

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
3	algorithm	2
3.1	三分搜	2
3.2	差分	2
3.3	greedy	2
3.4	dinic	3
3.5	SCC Tarjan	3
3.6	ArticulationPoints Tarjan	4
3.7	最小樹狀圖	4
3.8	JosephusProblem	4
3.9	KM	4
3.10	LCA 倍增法	5
3.11	MCMF	5
3.12	Dancing Links	6
4	DataStructure	6
4.1	線段樹 1D	6
4.2	線段樹 2D	6
4.3	權值線段樹	7
4.4	Trie	7
4.5	單調隊列	8
5	geometry	8
5.1	intersection	8
5.2	半平面相交	9
5.3	凸包	9
6	DP	9
6.1	抽屜	9
6.2	LCS 和 LIS	9
6.3	RangeDP	9
6.4	stringDP	9
6.5	TreeDP 有幾個 path 長度為 k	10
6.6	TreeDP reroot	10
6.7	WeightedLIS	10

1 字串

1.1 最長迴文子字串

```

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

```

```

38     cout<<ans-1<<"\n";
39     return 0;
40 }

```

1.2 KMP

```

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

```

2 math

2.1 SG

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

2.2 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 prime[MAXN];
5 int primeSize=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]) prime[primeSize++]=i;
11        for(int
12            j=0;j<primeSize&&i*prime[j]<=MAXN;+j){
13            isPrime[i*prime[j]]=false;
14            if(i%prime[j]==0) break;
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<(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    }
35 }

```

```

36 擴展歐幾里得算法
37 //ax+by=GCD(a,b)
38
39 int ext_euc(int a,int b,int &x,int &y){
40     if(b==0){
41         x=1,y=0;
42         return a;
43     }
44     int d=ext_euc(b,a%b,y,x);
45     y-=a/b*x;
46     return d;
47 }
48
49 int main(){
50     int a,b,x,y;
51     cin>>a>>b;
52     ext_euc(a,b,x,y);
53     cout<<x<<' '<<y<<endl;
54     return 0;
55 }
56
57
58 歌德巴赫猜想
59 solution : 把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
60 #define N 2000000
61 int ox[N],p[N],pr;
62 void PrimeTable(){
63     ox[0]=ox[1]=1;
64     pr=0;
65     for(int i=2;i<N;i++){
66         if(!ox[i]) p[pr++]=i;
67         for(int j=0;i*p[j]<N&&j<pr;j++)
68             ox[i*p[j]]=1;
69     }
70 }
71
72 int main(){
73     PrimeTable();
74     int n;
75     while(cin>>n,n){
76         int x;
77         for(x=1;;x+=2)
78             if(!ox[x]&&!ox[n-x]) break;
79         printf("%d = %d + %d\n",n,x,n-x);
80     }
81 }
82
83 problem : 給定整數 N ,
84 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
87 (強歌德巴赫猜想)。
88 如果 N 是奇數且 N-2 是質數 , 則答案為 2 (2+質數)。
89 其他狀況答案為 3 (弱歌德巴赫猜想)。
90
91 bool isPrime(int n){
92     for(int i=2;i<n;++i){
93         if(i*i>n) return true;
94         if(n%i==0) return false;
95     }
96     return true;
97 }
98
99 int main(){
100     int n;
101     cin>>n;
102     if(isPrime(n)) cout<<"1\n";
103     else if(n%2==0||isPrime(n-2))
104         cout<<"2\n";
105     else cout<<"3\n";
106 }

```

2.3 歐拉函數

```
58 }
59 }
```

3.2 差分

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

3.3 greedy

```
1 刪數字問題
2 //problem
3 給定一個數字 N (≤10^100)，需要刪除 K 個數字，
4 請問刪除 K 個數字後最小的數字為何？
5 //solution
6 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i
7 位數，
8 扣除高位數的影響較扣除低位數的大。
9 //code
10 int main(){
11     string s;
12     int k;
13     cin>>s>>k;
14     for(int i=0;i<k;++i){
15         if((int)s.size()==0) break;
16         int pos =(int)s.size()-1;
17         for(int j=0;j<(int)s.size()-1;++j){
18             if(s[j]>s[j+1]){
19                 pos=j;
20                 break;
21             }
22             s.erase(pos,1);
23         }
24         while((int)s.size()>0&&s[0]=='0')
25             s.erase(0,1);
26         if((int)s.size()) cout<<s<<'\n';
27         else cout<<0<<'\n';
28     }
29 最小區間覆蓋長度
30 //problem
31 給定 n 條線段區間為 [Li,Ri]，
32 請問最少要選幾個區間才能完全覆蓋 [0,S]？
33 //solution
34 先將所有區間依照左界由小到大批序，
35 對於當前區間 [Li,Ri]，要從左界 >Ri 的所有區間中，
36 找到有著最大的右界的區間，連接當前區間。
37 //problem
38 長度 n 的直線中有數個加熱器，
```

```
40 在 x 的加熱器可以讓 [x-r,x+r] 內的物品加熱，
41 問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
42 //solution
43 對於最左邊沒加熱的點a，選擇最遠可以加熱a的加熱器，
44 更新已加熱範圍，重複上述動作繼續尋找加熱器。
45 //code
46 int main(){
47     int n, r;
48     int a[1005];
49     cin>>n>>r;
50     for(int i=1;i<=n;++i) cin>>a[i];
51     int i=1,ans=0;
52     while(i<=n){
53         int R=min(i+r-1,n),L=max(i-r+1,0)
54         int nextR=-1;
55         for(int j=R;j>=L;--j){
56             if(a[j]){
57                 nextR=j;
58                 break;
59             }
60         }
61         if(nextR!=-1){
62             ans=-1;
63             break;
64         }
65         ++ans;
66         i=nextR+r;
67     }
68     cout<<ans<<'\n';
69 }
70 最多不重疊區間
71 //problem
72 給你 n 條線段區間為 [Li,Ri]，
73 請問最多可以選擇幾條不重疊的線段(頭尾可相連)？
74 //solution
75 依照右界由小到大批序，
76 每次取到一個不重疊的線段，答案 +1。
77 //code
78 struct Line{
79     int L,R;
80     bool operator<(const Line &rhs)const{
81         return R<rhs.R;
82     }
83 };
84 int main(){
85     int t;
86     cin>>t;
87     Line a[30];
88     while(t--){
89         int n=0;
90         while(cin>>a[n].L>>a[n].R,a[n].L||a[n].R)
91             ++n;
92         sort(a,a+n);
93         int ans=1,R=a[0].R;
94         for(int i=1;i<n;i++){
95             if(a[i].L>=R){
96                 ++ans;
97                 R=a[i].R;
98             }
99         }
100         cout<<ans<<'\n';
101     }
102 }
103 最小化最大延遲問題
104 //problem
105 給定 N 項工作，每項工作的需要處理時長為 Ti，
106 期限是 Di，第 i 項工作延遲的時間為
107     Li=max(0,Fi-Di)，
108 原本Fi 為第 i 項工作的完成時間，
109 求一種工作排序使 maxLi 最小。
110 //solution
111 按照到期時間從早到晚處理。
112 //code
113 struct Work{
114     int t, d;
115     bool operator<(const Work &rhs)const{
116         return d<rhs.d;
117     }
118 }
```

```
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    }
11    if(n>1) ans=ans-ans/n;
12    return ans;
13 }
14
15 3 algorithm
16 3.1 三分搜
17
18 1 題意
19 2 給定兩射線方向和速度，問兩射線最近距離。
20 3 題解
21 4 假設 F(t) 為兩射線在時間 t 的距離，F(t)
22    為二次函數，
23 5 可用三分查找二次函數最小值。
24 struct Point{
25     double x, y, z;
26     Point() {}
27     Point(double _x,double _y,double _z):
28         x(_x),y(_y),z(_z){}
29     friend istream& operator>>(istream& is,
30         Point& p) {
31         is >> p.x >> p.y >> p.z;
32         return is;
33     }
34 }
35 Point operator+(const Point &rhs) const{
36     return Point(x+rhs.x,y+rhs.y,z+rhs.z);
37 }
38 Point operator-(const Point &rhs) const{
39     return Point(x-rhs.x,y-rhs.y,z-rhs.z);
40 }
41 Point operator*(const double &d) const{
42     return Point(x*d,y*d,z*d);
43 }
44 Point operator/(const double &d) const{
45     return Point(x/d,y/d,z/d);
46 }
47 double dist(const Point &rhs) const{
48     double res = 0;
49     res+=(x-rhs.x)*(x-rhs.x);
50     res+=(y-rhs.y)*(y-rhs.y);
51     res+=(z-rhs.z)*(z-rhs.z);
52     return res;
53 }
54 };
55 int main(){
56     IOS; //輸入優化
57     int T;
58     cin>>T;
59     for(int ti=1;ti<=T;++ti){
60         double time;
61         Point x1,y1,d1,x2,y2,d2;
62         cin>>time>>x1>>y1>>x2>>y2;
63         d1=(y1-x1)/time;
64         d2=(y2-x2)/time;
65         double L=0,R=1e8,m1,m2,f1,f2;
66         double ans = x1.dist(x2);
67         while(abs(L-R)>1e-10){
68             m1=(L+R)/2;
69             m2=(m1+R)/2;
70             f1=((d1*m1)+x1).dist((d2*m1)+x2);
71             f2=((d1*m2)+x1).dist((d2*m2)+x2);
72             ans = min(ans,min(f1,f2));
73             if(f1<f2) R=m2;
74             else L=m1;
75         }
76         cout<<"Case "<<ti<<" ";
77         cout<<fixed<<setprecision(4)<<sqrt(ans)<<endl;
78     }
```

```

117 };
118 int main(){
119     int n;
120     Work a[10000];
121     cin>>n;
122     for(int i=0;i<n;++i)
123         cin>>a[i].t>>a[i].d;
124     sort(a,a+n);
125     int maxL=0,sumT=0;
126     for(int i=0;i<n;++i){
127         sumT+=a[i].t;
128         maxL=max(maxL,sumT-a[i].d);
129     }
130     cout<<maxL<<'\n';
131 }
132 最少延遲數量問題
133 //problem
134 給定 N 個工作，每個工作的需要處理時長為  $T_i$ ，
135 期限是  $D_i$ ，求一種工作排序使得逾期工作數量最小。
136 //solution
137 期限越早到期的工作越先做。將工作依照到期時間從早到晚按
138 依序放入工作列表中，如果發現有工作預期，
139 就從目前選擇的工作中，移除耗時最長的工作。
140 上述方法為 Moore-Hodgson's Algorithm。
141
142 //problem
143 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？
144 //solution
145 和最少延遲數量問題是相同的問題，只要將題敘做轉換。
146 工作處理時長  $\rightarrow$  烏龜重量
147 工作期限  $\rightarrow$  烏龜可承受重量
148 多少工作不延期  $\rightarrow$  可以疊幾隻烏龜
149 //code
150 struct Work{
151     int t, d;
152     bool operator<(const Work &rhs)const{
153         return d<rhs.d;
154     }
155 };
156 int main(){
157     int n=0;
158     Work a[10000];
159     priority_queue<int> pq;
160     while(cin>>a[n].t>>a[n].d)
161         ++n;
162     sort(a,a+n);
163     int sumT=0,ans=n;
164     for(int i=0;i<n;++i){
165         pq.push(a[i].t);
166         sumT+=a[i].t;
167         if(a[i].d<sumT){
168             int x=pq.top();
169             pq.pop();
170             sumT-=x;
171             --ans;
172         }
173     }
174     cout<<ans<<'\n';
175 }
176
177 任務調度問題
178 //problem
179 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
180 期限是  $D_i$ ，如果第  $i$  項工作延遲需要受到  $p_i$ 
    單位懲罰，
181 請問最少會受到多少單位懲罰。
182 //solution
183 依照懲罰由大到小排序，
184 每項工作依序嘗試可不可以放在
     $D_i - T_i + 1, D_i - T_i, \dots, 1, 0$ ，
185 如果有空間就放進去，否則延後執行。
186
187 //problem
188 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
189 期限是  $D_i$ ，如果第  $i$  項工作在期限內完成會獲得  $a_i$ 
    單位獎勵，
190 請問最多會獲得多少單位獎勵。
191 //solution

```

```

192 和上題相似，這題變成依照獎勵由大到小排序。
193 //code
194 struct Work{
195     int d,p;
196     bool operator<(const Work &rhs)const{
197         return p>rhs.p;
198     }
199 };
200 int main(){
201     int n;
202     Work a[100005];
203     bitset<100005> ok;
204     while(cin>>n){
205         ok.reset();
206         for(int i=0;i<n;++i)
207             cin>>a[i].d>>a[i].p;
208         sort(a,a+n);
209         int ans=0;
210         for(int i=0;i<n;++i){
211             int j=a[i].d;
212             while(j--){
213                 if(!ok[j]){
214                     ans+=a[i].p;
215                     ok[j]=true;
216                     break;
217                 }
218             }
219             cout<<ans<<'\n';
220         }
221     }

```

3.4 dinic

```

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

```

```

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

```

3.5 SCC Tarjan

```

1 //單純考SCC，每個SCC中找成本最小的蓋，如果有多個一樣小的
2 //注意以下程式有縮點，但沒存起來，存法就是開一個array
    -> ID[u] = SCCID
3 #define maxn 100005
4 #define MOD 1000000007
5 long long cost[maxn];
6 vector<vector<int>> G;
7 int SCC = 0;
8 stack<int> sk;
9 int dfn[maxn];
10 int low[maxn];
11 bool inStack[maxn];
12 int dfsTime = 1;
13 long long totalCost = 0;
14 long long ways = 1;
15 void dfs(int u) {
16     dfn[u] = low[u] = dfsTime;
17     ++dfsTime;
18     sk.push(u);
19     inStack[u] = true;
20     for (int v: G[u]) {
21         if (dfn[v] == 0) {
22             dfs(v);
23             low[u] = min(low[u], low[v]);
24         }
25         else if (inStack[v]) {
26             //屬於同個SCC且是我的back edge
27             low[u] = min(low[u], dfn[v]);
28         }
29     }
30     //如果是SCC
31     if (dfn[u] == low[u]) {
32         long long minCost = 0x3f3f3f3f;
33         int currWays = 0;
34         ++SCC;
35         while (1) {
36             int v = sk.top();
37             inStack[v] = 0;
38             sk.pop();
39             if (minCost > cost[v]) {
40                 minCost = cost[v];
41                 currWays = 1;
42             }
43             else if (minCost == cost[v]) {
44                 ++currWays;
45             }
46             if (v == u)
47                 break;
48         }
49         totalCost += minCost;
50         ways = (ways * currWays) % MOD;
51     }
52 }

```

```

53 int main() {
54     int n;
55     scanf("%d", &n);
56     for (int i = 1; i <= n; ++i)
57         scanf("%lld", &cost[i]);
58     G.assign(n + 5, vector<int>());
59     int m;
60     scanf("%d", &m);
61     int u, v;
62     for (int i = 0; i < m; ++i) {
63         scanf("%d %d", &u, &v);
64         G[u].emplace_back(v);
65     }
66     for (int i = 1; i <= n; ++i) {
67         if (dfn[i] == 0)
68             dfs(i);
69     }
70     printf("%lld %lld\n", totalCost, ways %
71           MOD);
72     return 0;

```

3.6 ArticulationPoints Tarjan

```

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

```

```

50         while (token = strtok(NULL, " "))
51             {
52                 v = atoi(token);
53                 G[u].emplace_back(v);
54                 G[v].emplace_back(u);
55             }
56         tarjan(1, -1);
57         printf("%d\n", res);
58     }
59     return 0;
60 }

```

3.7 最小樹狀圖

```

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

```

```

57     N = seq;
58     root = idx[root];
59     }
60 }

```

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11
4 int getWinner(int n, int k) {
5     int winner = 0;
6     for (int i = 1; i <= n; ++i)
7         winner = (winner + k) % i;
8     return winner;
9 }
10 int main() {
11     int n;
12     while (scanf("%d", &n) != EOF && n){
13         --n;
14         for (int k = 1; k <= n; ++k){
15             if (getWinner(n, k) == 11){
16                 printf("%d\n", k);
17                 break;
18             }
19         }
20     }
21     return 0;
22 }

```

3.9 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
28 // labeling看看能不能產生出新的增廣路(KM的增廣路要求L
29 // + 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         }

```

```

38 }
39 for (int i = 0; i < n; ++i) {
40     if (S[i]) Lx[i] -= diff;
41     if (T[i]) Ly[i] += diff;
42 }
43 }
44 void KM()
45 {
46     for (int i = 0; i < n; ++i) {
47         L[i] = -1;
48         Lx[i] = Ly[i] = 0;
49         for (int j = 0; j < n; ++j)
50             Lx[i] = max(Lx[i], W[i][j]);
51     }
52     for (int i = 0; i < n; ++i) {
53         while(1) {
54             memset(S, false, sizeof(S));
55             memset(T, false, sizeof(T));
56             if (match(i))
57                 break;
58             else
59                 update(); //去調整vertex
                        //labeling以增加增廣路徑
60         }
61     }
62 }
63 int main() {
64     while (scanf("%d", &n) != EOF) {
65         for (int i = 0; i < n; ++i)
66             for (int j = 0; j < n; ++j)
67                 scanf("%d", &W[i][j]);
68         KM();
69         int res = 0;
70         for (int i = 0; i < n; ++i) {
71             if (i != 0)
72                 printf(" %d", Lx[i]);
73             else
74                 printf("%d", Lx[i]);
75             res += Lx[i];
76         }
77         puts("");
78         for (int i = 0; i < n; ++i) {
79             if (i != 0)
80                 printf(" %d", Ly[i]);
81             else
82                 printf("%d", Ly[i]);
83             res += Ly[i];
84         }
85         puts("");
86         printf("%d\n", res);
87     }
88     return 0;
89 }

```

3.10 LCA 倍增法

```

1 //倍增法預處理  $O(n \log n)$ ，查詢  $O(\log n)$ ，利用 lca 找樹上任兩點距離
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] -> u 的第  $2^i$  個祖先
8 long long dis[maxn][31];
9 int dep[maxn]; //深度
10 void dfs(int u, int p) { //預處理 fa
11     fa[u][0] = p; //因為 u 的第  $2^0 = 1$  的祖先就是 p
12     dep[u] = dep[p] + 1;
13     //第  $2^i$  的祖先是 (第  $2^{i-1}$  個祖先) 的第  $2^{i-1}$  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] + dis[u][i-1];
18     }
19     //遍歷子節點
20     for (Edge& edge: G[u]) {
21         if (edge.v == p)
22             continue;
23         dis[edge.v][0] = edge.w;
24         dfs(edge.v, u);
25     }
26 }
27 long long lca(int x, int y)
28 { //此函數是找 lca 同時計算 x、y 的距離 -> dis(x, 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, deltaDep >>= 1)
36         if (deltaDep & 1)
37             res += dis[y][i], y = fa[y][i];
38     if (y == x) //x = y -> x、y 彼此是彼此的祖先
39         //往上找，一起跳，但 x、y 不能重疊
40         for (int i = 30; i >= 0 && y != x; --i) {
41             if (fa[x][i] != fa[y][i]) {
42                 res += dis[x][i] + dis[y][i];
43                 x = fa[x][i];
44                 y = fa[y][i];
45             }
46         }
47     //最後發現不能跳了，此時 x 的第  $2^0 = 1$  個祖先(或說 y 的第  $2^0 = 1$  的祖先)即為 x、y 的 lca
48     res += dis[x][0] + dis[y][0];
49     return res;
50 }
51 int main() {
52     int n, q;
53     while (~scanf("%d", &n) && n) {
54         int v, w;
55         G.assign(n + 5, vector<Edge>());
56         for (int i = 1; i <= n - 1; ++i) {
57             scanf("%d %d", &v, &w);
58             G[i + 1].push_back({i + 1, v + 1, w});
59             G[v + 1].push_back({v + 1, i + 1, w});
60         }
61         dfs(1, 0);
62         scanf("%d", &q);
63         int u;
64         while (q--) {
65             scanf("%d %d", &u, &v);
66             printf("%lldc", lca(u + 1, v + 1), (q ? ' ' : '\n'));
67         }
68     }
69     return 0;
70 }

```

3.11 MCMF

```

1 //倍增法預處理  $O(n \log n)$ ，查詢  $O(\log n)$ ，利用 lca 找樹上任兩點距離
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] -> u 的第  $2^i$  個祖先
8 long long dis[maxn][31];
9 int dep[maxn]; //深度
10 void dfs(int u, int p) { //預處理 fa
11     fa[u][0] = p; //因為 u 的第  $2^0 = 1$  的祖先就是 p
12     dep[u] = dep[p] + 1;
13     //第  $2^i$  的祖先是 (第  $2^{i-1}$  個祖先) 的第  $2^{i-1}$  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] + dis[u][i-1];
18     }
19     //遍歷子節點
20     for (Edge& edge: G[u]) {
21         if (edge.v == p)
22             continue;
23         dis[edge.v][0] = edge.w;
24         dfs(edge.v, u);
25     }
26 }
27 long long lca(int x, int y)
28 { //此函數是找 lca 同時計算 x、y 的距離 -> dis(x, 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, deltaDep >>= 1)
36         if (deltaDep & 1)
37             res += dis[y][i], y = fa[y][i];
38     if (y == x) //x = y -> x、y 彼此是彼此的祖先
39         //往上找，一起跳，但 x、y 不能重疊
40         for (int i = 30; i >= 0 && y != x; --i) {
41             if (fa[x][i] != fa[y][i]) {
42                 res += dis[x][i] + dis[y][i];
43                 x = fa[x][i];
44                 y = fa[y][i];
45             }
46         }
47     //最後發現不能跳了，此時 x 的第  $2^0 = 1$  個祖先(或說 y 的第  $2^0 = 1$  的祖先)即為 x、y 的 lca
48     res += dis[x][0] + dis[y][0];
49     return res;
50 }
51 int main() {
52     int n, q;
53     while (~scanf("%d", &n) && n) {
54         int v, w;
55         G.assign(n + 5, vector<Edge>());
56         for (int i = 1; i <= n - 1; ++i) {
57             scanf("%d %d", &v, &w);
58             G[i + 1].push_back({i + 1, v + 1, w});
59             G[v + 1].push_back({v + 1, i + 1, w});
60         }
61         dfs(1, 0);
62         scanf("%d", &q);
63         int u;
64         while (q--) {
65             scanf("%d %d", &u, &v);
66             printf("%lldc", lca(u + 1, v + 1), (q ? ' ' : '\n'));
67         }
68     }
69     return 0;
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 //<注> 在這題因為 G[] 中存的是 edgeIndex in edges[]
16 //
17 所以 parent 存的也是對應 edges[] 中的 edgeIndex (主要是)
18 int parent[maxn];
19 //maxFlow 時需要紀錄到 node u 時的 bottleneck
20 //同時也代表著 u 該次流出去的量
21 long long outFlow[maxn];
22 void addEdge(int u, int v, int cap, int cost) {
23     edges.emplace_back(Edge{u, v, cap, 0, cost});
24     edges.emplace_back(Edge{v, u, 0, 0, -cost});
25     m = edges.size();
26     G[u].emplace_back(m - 2);
27     G[v].emplace_back(m - 1);
28 }
29 //一邊求最短路的同时一邊 MaxFlow
30 bool SPFA(long long &maxFlow, long long &minCost) {
31     //memset(outFlow, 0x3f, sizeof(outFlow));
32     memset(dis, 0x3f, sizeof(dis));
33     memset(inqueue, false, sizeof(inqueue));
34     queue<int> q;
35     q.push(s);
36     dis[s] = 0;
37     inqueue[s] = true;
38     outFlow[s] = INF;
39     while (!q.empty()) {
40         int u = q.front();
41         q.pop();
42         inqueue[u] = false;
43         for (const int edgeIndex: G[u]) {
44             const Edge& edge = edges[edgeIndex];
45             if ((edge.cap > edge.flow) && (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], (long long)(edge.cap - edge.flow));
49                 if (!inqueue[edge.v]) {
50                     q.push(edge.v);
51                     inqueue[edge.v] = true;
52                 }
53             }
54         }
55     }
56     //如果 dis[t] > 0 代表根本不賺還倒賠
57     if (dis[t] > 0)
58         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 MaxFlow 演算法
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         ;
75     return minCost;
76 }
77 int main() {
78     int T;
79     scanf("%d", &T);
80 }

```



```

79 for (int Case = 1; Case <= T; ++Case){
80     //總共幾個月，囤貨成本
81     int M, I;
82     scanf("%d %d", &M, &I);
83     //node size
84     n = M + M + 2;
85     G.assign(n + 5, vector<int>());
86     edges.clear();
87     s = 0;
88     t = M + M + 1;
89     for (int i = 1; i <= M; ++i) {
90         int produceCost, produceMax,
91             sellPrice, sellMax,
92             inventoryMonth;
93         scanf("%d %d %d %d %d",
94             &produceCost, &produceMax,
95             &sellPrice, &sellMax,
96             &inventoryMonth);
97         addEdge(s, i, produceMax,
98             produceCost);
99         addEdge(M + i, t, sellMax,
100             -sellPrice);
101         for (int j = 0; j <=
102             inventoryMonth; ++j) {
103             if (i + j <= M)
104                 addEdge(i, M + i + j, INF,
105                     I * j);
106         }
107     }
108     printf("Case %d: %lld\n", Case,
109         -MCMF());
110 }
111 return 0;
112 }

```

3.12 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=c]]=0=c;
13        resSize = -1;
14        memset(rowHead, 0, sizeof(rowHead));
15        memset(colSize, 0, sizeof(colSize));
16    }
17    void insert(int r, int c) {
18        row[++seq]=r, col[seq]=c,
19        ++colSize[c];
20        U[seq]=c, D[seq]=D[c], U[D[c]]=seq,
21        D[c]=seq;
22        if(rowHead[r]) {
23            L[seq]=rowHead[r],
24            R[seq]=R[rowHead[r]];
25            L[R[rowHead[r]]]=seq,
26            R[rowHead[r]]=seq;
27        } else {
28            rowHead[r] = L[seq] = R[seq] =
29            seq;
30        }
31    }
32    void remove(int c) {
33        L[R[c]] = L[c], R[L[c]] = R[c];
34        for(int i=D[c]; i!=c; i=D[i]) {
35            for(int j=R[i]; j!=i; j=R[j]) {
36                U[D[j]] = U[j];
37                D[U[j]] = D[j];
38                --colSize[col[j]];
39            }
40        }
41    }
42 }

```

```

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

```

4 DataStructure

4.1 線段樹 1D

```

1 #define MAXN 1000
2 int data[MAXN]; //原數據
3 int st[4 * MAXN]; //線段樹
4 int tag[4 * MAXN]; //懶標
5 inline int pull(int l, int r) {
6     // 隨題目改變sum、max、min
7     // l、r是左右樹的index
8     return st[l] + st[r];
9 }
10 void build(int l, int r, int i) {
11     // 在[l, r]區間建樹，目前根的index為i
12     if (l == r) {
13         st[i] = data[l];
14         return;
15     }
16     int mid = l + ((r - l) >> 1);
17     build(l, mid, i * 2);
18     build(mid + 1, r, i * 2 + 1);
19     st[i] = pull(i * 2, i * 2 + 1);
20 }

```

```

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

```

4.2 線段樹 2D

```

1 //純2D segment tree 區間查詢單點修改最大最小值
2 #define maxn 2005 //500 * 4 + 5
3 int maxST[maxn][maxn], minST[maxn][maxn];
4 int N;
5 void modifyY(int index, int l, int r, int
6     val, int yPos, int xIndex, bool
7     xIsLeaf) {
8     if (l == r) {
9         if (xIsLeaf) {
10             maxST[xIndex][index] =
11             minST[xIndex][index] = val;
12             return;
13         }
14         maxST[xIndex][index] =
15         max(maxST[xIndex * 2][index],
16         maxST[xIndex * 2 + 1][index]);
17         minST[xIndex][index] =
18         min(minST[xIndex * 2][index],
19         minST[xIndex * 2 + 1][index]);
20     }
21     else {
22         int mid = (l + r) / 2;
23         if (yPos <= mid)
24             modifyY(index * 2, l, mid, val,
25                 yPos, xIndex, xIsLeaf);
26     }
27 }

```

```

18     else
19         modifyY(index * 2 + 1, mid + 1,
20             r, val, yPos, xIndex,
21             xIsLeaf);
22
23     maxST[xIndex][index] =
24         max(maxST[xIndex][index * 2],
25             maxST[xIndex][index * 2 + 1]);
26     minST[xIndex][index] =
27         min(minST[xIndex][index * 2],
28             minST[xIndex][index * 2 + 1]);
29 }
30
31 void modifyX(int index, int l, int r, int
32     val, int xPos, int yPos) {
33     if (l == r) {
34         modifyY(1, 1, N, val, yPos, index,
35             true);
36     }
37     else {
38         int mid = (l + r) / 2;
39         if (xPos <= mid)
40             modifyX(index * 2, l, mid, val,
41                 xPos, yPos);
42         else
43             modifyX(index * 2 + 1, mid + 1,
44                 r, val, xPos, yPos);
45         modifyY(1, 1, N, val, yPos, index,
46             false);
47     }
48 }
49
50 void queryY(int index, int l, int r, int
51     yql, int yqr, int xIndex, int& vmax,
52     int& vmin) {
53     if (yql <= l && r <= yqr) {
54         vmax = max(vmax,
55             maxST[xIndex][index]);
56         vmin = min(vmin,
57             minST[xIndex][index]);
58     }
59     else
60     {
61         int mid = (l + r) / 2;
62         if (yql <= mid)
63             queryY(index * 2, l, mid, yql,
64                 yqr, xIndex, vmax, vmin);
65         if (mid < yqr)
66             queryY(index * 2 + 1, mid + 1, r,
67                 yql, yqr, xIndex, vmax,
68                 vmin);
69     }
70 }
71
72 void queryX(int index, int l, int r, int
73     xql, int xqr, int yql, int yqr, int&
74     vmax, int& vmin) {
75     if (xql <= l && r <= xqr) {
76         queryY(1, 1, N, yql, yqr, index,
77             vmax, vmin);
78     }
79     else {
80         int mid = (l + r) / 2;
81         if (xql <= mid)
82             queryX(index * 2, l, mid, xql,
83                 xqr, yql, yqr, vmax, vmin);
84         if (mid < xqr)
85             queryX(index * 2 + 1, mid + 1, r,
86                 xql, xqr, yql, yqr, vmax,
87                 vmin);
88     }
89 }
90
91 int main() {
92     while (scanf("%d", &N) != EOF) {
93         int val;
94         for (int i = 1; i <= N; ++i) {
95             for (int j = 1; j <= N; ++j) {
96                 scanf("%d", &val);
97                 modifyX(1, 1, N, val, i, j);
98             }
99         }
100     }

```

```

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,
83                 &yql, &xqr, &yqr);
84             vmax = -0x3f3f3f3f;
85             vmin = 0x3f3f3f3f;
86             queryX(1, 1, N, xql, xqr,
87                 yql, yqr, vmax, vmin);
88             printf("%d %d\n", vmax, vmin);
89         }
90         else {
91             scanf("%d %d %d", &xql, &yql,
92                 &val);
93             modifyX(1, 1, N, val, xql,
94                 yql);
95         }
96     }
97     return 0;
98 }

```

4.3 權值線段樹

```

1 //權值線段樹 + 離散化 解決區間第k小問題
2 //其他網路上的解法: 2個heap, Treap, AVL tree
3 #define maxn 30005
4 int nums[maxn];
5 int getArr[maxn];
6 int id[maxn];
7 int st[maxn << 2];
8 void update(int index, int l, int r, int qx)
9 {
10     if (l == r)
11     {
12         ++st[index];
13         return;
14     }
15     int mid = (l + r) / 2;
16     if (qx <= mid)
17         update(index * 2, l, mid, qx);
18     else
19         update(index * 2 + 1, mid + 1, r, qx);
20     st[index] = st[index * 2] + st[index * 2
21         + 1];
22 }
23 //找區間第k個小的
24 int query(int index, int l, int r, int k) {
25     if (l == r)
26         return id[l];
27     int mid = (l + r) / 2;
28     //k比左子樹小
29     if (k <= st[index * 2])
30         return query(index * 2, l, mid, k);
31     else
32         return query(index * 2 + 1, mid + 1,
33             r, k - st[index * 2]);
34 }
35
36 int main() {
37     int t;
38     cin >> t;
39     bool first = true;
40     while (t--) {
41         if (first)
42             first = false;
43         else
44             puts("");
45         memset(st, 0, sizeof(st));
46         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 +
56         1) - (id + 1);
57     for (int i = 1; i <= m; ++i) {
58         nums[i] = lower_bound(id + 1, id
59             + stSize + 1, nums[i]) - id;
60     }
61     int addCount = 0;
62     int getCount = 0;
63     int k = 1;
64     while (getCount < n) {
65         if (getArr[getCount] == addCount)
66         {
67             printf("%d\n", query(1, 1,
68                 stSize, k));
69             ++k;
70             ++getCount;
71         }
72         else {
73             update(1, 1, stSize,
74                 nums[addCount + 1]);
75             ++addCount;
76         }
77     }
78     return 0;
79 }

```

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; r < len; ++r) {
27             int c = str[idx] - 'a';
28             if (!r = mp[r][c]) return result;
29             if (val[r])
30                 result = (result + dp[idx +
31                     1]) % mod;
32         }
33         return result;
34     }
35 };
36
37 int main() {
38     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，你就可以退役了。"--單調隊列
3
4 example
5
6 給出一個長度為 n 的數組，
7 輸出每 k 個連續的數中的最大值和最小值。
8
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14
15 void getmin() {
16     //
17     得到這個隊列裡的最小值，直接找到最後的就行了
18     int head=0, tail=0;
19     for(int i=1; i<=n; i++) {
20         while(head<=tail && a[q[tail]]>=a[i])
21             tail--;
22         q[++tail]=i;
23     }
24     for(int i=k; i<=n; i++) {
25         while(head<=tail && a[q[tail]]>=a[i])
26             tail--;
27         q[++tail]=i;
28         while(q[head]<=i-k) head++;
29         cout<<a[q[head]]<<" ";
30     }
31     cout<<endl;
32 }
33
34 void getmax() { // 和上面同理
35     int head=0, tail=0;
36     for(int i=1; i<=n; i++) {
37         while(head<=tail && a[q[tail]]<=a[i]) tail++;
38         q[++tail]=i;
39     }
40     for(int i=k; i<=n; i++) {
41         while(head<=tail && a[q[tail]]<=a[i]) tail++;
42         q[++tail]=i;
43         while(q[head]<=i-k) head++;
44         cout<<a[q[head]]<<" ";
45     }
46     cout<<endl;
47 }
48
49 int main(){
50     cin>>n>>k; //每k個連續的數

```

5 geometry

5.1 intersection

```

1 using LL = long long;
2
3 struct Point2D {
4     LL x, y;
5 };
6
7 struct Line2D {
8     Point2D s, e;
9     LL a, b, c; // L: ax + by = c
10    Line2D(Point2D s, Point2D e): s(s), e(e)
11    {
12        a = e.y - s.y;
13        b = s.x - e.x;
14        c = a * s.x + b * s.y;
15    }
16 };
17 // 用克拉克馬公式求二元一次解
18 Point2D intersection2D(Line2D l1, Line2D l2)
19 {
20     LL D = l1.a * l2.b - l2.a * l1.b;
21     LL Dx = l1.c * l2.b - l2.c * l1.b;
22     LL Dy = l1.a * l2.c - l2.a * l1.c;
23
24     if(D) { // intersection
25         double x = 1.0 * Dx / D;
26         double y = 1.0 * Dy / D;
27     } else {
28         if(Dx || Dy) // Parallel lines
29             return Point2D(0, 0);
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):
10         x(x), y(y) {}
11
12     Vector operator+(Vector v) {
13         return Vector(x+v.x, y+v.y);
14     }
15     Vector operator-(Vector v) {
16         return Vector(x-v.x, y-v.y);
17     }
18     Vector operator*(double val) {
19         return Vector(x*val, y*val);
20     }
21     double dot(Vector v) { return x*v.x + y*v.y; }
22     double cross(Vector v) { return x*v.y - y*v.x; }
23     double length() { return sqrt(dot(*this)); }
24     Vector unit_normal_vector() {
25         double len = length();
26         return Vector(-y/len, x/len);
27     }
28 };

```

```

29 using Point = Vector;
30
31 struct Line {
32     Point p;
33     Vector v;
34     double ang;
35     Line(Point p={}, Vector v={}): p(p),
36         v(v) {
37         ang = atan2(v.y, v.x);
38     }
39     bool operator<(const Line& l) const {
40         return ang < l.ang;
41     }
42     Point intersection(Line l) {
43         Vector u = p - l.p;
44         double t = l.v.cross(u) /
45             v.cross(l.v);
46         return p + v*t;
47     }
48 };
49 int n, m;
50 Line narrow[maxn]; // 要判斷的直線
51 Point poly[maxn]; // 能形成半平面交的凸包邊界點
52
53 // return true if point p is on the left of
54 // line l
55 bool onLeft(Point p, Line l) {
56     return l.v.cross(p-l.p) > 0;
57 }
58
59 int halfplaneIntersection() {
60     int l, r;
61     Line L[maxn]; // 排序後的向量隊列
62     Point P[maxn]; // s[i] 跟 s[i-1] 的交點
63
64     L[l=r=0] = narrow[0]; // notice: narrow
65     // is sorted
66     for(int i=1; i<=n; i++) {
67         while(l<r && !onLeft(P[r-1],
68             narrow[i])) r--;
69         while(l<r && !onLeft(P[l],
70             narrow[i])) l++;
71
72         L[++r] = narrow[i];
73         if(l < r) P[r-1] =
74             L[r-1].intersection(L[r]);
75     }
76
77     while(l<r && !onLeft(P[r-1], L[l])) r--;
78     if(r-l <= 1) return 0;
79
80     P[r] = L[r].intersection(L[l]);
81
82     int m=0;
83     for(int i=l; i<=r; i++) {
84         poly[m++] = P[i];
85     }
86     return m;
87 }
88
89 Point pt[maxn];
90 Vector vec[maxn];
91 Vector normal[maxn]; // normal[i] = vec[i]
92 // 的單位法向量
93
94 double bsearch(double l=0.0, double r=1e4) {
95     if(abs(r-l) < eps) return l;
96
97     double mid = (l + r) / 2;
98
99     for(int i=0; i<=m; i++) {
100         narrow[i] = Line(pt[i]+normal[i]*mid,
101             vec[i]);
102     }

```



```

96
97 if(halfplaneIntersection())
98     return bsearch(mid, r);
99 else return bsearch(l, mid);
100 }
101
102 int main() {
103     while(~scanf("%d", &n) && n) {
104         for(int i=0; i<n; i++) {
105             double x, y;
106             scanf("%lf%lf", &x, &y);
107             pt[i] = {x, y};
108         }
109         for(int i=0; i<n; i++) {
110             vec[i] = pt[(i+1)%n] - pt[i];
111             normal[i] =
                vec[i].unit_normal_vector();
112         }
113
114         printf("%.6lf\n", bsearch());
115     }
116     return 0;
117 }

```

5.3 凸包

```

1 //
2 // Q: 平面上給定多個區域，由多個座標點所形成，再給定
3 // 多點(x,y)，判斷有落點的區域(destroyed)的面積總和
4 #include <bits/stdc++.h>
5 using namespace std;
6 const int maxn = 500 + 10;
7 const int maxCoordinate = 500 + 10;
8
9 struct Point {
10     int x, y;
11 };
12
13 int n;
14 bool destroyed[maxn];
15 Point arr[maxn];
16 vector<Point> polygons[maxn];
17
18 void scanAndSortPoints() {
19     int minX = maxCoordinate, minY =
        maxCoordinate;
20     for(int i=0; i<n; i++) {
21         int x, y;
22         scanf("%d%d", &x, &y);
23         arr[i] = (Point){x, y};
24         if(y < minY || (y == minY && x <
            minX)) {
25             // If there are floating points, use:
26             // if(y<minY || (abs(y-minY)<eps &&
27             // x<minX)) {
28                 minX = x, minY = y;
29             }
30     }
31     sort(arr, arr+n, [minX, minY](Point& a,
        Point& b){
32         double theta1 = atan2(a.y - minY, a.x
            - minX);
33         double theta2 = atan2(b.y - minY, b.x
            - minX);
34         return theta1 < theta2;
35     });
36     return;
37 }
38
39 // returns cross product of u(AB) x v(AC)
40 int cross(Point& A, Point& B, Point& C) {
41     int u[2] = {B.x - A.x, B.y - A.y};
42     int v[2] = {C.x - A.x, C.y - A.y};
43     return (u[0] * v[1]) - (u[1] * v[0]);
44 }

```

```

44
45 // size of arr = n >= 3
46 // st = the stack using vector, m = index of
    the top
47 vector<Point> convex_hull() {
48     vector<Point> st(arr, arr+3);
49     for(int i=3, m=2; i<n; i++, m++) {
50         while(m >= 2) {
51             if(cross(st[m], st[m-1], arr[i])
                < 0)
52                 break;
53             st.pop_back();
54             m--;
55         }
56         st.push_back(arr[i]);
57     }
58     return st;
59 }
60
61 bool inPolygon(vector<Point>& vec, Point p) {
62     vec.push_back(vec[0]);
63     for(int i=1; i<vec.size(); i++) {
64         if(cross(vec[i-1], vec[i], p) < 0) {
65             vec.pop_back();
66             return false;
67         }
68     }
69     vec.pop_back();
70     return true;
71 }
72
73
74
75
76 double calculateArea(vector<Point>& v) {
77     v.push_back(v[0]); // make v[n] =
        v[0]
78     double result = 0.0;
79     for(int i=1; i<v.size(); i++)
80         result += v[i-1].x*v[i].y -
            v[i-1].y*v[i].x;
81     v.pop_back();
82     return result / 2.0;
83 }
84
85 int main() {
86     int p = 0;
87     while(~scanf("%d", &n) && (n != -1)) {
88         scanAndSortPoints();
89         polygons[p++] = convex_hull();
90     }
91
92     int x, y;
93     double result = 0.0;
94     while(~scanf("%d%d", &x, &y)) {
95         for(int i=0; i<p; i++) {
96             if(inPolygon(polygons[i],
                (Point){x, y}))
97                 destroyed[i] = true;
98         }
99     }
100     for(int i=0; i<p; i++) {
101         if(destroyed[i])
102             result +=
                calculateArea(polygons[i]);
103     }
104     printf("%.2lf\n", result);
105     return 0;
106 }

```

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 = (底下i -
    1個抽屜且1個安全且最上方L) + (底下n -
    1個抽屜0個安全且最上方為0)
6 dp[i][0][0] = dp[i - 1][1][1] + dp[i -
    1][0][0];
7 for (int j = 1; j <= i; ++j) {
8     dp[i][j][0] = dp[i - 1][j + 1][1] +
        dp[i - 1][j][0];
9     dp[i][j][1] = dp[i - 1][j - 1][1] +
        dp[i - 1][j - 1][0];
10 }
11 } //答案在 dp[n][s][0] + dp[n][s][1]);

```

6.2 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 A 的子序列。
7 //LCS 和 LIS 題目轉換
8 LIS 轉成 LCS
9     1. A 為原序列，B=sort(A)
10    2. 對 A,B 做 LCS
11 LCS 轉成 LIS
12    1. A, B 為原本的兩序列
13    2. 最 A 序列作編號轉換，將轉換規則套用在 B
14    3. 對 B 做 LIS
15    4. 重複的數字在編號轉換時後要變成不同的數字，
16        越早出現的數字要越小
17    5. 如果有數字在 B 裡面而不在 A 裡面，
18        直接忽略這個數字不做轉換即可

```

6.3 RangedDP

```

1 //區間dp
2 int dp[55][55]; // dp[i][j] -> [i,
    j]切割區間中最小的cost
3 int cuts[55];
4 int solve(int i, int j) {
5     if (dp[i][j] != -1)
6         return dp[i][j];
7     //代表沒有其他切法，只能是cuts[j] - cuts[i]
8     if (i == j - 1)
9         return dp[i][j] = 0;
10    int cost = 0x3f3f3f3f;
11    for (int m = i + 1; m < j; ++m) {
12        //枚舉區間中間切點
13        cost = min(cost, solve(i, m) +
            solve(m, j) + cuts[j] - cuts[i]);
14    }
15    return dp[i][j] = cost;
16 }
17 int main() {
18     int l;
19     int 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] = l;
26         memset(dp, -1, sizeof(dp));
27         printf("The minimum cutting is
            %d.\n", solve(0, n + 1));
28     }
29     return 0;
30 }

```

6.4 stringDP

• Edit distance

S_1 最少需要經過幾次增、刪或換字變成 S_2

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

• Longest Palindromic Subsequence

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

6.5 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; ++i) {
16            //子樹v距離i-1的等於對於u來說距離i的
17            dp[u][i] += dp[v][i-1];
18        }
19    }
20    //統計在u子樹中距離u為k的數量
21    res += dp[u][k];
22    long long cnt = 0;
23    for (int v: G[u]) {
24        if (v == p)
25            continue; //重點算法
26        for (int x = 0; x <= k-2; ++x) {
27            cnt += 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 }
```

6.6 TreeDP reroot

```
1 /*re-root dp on tree O(n + n + n) -> O(n)*/*
2 class Solution {
3 public:
```

```
4     vector<int> sumOfDistancesInTree(int n,
5     vector<vector<int>>& edges) {
6         this->res.assign(n, 0);
7         G.assign(n+5, vector<int>());
8         for (vector<int>& edge: edges) {
9             G[edge[0]].emplace_back(edge[1]);
10            G[edge[1]].emplace_back(edge[0]);
11        }
12        memset(this->visited, 0,
13            sizeof(this->visited));
14        this->dfs(0);
15        memset(this->visited, 0,
16            sizeof(this->visited));
17        this->res[0] = this->dfs2(0, 0);
18        memset(this->visited, 0,
19            sizeof(this->visited));
20        this->dfs3(0, n);
21        return this->res;
22    }
23 private:
24     vector<vector<int>>> G;
25     bool visited[30005];
26     int subtreeSize[30005];
27     vector<int> res;
28     //求subtreeSize
29     int dfs(int u) {
30         this->visited[u] = true;
31         for (int v: this->G[u]) {
32             if (!this->visited[v]) {
33                 this->subtreeSize[u] +=
34                     this->dfs(v);
35             }
36         }
37         //自己
38         this->subtreeSize[u] += 1;
39         return this->subtreeSize[u];
40     }
41     //求res[0], 0到所有點的距離
42     int dfs2(int u, int dis) {
43         this->visited[u] = true;
44         int sum = 0;
45         for (int v: this->G[u]) {
46             if (!visited[v]) {
47                 sum += this->dfs2(v, dis+1);
48             }
49         }
50         //要加上自己的距離
51         return sum + dis;
52     }
53     //算出所有的res
54     void dfs3(int u, int n) {
55         this->visited[u] = true;
56         for (int v: this->G[u]) {
57             if (!visited[v]) {
58                 this->res[v] = this->res[u] +
59                     n - 2 *
60                     this->subtreeSize[v];
61                 this->dfs3(v, n);
62             }
63         }
64     }
65 }
```

6.7 Weighted LIS

```
1 #define maxn 200005
2 long long dp[maxn];
3 long long height[maxn];
4 long long B[maxn];
5 long long st[maxn << 2];
6 void update(int p, int index, int l, int r,
7     long long v) {
8     if (l == r) {
9         st[index] = v;
10        return;
11    }
12    int mid = (l + r) >> 1;
13    if (p <= mid)
14        update(p, (index << 1), l, mid, v);
15    else
16        update(p, (index << 1) + 1, mid + 1,
17            r, v);
18    st[index] = max(st[index << 1],
19        st[(index << 1) + 1]);
20 }
21 long long query(int index, int l, int r, int
22     ql, int qr) {
23     if (ql <= l && r <= qr)
24         return st[index];
25     int mid = (l + r) >> 1;
26     long long res = -1;
27     if (ql <= mid)
28         res = max(res, query(index << 1, l,
29             mid, ql, qr));
30     if (mid < qr)
31         res = max(res, query((index << 1) +
32             1, mid + 1, r, ql, qr));
33     return res;
34 }
35 int main() {
36     int n;
37     scanf("%d", &n);
38     for (int i = 1; i <= n; ++i)
39         scanf("%lld", &height[i]);
40     for (int i = 1; i <= n; ++i)
41         scanf("%lld", &B[i]);
42     long long res = B[1];
43     update(height[1], 1, 1, n, B[1]);
44     for (int i = 2; i <= n; ++i) {
45         long long temp;
46         if (height[i] - 1 >= 1)
47             temp = B[i] + query(1, 1, n, 1,
48                 height[i] - 1);
49         else
50             temp = B[i];
51         update(height[i], 1, 1, n, temp);
52         res = max(res, temp);
53     }
54     printf("%lld\n", res);
55     return 0;
56 }
```