
140             q.push(merge(u, v));
141         }
142         Q[i] = q.front();
143     }
144     mark[1] = true;
145     for(int
146         a=1,b=1,p;Q[a];b=a,mark[b]=true){
147         //尋找最小入邊以及其端點，保證無環
148         do {
149             ed[a] = extract(Q[a]);
150             a = id[ed[a]->u];
151         } while (a == b && Q[a]);
152         if (a == b) break;
153         if (!mark[a]) continue;

```



```

144 //對發現的環進行收縮，以及環內的
145 //總權值更新
146 for (a = b, n++; a != n; a =
    p) {
147     id.fa[a] = fa[a] = n;
148     if (Q[a]) Q[a]->constant -=
        ed[a]->w;
149     Q[n] = merge(Q[n], Q[a]);
150     p = id[ed[a]->u];
151     nxt[p == n ? b : p] = a;
152 }
153 }
154 }
155 ll expand(int x, int r);
156 ll expand_iter(int x) {
157     ll r = 0;
158     for(int u=nxt[x]; u!=x; u=nxt[u]){
159         if (ed[u]->w0 >= INF)
160             return INF;
161         else
162             r+=expand(ed[u]->v,u)+ed[u]->w0;
163     }
164     return r;
165 }
166 ll expand(int x, int t) {
167     ll r = 0;
168     for (; x != t; x = fa[x]) {
169         r += expand_iter(x);
170         if (r >= INF) return INF;
171     }
172     return r;
173 }
174 void link(int u, int v, int w) {
175     in[v].push_back({u, v, w, w});
176 }
177 int main() {
178     int rt;
179     scanf("%d %d %d", &n, &m, &rt);
180     for (int i = 0; i < m; i++) {
181         int u, v, w;
182         scanf("%d %d %d", &u, &v, &w);
183         link(u, v, w);
184     }
185     //保證強連通
186     for (int i = 1; i <= n; i++)
187         link(i > 1 ? i - 1 : n, i,
            INF);
188     contract();
189     ll ans = expand(rt, n);
190     if (ans >= INF)
191         puts("-1");
192     else
193         printf("%lld\n", ans);
194     return 0;
195 }

```

### 3.8 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0
4 int getWinner(int n, int k) {
5     int winner = 0;
6     for (int i = 1; i <= n; ++i)
7         winner = (winner + k) % i;
8     return winner;
9 }
10 int main() {
11     int n;
12     while (scanf("%d", &n) != EOF
        && n){
13         --n;
14         for (int k = 1; k <= n;
            ++k){

```

```

15         if (getWinner(n, k) ==
            11){
16             printf("%d\n", k);
17             break;
18         }
19     }
20 }
21 return 0;
22 }

```

### 3.9 KM

```

1 /*題意：
2     給定一個W矩陣，現在分成row、column
3     W[i][j]=k即代表column[i] +
        row[j]要>=k
4     求row[]與
        column[]的所有值在滿足矩陣W的
5     row[] +
        column[]所有元素相加起來要最小
6     利用KM求二分圖最大權匹配
7     Lx -> vertex labeling of X
8     Ly -> vertex labeling of y
9     一開始Lx[i] = max(W[i][j]), Ly
        = 0
10    Lx[i] + Ly[j] >= W[i][j]
11    要最小化全部的(Lx[i] +
        Ly[j])加總
12    不斷的調整vertex
        labeling去找到一條交錯邊皆滿足
13    + Ly[j] == W[i][j]的增廣路
14    最後會得到正確的二分圖完美匹配中的
        意義是將最大化所有匹配邊權重的和
15 #define maxn 505
16 int W[maxn][maxn];
17 int Lx[maxn], Ly[maxn];
18 bool S[maxn], T[maxn];
19 //L[i] = j -> S_i配給T_j, -1 for
        還沒匹配
20 int L[maxn];
21 int n;
22 bool match(int i) {
23     S[i] = true;
24     for (int j = 0; j < n; ++j) {
25         // KM重點
26         // Lx + Ly >=
            selected_edge(x, y)
27         // 要想辦法降低Lx + Ly
28         // 所以選Lx + Ly ==
            selected_edge(x, y)
29         if (Lx[i] + Ly[j] ==
            W[i][j] && !T[j]) {
30             T[j] = true;
31             if ((L[j] == -1) ||
                match(L[j])) {
32                 L[j] = i;
33                 return true;
34             }
35         }
36     }
37     return false;
38 }
39 // 修改二分圖上的交錯路徑上點的權重
40 // 此舉是在通過調整vertex
        labeling看看能不能產生出新的增廣路
41 //
        在這裡優先從最小的diff調調看，才
42 void update()
43 {
44     int diff = 0x3f3f3f3f;
45     for (int i = 0; i < n; ++i) {
46         if (S[i]) {
47             for (int j = 0; j < n;
                ++j) {

```

```

48                 diff =
                    min(diff,
                        Lx[i] +
                        Ly[j] -
                        W[i][j]);
49             }
50         }
51     }
52     for (int i = 0; i < n; ++i) {
53         if (S[i]) Lx[i] -= diff;
54         if (T[i]) Ly[i] += diff;
55     }
56 }
57 void KM()
58 {
59     for (int i = 0; i < n; ++i) {
60         L[i] = -1;
61         Lx[i] = Ly[i] = 0;
62         for (int j = 0; j < n; ++j)
63             Lx[i] = max(Lx[i],
                W[i][j]);
64     }
65     for (int i = 0; i < n; ++i) {
66         while(1) {
67             memset(S, false,
                sizeof(S));
68             memset(T, false,
                sizeof(T));
69             if (match(i))
70                 break;
71             else
72                 update();
73             //去調整vertex
                labeling以增加增廣路徑
74         }
75     }
76 }
77 int main() {
78     while (scanf("%d", &n) != EOF)
79     {
80         for (int i = 0; i < n; ++i)
81             for (int j = 0; j < n;
                ++j)
82                 scanf("%d",
                    &W[i][j]);
83         KM();
84         int res = 0;
85         for (int i = 0; i < n;
            ++i) {
86             if (i != 0)
87                 printf("%d",
                    Lx[i]);
88             else
89                 printf("%d",
                    Lx[i]);
90             res += Lx[i];
91         }
92         puts("");
93         for (int i = 0; i < n;
            ++i) {
94             if (i != 0)
95                 printf("%d",
                    Ly[i]);
96             else
97                 printf("%d",
                    Ly[i]);
98             res += Ly[i];
99         }
100         puts("");
101         printf("%d\n", res);
102     }
103     return 0;
104 }

```

### 3.10 LCA 倍增法

```

1 //倍增法預處理  $O(n \log n)$ ，查詢  $O(\log n)$ ，58
2 #define maxn 100005
3 struct Edge {
4     int u, v, w;
5 };
6 vector<vector<Edge>> G; // tree
7 int fa[maxn][31]; // fa[u][i] ->
8     u 的第  $2^i$  個祖先
9 long long dis[maxn][31];
10 int dep[maxn]; // 深度
11 void dfs(int u, int p) { // 預處理 fa
12     fa[u][0] = p; // 因為 u 的第  $2^0 = 1$ 
13     的祖先就是 p
14     dep[u] = dep[p] + 1;
15     // 第  $2^i$  的祖先是 (第  $2^{i-1}$ 
16     個祖先) 的第  $2^{i-1}$ 
17     的祖先
18     // ex: 第 8 個祖先是
19     (第 4 個祖先) 的第 4 個祖先
20     for (int i = 1; i < 31; ++i) {
21         fa[u][i] = fa[fa[u][i-1]][i-1];
22         dis[u][i] = dis[fa[u][i-1]][i-1] + dis[u][i-1];
23     }
24     // 遍歷子節點
25     for (Edge& edge: G[u]) {
26         if (edge.v == p) continue;
27         dis[edge.v][0] = edge.w;
28         dfs(edge.v, u);
29     }
30 }
31 long long lca(int x, int y)
32 { // 此函數是找 lca 同時計算 x、y 的距離
33   -> dis(x, lca) + dis(lca, y)
34   // 讓 y 比 x 深
35   if (dep[x] > dep[y]) swap(x, y);
36   int deltaDep = dep[y] - dep[x];
37   long long res = 0;
38   // 讓 y 與 x 在同一個深度
39   for (int i = 0; deltaDep != 0;
40         ++i, deltaDep >>= 1)
41       if (deltaDep & 1)
42         res += dis[y][i], y = fa[y][i];
43   if (y == x) // x = y ->
44     x、y 彼此是彼此的祖先
45     return res;
46   // 往上找，一起跳，但 x、y 不能重疊
47   for (int i = 30; i >= 0 && y
48         != x; --i) {
49       if (fa[x][i] != fa[y][i]) {
50         res += dis[x][i] +
51             dis[y][i];
52         x = fa[x][i];
53         y = fa[y][i];
54     }
55 }
56 // 最後發現不能跳了，此時 x 的第  $2^0 = 1$ 
57 個祖先 (或說 y 的第  $2^0 = 1$  的祖先) 即為 x、y 的 lca
58 res += dis[x][0] + dis[y][0];
59 return res;
60 }
61 int main() {
62     int n, q;
63     while (~scanf("%d", &n) && n) {
64         int v, w;
65         G.assign(n + 5, vector<Edge>());
66         for (int i = 1; i <= n - 1; ++i) {
67             scanf("%d %d", &v, &w);
68         }
69     }
70 }

```

### 3.11 MCMF

```

1 #define maxn 225
2 #define INF 0x3f3f3f3f
3 struct Edge {
4     int u, v, cap, flow, cost;
5 };
6 // node size, edge size, source, target
7 int n, m, s, t;
8 vector<vector<int>> G;
9 vector<Edge> edges;
10 // SPFA 用
11 bool inqueue[maxn];
12 // SPFA 用的 dis[]
13 long long dis[maxn];
14 // maxFlow 一路扣回去時要知道 parent
15 // 在題因為 G[][] 中存的是 edgeIndex in edges[]
16 // 所以 parent 存的也是對應 edges[] 中的
17 int parent[maxn];
18 // maxFlow 時需要紀錄到 node u 時的 bottleneck
19 // 同時也代表著 u 該次流出去的量
20 long long outFlow[maxn];
21 void addEdge(int u, int v, int cap, int cost) {
22     edges.emplace_back(Edge{u, v, cap, 0, cost});
23     edges.emplace_back(Edge{v, u, 0, 0, -cost});
24     m = edges.size();
25     G[u].emplace_back(m - 2);
26     G[v].emplace_back(m - 1);
27 }
28 // 一邊求最短路的同時一邊 MaxFlow
29 bool SPFA(long long& maxFlow, long long& minCost) {
30     // memset(outFlow, 0x3f, sizeof(outFlow));
31     memset(dis, 0x3f, sizeof(dis));
32     memset(inqueue, false, sizeof(inqueue));
33     queue<int> q;
34     q.push(s);
35     dis[s] = 0;
36     inqueue[s] = true;
37     outFlow[s] = INF;
38     while (!q.empty()) {
39         int u = q.front();
40         q.pop();
41         inqueue[u] = false;
42         for (const int edgeIndex: G[u]) {
43             const Edge& edge = edges[edgeIndex];

```

```

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

```

```

    &inventoryMonth);
    addEdge(s, i,
        produceMax,
        produceCost);
    addEdge(M + i, t,
        sellMax,
        -sellPrice);
    for (int j = 0; j <=
        inventoryMonth;
        ++j) {
        if (i + j <= M)
            addEdge(i, M +
                i + j,
                INF, I *
                j);
    }
    printf("Case %d: %lld\n",
        Case, -MCMF());
}
return 0;
}

```

### 3.12 Dancing Links

```

1 struct DLX {
2     int seq, resSize;
3     int col[maxn], row[maxn];
4     int U[maxn], D[maxn], R[maxn],
      L[maxn];
5     int rowHead[maxn],
      colSize[maxn];
6     int result[maxn];
7     DLX(int r, int c) {
8         for(int i=0; i<=c; i++) {
9             L[i] = i-1, R[i] = i+1;
10            U[i] = D[i] = i;
11        }
12        L[R[seq]=0]=c;
13        resSize = -1;
14        memset(rowHead, 0,
            sizeof(rowHead));
15        memset(colSize, 0,
            sizeof(colSize));
16    }
17    void insert(int r, int c) {
18        row[++seq]=r, col[seq]=c,
            ++colSize[c];
19        U[seq]=c, D[seq]=D[c],
            U[D[c]]=seq, D[c]=seq;
20        if(rowHead[r]) {
21            L[seq]=rowHead[r],
                R[seq]=R[rowHead[r]];
22            L[R[rowHead[r]]]=seq,
                R[rowHead[r]]=seq;
23        } else {
24            rowHead[r] = L[seq] =
                R[seq] = seq;
25        }
26    }
27    void remove(int c) {
28        L[R[c]] = L[c], R[L[c]] =
            R[c];
29        for(int i=D[c]; i!=c;
            i=D[i]) {
30            for(int j=R[i]; j!=i;
                j=R[j]) {
31                U[D[j]] = U[j];
32                D[U[j]] = D[j];
33                --colSize[col[j]];
34            }
35        }
36    }
37    void recover(int c) {
38        for(int i=U[c]; i!=c;
            i=U[i]) {

```

```

39        for(int j=L[i]; j!=i;
            j=L[j]) {
40            U[D[j]] = D[U[j]]
                = j;
41            ++colSize[col[j]];
42        }
43    }
44    L[R[c]] = R[L[c]] = c;
45    }
46    bool dfs(int idx=0) { //
47        判斷其中一解版
48        if(R[0] == 0) {
49            resSize = idx;
50            return true;
51        }
52        int c = R[0];
53        for(int i=R[0]; i; i=R[i])
54        {
55            if(colSize[i] <
                colSize[c]) c = i;
56        }
57        remove(c);
58        for(int i=D[c]; i!=c;
            i=D[i]) {
59            result[idx] = row[i];
60            for(int j=R[i]; j!=i;
                j=R[j])
61            remove(col[j]);
62            if(dfs(idx+1)) return
                true;
63            for(int j=L[i]; j!=i;
                j=L[j])
64            recover(col[j]);
65        }
66        recover(c);
67        return false;
68    }
69    void dfs(int idx=0) { //
70        判斷最小 dfs depth 版
71        if(R[0] == 0) {
72            resSize = min(resSize,
                idx); // 注意 init 值
73            return;
74        }
75        int c = R[0];
76        for(int i=R[0]; i; i=R[i])
77        {
78            if(colSize[i] <
                colSize[c]) c = i;
79        }
80        remove(c);
81        for(int i=D[c]; i!=c;
            i=D[i]) {
82            for(int j=R[i]; j!=i;
                j=R[j])
83            remove(col[j]);
84            dfs(idx+1);
85            for(int j=L[i]; j!=i;
                j=L[j])
86            recover(col[j]);
87        }
88        recover(c);
89    }

```

## 4 DataStructure

### 4.1 線段樹 1D

```

1 #define MAXN 1000
2 int data[MAXN]; // 原數據
3 int st[4 * MAXN]; // 線段樹
4 int tag[4 * MAXN]; // 懶標
5 inline int pull(int l, int r) {
6     // 隨題目改變 sum、max、min
7     // l、r 是左右樹的 index
8     return st[l] + st[r];
9 }

```

```

10 void build(int l, int r, int i) {
11     // 在 [l,
12     r] 區間建樹，目前根的 index 為 i
13     if (l == r) {
14         st[i] = data[l];
15         return;
16     }
17     int mid = l + ((r - l) >> 1);
18     build(l, mid, i * 2);
19     build(mid + 1, r, i * 2 + 1);
20     st[i] = pull(i * 2, i * 2 + 1);
21 }
22 int query(int ql, int qr, int l,
    int r, int i) {
23     // [ql, qr] 是查詢區間, [l,
24     r] 是當前節點包含的區間
25     if (ql <= l && r <= qr)
26         return st[i];
27     int mid = l + ((r - l) >> 1);
28     if (tag[i]) {
29         // 如果當前懶標有值則更新左右節點
30         st[i * 2] += tag[i] * (mid
            - l + 1);
31         st[i * 2 + 1] += tag[i] *
            (r - mid);
32         tag[i * 2] +=
            tag[i]; // 下傳懶標至左節點
33         tag[i * 2 + 1] +=
            tag[i]; // 下傳懶標至右節點
34         tag[i] = 0;
35     }
36     int sum = 0;
37     if (ql <= mid)
38         sum += query(ql, qr, l,
            mid, i * 2);
39     if (qr > mid)
40         sum += query(ql, qr, mid +
            1, r, i * 2 + 1);
41     return sum;
42 }
43 void update(int ql, int qr, int
    l, int r, int i, int c) {
44     // [ql, qr] 是查詢區間, [l,
45     r] 是當前節點包含的區間
46     // c 是變化量
47     if (ql <= l && r <= qr) {
48         st[i] += (r - l + 1) * c;
49         // 求和，此需乘上區間長度
50         tag[i] += c;
51         return;
52     }
53     int mid = l + ((r - l) >> 1);
54     if (tag[i] && l != r) {
55         // 如果當前懶標有值則更新左右節點
56         st[i * 2] += tag[i] * (mid
            - l + 1);
57         st[i * 2 + 1] += tag[i] *
            (r - mid);
58         tag[i * 2] +=
            tag[i]; // 下傳懶標至左節點
59         tag[i * 2 + 1] +=
            tag[i]; // 下傳懶標至右節點
60         tag[i] = 0;
61     }
62     if (ql <= mid) update(ql, qr,
        l, mid, i * 2, c);
63     if (qr > mid) update(ql, qr,
        mid + 1, r, i * 2 + 1, c);
64     st[i] = pull(i * 2, i * 2 + 1);
65 }
66 // 如果是直接改值而不是加值，query 與 update 中
67 // 改值從 += 改成 =

```

### 4.2 線段樹 2D



```

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

```

```

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

```

```

88     }
89     else {
90         scanf("%d %d %d",
           &xql, &yql,
           &val);
91         modifyX(1, 1, N,
           val, xql, yql);
92     }
93 }
94 return 0;
95 }

```

### 4.3 權值線段樹

```

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

```

```

52 // 防止 m == 0
53 if (m)
54     sort(id + 1, id + m + 1);
55 int stSize = unique(id + 1, id + m + 1) - (id + 1);
56 for (int i = 1; i <= m; ++i) {
57     nums[i] = lower_bound(id + 1, id + stSize + 1, nums[i]) - id;
58 }
59 int addCount = 0;
60 int getCount = 0;
61 int k = 1;
62 while (getCount < n) {
63     if (getArr[getCount] == addCount) {
64         printf("%d\n", query(1, 1, stSize, k));
65         ++k;
66         ++getCount;
67     }
68     else {
69         update(1, 1, stSize, nums[addCount + 1]);
70         ++addCount;
71     }
72 }
73 }
74 return 0;
75 }

```

#### 4.4 Trie

```

1 const int maxn = 300000 + 10;
2 const int mod = 20071027;
3 int dp[maxn];
4 int mp[4000*100 + 10][26];
5 char str[maxn];
6 struct Trie {
7     int seq;
8     int val[maxn];
9     Trie() {
10         seq = 0;
11         memset(val, 0, sizeof(val));
12         memset(mp, 0, sizeof(mp));
13     }
14     void insert(char* s, int len) {
15         int r = 0;
16         for (int i = 0; i < len; i++) {
17             int c = s[i] - 'a';
18             if (!mp[r][c]) mp[r][c] = ++seq;
19             r = mp[r][c];
20         }
21         val[r] = len;
22         return;
23     }
24     int find(int idx, int len) {
25         int result = 0;
26         for (int r = 0; idx < len; idx++) {
27             int c = str[idx] - 'a';
28             if (!(r = mp[r][c])) return result;
29             if (val[r]) result = (result + dp[idx + 1]) % mod;
30         }

```

```

31     }
32     return result;
33 }
34 };
35 int main() {
36     int n, tc = 1;
37     while (~scanf("%s%d", str, &n)) {
38         Trie tr;
39         int len = strlen(str);
40         char word[100+10];
41         memset(dp, 0, sizeof(dp));
42         dp[len] = 1;
43         while (n--) {
44             scanf("%s", word);
45             tr.insert(word, strlen(word));
46         }
47         for (int i = len - 1; i >= 0; i--) {
48             dp[i] = tr.find(i, len);
49             printf("Case %d: %d\n", tc++, dp[0]);
50         }
51         return 0;
52 }
53 /****Input****
54 * abcd
55 * 4
56 * a b cd ab
57 * ****
58 * ****Output****
59 * Case 1: 2
60 * ****

```

#### 4.5 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以淘汰他"
3
4 example
5 給出一個長度為 n 的數組，
6 輸出每 k
7 個連續的數中的最大值和最小值。
8
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14
15 void getmin() {
16     // 得到這個隊列裡的最小值，直接
17     int head = 0, tail = 0;
18     for (int i = 1; i <= n; i++) {
19         while (head <= tail && a[q[tail]] >= a[i]) tail--;
20         q[++tail] = i;
21     }
22     for (int i = k; i <= n; i++) {
23         while (head <= tail && a[q[tail]] >= a[i]) tail--;
24         q[++tail] = i;
25         while (q[head] <= i - k) head++;
26         cout << a[q[head]] << " ";
27     }
28     cout << endl;
29 }
30
31 void getmax() { // 和上面同理
32     int head = 0, tail = 0;
33     for (int i = 1; i <= n; i++) {
34         while (head <= tail && a[q[tail]] <= a[i]) tail--;
35         q[++tail] = i;

```

```

36     }
37     for (int i = k; i <= n; i++) {
38         while (head <= tail && a[q[tail]] <= a[i]) tail--;
39         q[++tail] = i;
40         while (q[head] <= i - k) head++;
41         cout << a[q[head]] << " ";
42     }
43     cout << endl;
44 }
45
46 int main() {
47     cin >> n >> k; // 每 k 個連續的數
48     for (int i = 1; i <= n; i++) cin >> a[i];
49     getmin();
50     getmax();
51     return 0;
52 }

```

### 5 geometry

#### 5.1 intersection

```

1 using LL = long long;
2
3 struct Point2D {
4     LL x, y;
5 };
6
7 struct Line2D {
8     Point2D s, e;
9     LL a, b, c; // L:
10     // ax + by = c
11     Line2D(Point2D s, Point2D e) {
12         s(s), e(e) {
13             a = e.y - s.y;
14             b = s.x - e.x;
15             c = a * s.x + b * s.y;
16         }
17     };
18
19 // 用克拉馬公式求二元一次解
20 Point2D intersection2D(Line2D l1, Line2D l2) {
21     LL D = l1.a * l2.b - l2.a * l1.b;
22     LL Dx = l1.c * l2.b - l2.c * l1.b;
23     LL Dy = l1.a * l2.c - l2.a * l1.c;
24
25     if (D) { // intersection
26         double x = 1.0 * Dx / D;
27         double y = 1.0 * Dy / D;
28     } else { // Parallel lines
29         // Same line
30     }
31 }

```

#### 5.2 半平面相交

```

1 // Q: 給定一張凸包(已排序的點)，
2 // 找出圖中離凸包外最遠的距離
3
4 const int maxn = 100 + 10;
5 const double eps = 1e-7;
6
7 struct Vector {
8     double x, y;
9     Vector(double x = 0.0, double y = 0.0): x(x), y(y) {}
10
11     Vector operator+(Vector v) {

```

```

12     return Vector(x+v.x,
13                  y+v.y);
14 }
15 Vector operator-(Vector v) {
16     return Vector(x-v.x,
17                  y-v.y);
18 }
19 Vector operator*(double val) {
20     return Vector(x*val,
21                  y*val);
22 }
23 double dot(Vector v) { return
24     x*v.x + y*v.y; }
25 double cross(Vector v) {
26     return x*v.y - y*v.x; }
27 double length() { return
28     sqrt(dot(*this)); }
29 Vector unit_normal_vector() {
30     double len = length();
31     return Vector(-y/len,
32                  x/len);
33 }
34 };
35 using Point = Vector;
36 struct Line {
37     Point p;
38     Vector v;
39     double ang;
40     Line(Point p={}, Vector v={}):
41         p(p), v(v) {
42         ang = atan2(v.y, v.x);
43     }
44     bool operator<(const Line& l)
45     const {
46         return ang < l.ang;
47     }
48     Point intersection(Line l) {
49         Vector u = p - l.p;
50         double t = l.v.cross(u) /
51             v.cross(l.v);
52         return p + v*t;
53     }
54 };
55 int n, m;
56 Line narrow[maxn]; //
57     要判斷的直線
58 Point poly[maxn]; //
59     能形成半平面交的凸包邊界點
60 // return true if point p is on
61 // the left of line l
62 bool onLeft(Point p, Line l) {
63     return l.v.cross(p-l.p) > 0;
64 }
65 int halfplaneIntersection() {
66     int l, r;
67     Line L[maxn]; //
68     排序後的向量隊列
69     Point P[maxn]; // s[i]
70     跟 s[i-1] 的交點
71     L[l=r=0] = narrow[0]; //
72     notice: narrow is sorted
73     for(int i=1; i<n; i++) {
74         while(l<r &&
75             !onLeft(P[r-1],
76                 narrow[i])) r--;
77         while(l<r && !onLeft(P[l],
78             narrow[i])) l++;
79         L[++r] = narrow[i];
80         if(l < r) P[r-1] =
81             L[r-1].intersection(L[r]);
82     }
83     int n;
84 }
85 }
86 while(l<r && !onLeft(P[r-1],
87     L[l])) r--;
88 if(r-l <= 1) return 0;
89 P[r] = L[r].intersection(L[l]);
90 int m=0;
91 for(int i=1; i<=r; i++) {
92     poly[m++] = P[i];
93 }
94 return m;
95 }
96 Point pt[maxn];
97 Vector vec[maxn];
98 Vector normal[maxn]; // normal[i] =
99     vec[i] 的單位法向量
100 double bsearch(double l=0.0,
101     double r=1e4) {
102     if(abs(r-l) < eps) return l;
103     double mid = (l + r) / 2;
104     for(int i=0; i<n; i++) {
105         narrow[i] =
106             Line(pt[i]+normal[i]*mid,
107                 vec[i]);
108     }
109     if(halfplaneIntersection())
110         return bsearch(mid, r);
111     else return bsearch(l, mid);
112 }
113 int main() {
114     while(~scanf("%d", &n) && n) {
115         for(int i=0; i<n; i++) {
116             double x, y;
117             scanf("%lf%lf", &x,
118                 &y);
119             pt[i] = {x, y};
120         }
121         for(int i=0; i<n; i++) {
122             vec[i] = pt[(i+1)%n] -
123                 pt[i];
124             normal[i] =
125                 vec[i].unit_normal_vec
126         }
127         printf("%.6lf\n",
128             bsearch());
129     }
130     return 0;
131 }
132 }
133 }
134 bool destroyed[maxn];
135 Point arr[maxn];
136 vector<Point> polygons[maxn];
137 void scanAndSortPoints() {
138     int minX = maxCoordinate, minY
139         = maxCoordinate;
140     for(int i=0; i<n; i++) {
141         int x, y;
142         scanf("%d%d", &x, &y);
143         arr[i] = (Point){x, y};
144         if(y < minY || (y == minY
145             && x < minX)) {
146             // If there are floating
147             // points, use:
148             // if(y<minY ||
149             // (abs(y-minY)<eps &&
150             // x<minX)) {
151                 minX = x, minY = y;
152             }
153     }
154     sort(arr, arr+n, [minX,
155         minY](Point& a, Point& b){
156         double theta1 = atan2(a.y
157             - minY, a.x - minX);
158         double theta2 = atan2(b.y
159             - minY, b.x - minX);
160         return theta1 < theta2;
161     });
162     return;
163 }
164 // returns cross product of u(AB)
165 // x v(AC)
166 int cross(Point& A, Point& B,
167     Point& C) {
168     int u[2] = {B.x - A.x, B.y -
169         A.y};
170     int v[2] = {C.x - A.x, C.y -
171         A.y};
172     return (u[0] * v[1]) - (u[1] *
173         v[0]);
174 }
175 // size of arr = n >= 3
176 // st = the stack using vector, m
177 // = index of the top
178 vector<Point> convex_hull() {
179     vector<Point> st(arr, arr+3);
180     for(int i=3, m=2; i<n; i++,
181         m++) {
182         while(m >= 2) {
183             if(cross(st[m],
184                 st[m-1], arr[i]) <
185                 0)
186                 break;
187             st.pop_back();
188             m--;
189         }
190         st.push_back(arr[i]);
191     }
192     return st;
193 }
194 bool inPolygon(vector<Point>& vec,
195     Point p) {
196     vec.push_back(vec[0]);
197     for(int i=1; i<vec.size();
198         i++) {
199         if(cross(vec[i-1], vec[i],
200             p) < 0) {
201             vec.pop_back();
202             return false;
203         }
204     }
205     vec.pop_back();
206     return true;
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

### 5.3 凸包

```

1 //
2 //
3 // Q: 平面上給定多個區域，由多個座標
4 // 多點(x,y)，判斷有落點的區域(destroy)
5 #include <bits/stdc++.h>
6 using namespace std;
7 const int maxn = 500 + 10;
8 const int maxCoordinate = 500 + 10;
9 struct Point {
10     int x, y;
11 };
12 int n;

```

```

71 }
72
73     1 | x1    x2    x3    x4    x5
74     A = - |    x    x    x    x
75           x ... x |
76           2 | y1    y2    y3    y4    y5
77           yn |
78
79 double
80 calculateArea(vector<Point>&
81 v) {
82     v.push_back(v[0]); //
83     make v[n] = v[0]
84     double result = 0.0;
85     for(int i=1; i<v.size(); i++)
86         result += v[i-1].x*v[i].y
87             - v[i-1].y*v[i].x;
88     v.pop_back();
89     return result / 2.0;
90 }
91
92 int main() {
93     int p = 0;
94     while(~scanf("%d", &n) && (n
95         != -1)) {
96         scanAndSortPoints();
97         polygons[p++] =
98             convex_hull();
99     }
100
101     int x, y;
102     double result = 0.0;
103     while(~scanf("%d%d", &x, &y)) {
104         for(int i=0; i<p; i++) {
105             if(inPolygon(polygons[i],
106                 (Point){x, y}))
107                 destroyed[i] =
108                     true;
109         }
110         for(int i=0; i<p; i++) {
111             if(destroyed[i])
112                 result +=
113                     calculateArea(polygons[i]);
114         }
115         printf("%.2lf\n", result);
116         return 0;
117     }
118 }

```

## 6 DP

### 6.1 抽屜

```

1 long long dp[70][70][2];
2 // 初始條件
3 dp[1][0][0] = dp[1][1][1] = 1;
4 for (int i = 2; i <= 66; ++i){
5     // i個抽屜0個安全且上方0 =
6     // (底下i -
7     // 1個抽屜且1個安全且最上面L)
8     // + (底下n -
9     // 1個抽屜0個安全且最上方為0)
10     dp[i][0][0] = dp[i - 1][1][1]
11     + dp[i - 1][0][0];
12     for (int j = 1; j <= i; ++j) {
13         dp[i][j][0] = dp[i - 1][j]
14         + 1[j][1] + dp[i - 1][j][0];
15         dp[i][j][1] = dp[i - 1][j]
16         - 1[j][1] + dp[i - 1][j]
17         - 1[j][0];
18     }
19 } //答案在 dp[n][s][0] +
20 dp[n][s][1];

```

## 6.2 Deque 最大差距

```

1 /*定義dp[l][r]是1 ~
2     r時與先手最大差異值
3 Deque可以拿頭尾
4 所以轉移式中dp[l][r]與dp[l +
5     1][r]、dp[l][r - 1]有關
6 轉移式:
7 dp[l][r] = max{a[l] - solve(l +
8     1, r), a[r] - solve(l, r -
9     1)}
10 裡面用減的主要是因為求的是相減且會
11 #define maxn 3005
12 bool vis[maxn][maxn];
13 long long dp[maxn][maxn];
14 long long a[maxn];
15 long long solve(int l, int r) {
16     if (l > r)
17         return 0;
18     if (vis[l][r])
19         return dp[l][r];
20     vis[l][r] = true;
21     long long res = a[l] - solve(l
22         + 1, r);
23     res = max(res, a[r] - solve(l,
24         r - 1));
25     return dp[l][r] = res;
26 }
27
28 int main() {
29     ...
30     printf("%lld\n", solve(1, n));
31 }

```

## 6.3 LCS 和 LIS

```

1 //最長共同子序列(LCS)
2 給定兩序列 A,B，求最長的序列 C，
3 C同時為 A,B 的子序列。
4 //最長遞增子序列(LIS)
5 給你一個序列 A，求最長的序列 B，
6 B是一個(非)嚴格遞增序列，且為
7 A的子序列。
8 //LCS 和 LIS 題目轉換
9 LIS 轉成 LCS
10 1. A 為原序列，B=sort(A)
11 2. 對 A,B 做 LCS
12 LCS 轉成 LIS
13 1. A, B 為原本的兩序列
14 2. 最 A
15 序列作編號轉換，將轉換規則套用
16 B
17 3. 對 B 做 LIS
18 4.
19 重複的數字在編號轉換時後要變成
20 越早出現的數字要越小
21 5. 如果有數字在 B 裡面而不在 A
22 裡面，
23 直接忽略這個數字不做轉換即可

```

## 6.4 RangeDP

```

1 //區間dp
2 int dp[55][55]; // dp[i][j] -> [i,
3 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]
9     - cuts[i]
10     if (i == j - 1)
11         return dp[i][j] = 0;
12     int cost = 0x3f3f3f3f;
13     for (int m = i + 1; m < j;
14         ++m) {

```

```

12 //枚舉區間中間切點
13 cost = min(cost, solve(i,
14     m) + solve(m, j) +
15     cuts[j] - cuts[i]);
16 }
17 return dp[i][j] = cost;
18 }
19
20 int main() {
21     int l;
22     int n;
23     while (scanf("%d", &l) != EOF
24         && l){
25         scanf("%d", &n);
26         for (int i = 1; i <= n;
27             ++i)
28             scanf("%d", &cuts[i]);
29         cuts[0] = 0;
30         cuts[n + 1] = l;
31         memset(dp, -1, sizeof(dp));
32         printf("The minimum
33             cutting is %d.\n",
34             solve(0, n + 1));
35     }
36     return 0;
37 }

```

## 6.5 stringDP

• Edit distance

$S_1$  最少需要經過幾次增、刪或換字變成  $S_2$

$$dp[i][j] = \begin{cases} \min \begin{cases} dp[i-1][j-1] \\ dp[i][j-1] \\ dp[i-1][j] \end{cases} + 1 \end{cases}$$

• Longest Palindromic Subsequence

$$dp[l][r] = \begin{cases} 1 \\ \max \{ dp[l+1][r], dp[l][r-1] \} \end{cases}$$

## 6.6 TreeDP 有幾個 path 長度為 k

```

1 #define maxn 50005
2 #define maxk 505
3 //dp[u][u]的child且距離u長度k的數量
4 long long dp[maxn][maxk];
5 vector<vector<int>>> G;
6 int n, k;
7 long long res = 0;
8 void dfs(int u, int p) {
9     //u自己
10     dp[u][0] = 1;
11     for (int v: G[u]) {
12         if (v == p)
13             continue;
14         dfs(v, u);
15         for (int i = 1; i <= k;
16             ++i) {
17             //子樹v距離i -
18             //1的等於對於u來說距離i的
19             dp[u][i] += dp[v][i -
20                 1];
21         }
22     }
23     //統計在u子樹中距離u為k的數量
24     res += dp[u][k];
25     long long cnt = 0;
26     for (int v: G[u]) {
27         if (v == p)
28             continue; //重點算法
29         for (int x = 0; x <= k -
30             2; ++x) {

```

```

27         cnt += dp[v][x] *
28             (dp[u][k - x - 1]
29              - dp[v][k - x -
30              2]);
31     }
32     res += cnt / 2;
33 }
34 int main() {
35     ...
36     dfs(1, -1);
37     printf("%lld\n", res);
38     return 0;
39 }

```

## 6.7 TreeDP reroot

```

1  /*re-root dp on tree O(n + n + n)
2  -> O(n)*/
3  class Solution {
4  public:
5      vector<int>
6      sumOfDistancesInTree(int
7      n, vector<vector<int>>&
8      edges) {
9          this->res.assign(n, 0);
10         G.assign(n + 5,
11         vector<int>());
12         for (vector<int>& edge:
13         edges) {
14             G[edge[0]].emplace_back(edge[1]);
15             G[edge[1]].emplace_back(edge[0]);
16         }
17         memset(this->visited, 0,
18         sizeof(this->visited));
19         this->dfs(0);
20         memset(this->visited, 0,
21         sizeof(this->visited));
22         this->res[0] =
23         this->dfs2(0, 0);
24         memset(this->visited, 0,
25         sizeof(this->visited));
26         this->dfs3(0, n);
27         return this->res;
28     }
29 private:
30     vector<vector<int>> G;
31     bool visited[30005];
32     int subtreeSize[30005];
33     vector<int> res;
34     //求 subtreeSize
35     int dfs(int u) {
36         this->visited[u] = true;

```

```

37         for (int v: this->G[u]) {
38             if (!this->visited[v])
39             {
40                 this->subtreeSize[u]
41                 +=
42                 this->dfs(v);
43             }
44         }
45         //自己
46         this->subtreeSize[u] += 1;
47         return
48         this->subtreeSize[u];
49     }
50     //求 res[0], 0 到所有点的距离
51     int dfs2(int u, int dis) {
52         this->visited[u] = true;
53         int sum = 0;
54         for (int v: this->G[u]) {
55             if (!visited[v]) {
56                 sum +=
57                 this->dfs2(v,
58                 dis + 1);
59             }
60         }
61         //要加上自己的距离
62         return sum + dis;
63     }
64     //算出所有的 res
65     void dfs3(int u, int n) {
66         this->visited[u] = true;
67         for (int v: this->G[u]) {
68             if (!visited[v]) {
69                 this->res[v] =
70                 this->res[u] +
71                 n - 2 *
72                 this->subtreeSize[u];
73                 this->dfs3(v, n);
74             }
75         }
76     }
77 }

```

## 6.8 Weighted LIS

```

1  #define maxn 200005
2  long long dp[maxn];
3  long long height[maxn];
4  long long B[maxn];
5  long long st[maxn << 2];
6  void update(int p, int index, int
7  l, int r, 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,
15         mid, v);
16     else
17         update(p, (index << 1) +
18         1, mid + 1, r, v);
19     st[index] = max(st[index <<
20     1], st[(index << 1) + 1]);
21 }
22 long long query(int index, int l,
23 int r, int ql, int qr) {
24     if (ql <= l && r <= qr)
25         return st[index];
26     int mid = (l + r) >> 1;
27     long long res = -1;
28     if (ql <= mid)
29         res = max(res, query(index
30         << 1, l, mid, ql, qr));
31     if (mid < qr)
32         res = max(res,
33         query((index << 1) +
34         1, mid + 1, r, ql,
35         qr));
36     return res;
37 }
38 int main() {
39     int n;
40     scanf("%d", &n);
41     for (int i = 1; i <= n; ++i)
42         scanf("%lld", &height[i]);
43     for (int i = 1; i <= n; ++i)
44         scanf("%lld", &B[i]);
45     long long res = B[1];
46     update(height[1], 1, 1, n,
47     B[1]);
48     for (int i = 2; i <= n; ++i) {
49         long long temp;
50         if (height[i] - 1 >= 1)
51             temp = B[i] + query(1,
52             1, n, 1, height[i]
53             - 1);
54         else
55             temp = B[i];
56         update(height[i], 1, 1, n,
57         temp);
58         res = max(res, temp);
59     }
60     printf("%lld\n", res);
61     return 0;
62 }

```