

Contents

1 字串

- 1.1 最長迴文子字串
- 1.2 KMP

2 math

- 2.1 SG
- 2.2 Fibonacci
- 2.3 矩陣快速冪
- 2.4 質數與因數
- 2.5 歐拉函數

3 algorithm

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

4 DataStructure

- 4.1 線段樹 1D
- 4.2 線段樹 2D
- 4.3 權值線段樹
- 4.4 Trie
- 4.5 單調隊列

5 geometry

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

6 DP

- 6.1 抽屜
- 6.2 Deque 最大差距
- 6.3 LCS 和 LIS
- 6.4 RangeDP
- 6.5 stringDP
- 6.6 樹 DP 有幾個 path 長度為 k
- 6.7 TreeDP reroot
- 6.8 WeightedLIS

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& s) {
4     nextArr[0] = 0;
5     int prefixLen = 0;
6     for (int i = 1; i < s.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 && s[prefixLen]!=s[i])
11            prefixLen = nextArr[prefixLen - 1];
12        //一樣就繼承之前的前後綴長度+1
13        if (s[prefixLen] == s[i])
14            ++prefixLen;
15        nextArr[i] = prefixLen;
16    }
17    for (int i = 0; i < s.size() - 1; ++i) {
18        vis[nextArr[i]] = true;
19    }
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 Fibonacci

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

2.3 矩陣快速冪

```

1 using ll = long long;
2 using mat = vector<vector<ll>>>;
3 const int mod = 1e9 + 7;
4
5 mat operator*(mat A, mat B) {
6     mat res(A.size(),
7             vector<ll>(B[0].size()));
8     for(int i=0; i<A.size(); i++) {
9         for(int j=0; j<B[0].size(); j++) {
10            for(int k=0; k<B.size(); k++) {
11                res[i][j] += A[i][k] *
12                    B[k][j] % mod;
13            }
14        }
15        return res;
16    }
17 }
18 mat I = ;
19 // compute matrix M^n
20 // 需先 init I 矩陣
21 mat mpow(mat& M, int n) {
22     if(n <= 1) return n ? M : I;
23     mat v = mpow(M, n>>1);
24     return (n & 1) ? v*v*M : v*v;

```

```

25 }
26
27 // 迴圈版本
28 mat mpow(mat M, int n) {
29     mat res(M.size(),
30             vector<ll>(M[0].size()));
31     for(int i=0; i<res.size(); i++)
32         res[i][i] = 1;
33     for(; n; n>>=1) {
34         if(n & 1) res = res * M;
35         M = M * M;
36     }
37     return res;
38 }

```

2.4 質數與因數

```

1 歐拉篩O(n)
2 #define MAXN 47000 //sqrt(2^31)=46,340...
3 bool isPrime[MAXN];
4 int p[MAXN];
5 int pSize=0;
6 void getPrimes(){
7     memset(isPrime,true,sizeof(isPrime));
8     isPrime[0]=isPrime[1]=false;
9     for(int i=2;i<MAXN;i++){
10        if(isPrime[i]) p[pSize++]=i;
11        for(int j=0;j<pSize&&i*p[j]<=MAXN;j++){
12            isPrime[i*p[j]]=false;
13            if(i%p[j]==0) break;
14        }
15    }
16 }
17
18 最大公因數 O(log(min(a,b)))
19 int GCD(int a, int b){
20     if(b == 0) return a;
21     return GCD(b, a%b);
22 }
23
24 質因數分解
25 void primeFactorization(int n){
26     for(int i=0; i<p.size(); ++i) {
27         if(p[i]*p[i] > n) break;
28         if(n % p[i]) continue;
29         cout << p[i] << ' ';
30         while(n%p[i] == 0) n /= p[i];
31     }
32     if(n != 1) cout << n << ' ';
33     cout << '\n';
34 }
35
36 擴展歐幾里得算法 ax + by = GCD(a, b)
37 int ext_euc(int a, int b, int &x, int &y) {
38     if(b == 0){
39         x = 1, y = 0;
40         return a;
41     }
42     int d = ext_euc(b, a%b, y, x);
43     y -= a/b*x;
44     return d;
45 }
46 int main(){
47     int a, b, x, y;
48     cin >> a >> b;
49     ext_euc(a, b, x, y);
50     cout << x << ' ' << y << endl;
51     return 0;
52 }
53
54
55 歌德巴赫猜想
56 解：把偶數 N (6≤N≤10^6) 寫成兩個質數的 和。
57 #define N 20000000
58 int ox[N], p[N], pr;
59 void PrimeTable(){

```

```

61 ox[0] = ox[1] = 1;
62 pr = 0;
63 for(int i=2;i<N;i++){
64     if(!ox[i]) p[pr++] = i;
65     for(int j=0; i*p[j]<N&&j<pr; j++)
66         ox[i*p[j]] = 1;
67 }
68 }
69 int main(){
70     PrimeTable();
71     int n;
72     while(cin>>n, n){
73         int x;
74         for(x=1;; x+=2)
75             if(!ox[x] && !ox[n-x]) break;
76         printf("%d = %d + %d\n", n, x, n-x);
77     }
78 }
79
80 problem :
81 給定整數 N，求 N 最少可以拆成多少個質數的和。
82 如果 N 是質數，則答案為 1。
83 如果 N 是偶數 (N!=2)，則答案為 2 (強歌德巴赫猜想)。
84 如果 N 是奇數且 N-2 是質數，則答案為 2 (2+質數)。
85 其他狀況答案為 3 (弱歌德巴赫猜想)。
86
87 bool isPrime(int n){
88     for(int i=2;i<n;++i){
89         if(i*i>n) return true;
90         if(n%i==0) return false;
91     }
92     return true;
93 }
94 int main(){
95     int n;
96     cin>>n;
97     if(isPrime(n)) cout<<"1\n";
98     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
99     else cout<<"3\n";
100 }

```

2.5 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    }
11    if(n>1) ans=ans-ans/n;
12    return ans;
13 }

```

3 algorithm

3.1 三分搜

```

1 題意
2 給定兩射線方向和速度，問兩射線最近距離。
3 題解
4 假設 F(t) 為兩射線在時間 t 的距離，F(t)
  為二次函數，
5 可用三分查找二次函數最小值。
6 struct Point{
7     double x, y, z;
8     Point() {}
9     Point(double _x, double _y, double _z):
10         x(_x), y(_y), z(_z){}
11     friend istream& operator>>(istream& is,
12         Point& p) {
13         is >> p.x >> p.y >> p.z;
14         return is;
15     }
16     Point operator+(const Point &rhs) const{

```

```

16         return Point(x+rhs.x, y+rhs.y, z+rhs.z);
17     }
18     Point operator-(const Point &rhs) const{
19         return Point(x-rhs.x, y-rhs.y, z-rhs.z);
20     }
21     Point operator*(const double &d) const{
22         return Point(x*d, y*d, z*d);
23     }
24     Point operator/(const double &d) const{
25         return Point(x/d, y/d, z/d);
26     }
27     double dist(const Point &rhs) const{
28         double res = 0;
29         res+=(x-rhs.x)*(x-rhs.x);
30         res+=(y-rhs.y)*(y-rhs.y);
31         res+=(z-rhs.z)*(z-rhs.z);
32         return res;
33     }
34 };
35 int main(){
36     IOS; //輸入優化
37     int T;
38     cin>>T;
39     for(int ti=1; ti<=T; ++ti){
40         double time;
41         Point x1, y1, d1, x2, y2, d2;
42         cin>>time>>x1>>y1>>x2>>y2;
43         d1=(y1-x1)/time;
44         d2=(y2-x2)/time;
45         double L=0, R=1e8, m1, m2, f1, f2;
46         double ans = x1.dist(x2);
47         while(abs(L-R)>1e-10){
48             m1=(L+R)/2;
49             m2=(m1+R)/2;
50             f1=((d1*m1)+x1).dist((d2*m1)+x2);
51             f2=((d1*m2)+x1).dist((d2*m2)+x2);
52             ans = min(ans, min(f1, f2));
53             if(f1<f2) R=m2;
54             else L=m1;
55         }
56         cout<<"Case "<<ti<<" : ";
57         cout << fixed << setprecision(4) <<
58             sqrt(ans) << '\n';
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
  位數，
7 扣除高位數的影響較扣除低位數的大。
8 //code
9 int main(){
10     string s;
11     int k;
12     cin>>s>>k;
13     for(int i=0; i<k; ++i){
14         if((int)s.size()==0) break;
15         int pos =(int)s.size()-1;
16         for(int j=0; j<(int)s.size()-1; ++j){
17             if(s[j]>s[j+1]){
18                 pos=j;
19                 break;
20             }
21         }
22         s.erase(pos, 1);
23     }
24     while((int)s.size()>0&&s[0]=='0')
25         s.erase(0, 1);
26     if((int)s.size()) cout<<s<<'\n';
27     else cout<<0<<'\n';
28 }
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排序，
35 對於當前區間 [Li, Ri]，要從左界 >Ri 的所有區間中，
36 找到有著最大的右界的區間，連接當前區間。
37
38 //problem
39 長度 n 的直線中有數個加熱器，
40 在 x 的加熱器可以讓 [x-r, x+r] 內的物品加熱，
41 問最少要幾個加熱器可以把 [0, n] 的範圍加熱。
42 //solution
43 對於最左邊沒加熱的點 a，選擇最遠可以加熱 a 的加熱器，
44 更新已加熱範圍，重複上述動作繼續尋找加熱器。
45 //code
46 int main(){
47     int n, r;
48     int a[1005];
49     cin>>n>>r;
50     for(int i=1; i<=n; ++i) cin>>a[i];
51     int i=1, ans=0;
52     while(i<=n){
53         int R=min(i+r-1, n), L=max(i-r+1, 0)
54         int nextR=-1;
55         for(int j=R; j>=L; --j){
56             if(a[j]){
57                 nextR=j;
58                 break;
59             }
60         }
61         if(nextR!=-1){
62             ans=-1;
63             break;
64         }
65         ++ans;
66         i=nextR+r;
67     }
68     cout<<ans<<'\n';
69 }
70 最多不重疊區間
71 //problem
72 給你 n 條線段區間為 [Li, Ri]，
73 請問最多可以選擇幾條不重疊的線段(頭尾可相連)？
74 //solution
75 依照右界由小到大排序，

```

```

76 每次取到一個不重疊的線段，答案 +1。
77 //code
78 struct Line{
79     int L,R;
80     bool operator<(const Line &rhs)const{
81         return R<rhs.R;
82     }
83 };
84 int main(){
85     int t;
86     cin>>t;
87     Line a[30];
88     while(t--){
89         int n=0;
90         while(cin>>a[n].L>>a[n].R,a[n].L||a[n].R){
91             ++n;
92         }
93         sort(a,a+n);
94         int ans=1,R=a[0].R;
95         for(int i=1;i<n;i++){
96             if(a[i].L>=R){
97                 ++ans;
98                 R=a[i].R;
99             }
100         }
101         cout<<ans<<'\n';
102     }
103 }
104 //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 };
119 int main(){
120     int n;
121     Work a[10000];
122     cin>>n;
123     for(int i=0;i<n;++i)
124         cin>>a[i].t>>a[i].d;
125     sort(a,a+n);
126     int maxL=0,sumT=0;
127     for(int i=0;i<n;++i){
128         sumT+=a[i].t;
129         maxL=max(maxL,sumT-a[i].d);
130     }
131     cout<<maxL<<'\n';
132 }
133 //problem
134 給定 N 個工作，每個工作的需要處理時長為 Ti，
135 期限是 Di，求一種工作排序使得逾期工作數量最小。
136 //solution
137 期限越早到期的工作越先做。
138 將工作依照到期時間從早到晚排序，
139 依序放入工作列表中，如果發現有工作預期，
140 就從目前選擇的工作中，移除耗時最長的工作。
141 上述方法為 Moore-Hodgson's Algorithm。
142 //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 → 烏龜重量
147 工作期限 → 烏龜可承受重量
148 多少工作不延期 → 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 //task scheduling problem
177 //problem
178 給定 N 項工作，每項工作的需要處理時長為 Ti，
179 期限是 Di，如果第 i 項工作延遲需要受到 pi
180     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
185     Di-Ti+1,Di-Ti,...,1,0，
186 如果有空間就放進去，否則延後執行。
187 //problem
188 給定 N 項工作，每項工作的需要處理時長為 Ti，
189 期限是 Di，如果第 i 項工作在期限內完成會獲得 ai
190     單位獎勵，
191 請問最多會獲得多少單位獎勵。
192 //solution
193 和上題相似，這題變成依照獎勵由大到小排序。
194 //code
195 struct Work{
196     int d,p;
197     bool operator<(const Work &rhs)const{
198         return p>rhs.p;
199     }
200 };
201 int main(){
202     int n;
203     Work a[100005];
204     bitset<100005> ok;
205     while(cin>>n){
206         ok.reset();
207         for(int i=0;i<n;++i)
208             cin>>a[i].d>>a[i].p;
209         sort(a,a+n);
210         int ans=0;
211         for(int i=0;i<n;++i){
212             int j=a[i].d;
213             while(j--){
214                 if(!ok[j]){
215                     ans+=a[i].p;
216                     ok[j]=true;
217                     break;
218                 }
219             }
220             cout<<ans<<'\n';
221         }
222     }
223 }
224 //dinic
225 //code
226 struct Edge{
227     int s, t, cap, flow;
228 };
229 int n, m, S, T;
230 int level[maxn], dfs_idx[maxn];
231 vector<Edge> E;
232 vector<vector<int>> G;
233 void init() {
234     S = 0;
235     T = n + m;
236     E.clear();
237     G.assign(maxn, vector<int>());
238 }
239 void addEdge(int s, int t, int cap) {
240     E.push_back({s, t, cap, 0});
241     E.push_back({t, s, 0, 0});
242     G[s].push_back(E.size()-2);
243     G[t].push_back(E.size()-1);
244 }
245 bool bfs() {
246     queue<int> q({S});
247     memset(level, -1, sizeof(level));
248     level[S] = 0;
249     while(!q.empty()) {
250         int cur = q.front();
251         q.pop();
252         for(int i : G[cur]) {
253             Edge e = E[i];
254             if(level[e.t]==-1 &&
255                 e.cap>e.flow) {
256                 level[e.t] = level[e.s] + 1;
257                 q.push(e.t);
258             }
259         }
260     }
261     return level[T];
262 }
263 int dfs(int cur, int lim) {
264     if(cur==T || lim==0) return lim;
265     int result = 0;
266     for(int& i=dfs_idx[cur]; i<G[cur].size()
267         && lim; i++) {
268         Edge& e = E[G[cur][i]];
269         if(level[e.s]+1 != level[e.t])
270             continue;
271         int flow = dfs(e.t, min(lim,
272             e.cap-e.flow));
273         if(flow <= 0) continue;
274         e.flow += flow;
275         result += flow;
276         E[G[cur][i]^1].flow -= flow;
277         lim -= flow;
278     }
279     return result;
280 }
281 int dinic() { // O(V^2E)
282     int result = 0;
283     while(bfs()) {
284         memset(dfs_idx, 0, sizeof(dfs_idx));
285         result += dfs(S, inf);
286     }
287     return result;
288 }
289 //SCC Tarjan
290 //code
291 //單純考SCC，每個SCC中找成本最小的蓋，如果有多個一樣小
292 //的要數出來，因為題目要方法數
293 //注意以下程式有縮點，但沒存起來，
294 //存法就是開一個array -> ID[u] = SCCID
295 #define maxn 100005
296 #define MOD 1000000007
297 long long cost[maxn];
298 vector<vector<int>> G;
299 int SCC = 0;
300 stack<int> sk;
301 const int maxn = 1e5 + 10;

```

```

11 int dfn[maxn];
12 int low[maxn];
13 bool inStack[maxn];
14 int dfsTime = 1;
15 long long totalCost = 0;
16 long long ways = 1;
17 void dfs(int u) {
18     dfn[u] = low[u] = dfsTime;
19     ++dfsTime;
20     sk.push(u);
21     inStack[u] = true;
22     for (int v: G[u]) {
23         if (dfn[v] == 0) {
24             dfs(v);
25             low[u] = min(low[u], low[v]);
26         }
27         else if (inStack[v]) {
28             //屬於同個SCC且是我的back edge
29             low[u] = min(low[u], dfn[v]);
30         }
31     }
32     //如果是SCC
33     if (dfn[u] == low[u]) {
34         long long minCost = 0x3f3f3f3f;
35         int currWays = 0;
36         ++SCC;
37         while (1) {
38             int v = sk.top();
39             inStack[v] = 0;
40             sk.pop();
41             if (minCost > cost[v]) {
42                 minCost = cost[v];
43                 currWays = 1;
44             }
45             else if (minCost == cost[v]) {
46                 ++currWays;
47             }
48             if (v == u)
49                 break;
50         }
51         totalCost += minCost;
52         ways = (ways * currWays) % MOD;
53     }
54 }
55 int main() {
56     int n;
57     scanf("%d", &n);
58     for (int i = 1; i <= n; ++i)
59         scanf("%lld", &cost[i]);
60     G.assign(n + 5, vector<int>());
61     int m;
62     scanf("%d", &m);
63     int u, v;
64     for (int i = 0; i < m; ++i) {
65         scanf("%d %d", &u, &v);
66         G[u].emplace_back(v);
67     }
68     for (int i = 1; i <= n; ++i) {
69         if (dfn[i] == 0)
70             dfs(i);
71     }
72     printf("%lld %lld\n", totalCost, ways % MOD);
73     return 0;
74 }

```

3.6 ArticulationPoints Tarjan

```

1 vector<vector<int>>> G;
2 int N, timer;
3 bool visited[105];
4 int dfn[105]; // 第一次visit的時間
5 int low[105];
6 //最小能回到的父節點
7 //(不能是自己的parent)的visTime
8 int res;
9 //求割點數量

```

```

10 void tarjan(int u, int parent) {
11     int child = 0;
12     bool isCut = false;
13     visited[u] = true;
14     dfn[u] = low[u] = ++timer;
15     for (int v: G[u]) {
16         if (!visited[v]) {
17             ++child;
18             tarjan(v, u);
19             low[u] = min(low[u], low[v]);
20             if (parent != -1 && low[v] >= dfn[u])
21                 isCut = true;
22         }
23         else if (v != parent)
24             low[u] = min(low[u], dfn[v]);
25     }
26     //If u is root of DFS
27     //tree->有兩個以上的children
28     if (parent == -1 && child >= 2)
29         isCut = true;
30     if (isCut) ++res;
31 }
32 int main() {
33     char input[105];
34     char* token;
35     while (scanf("%d", &N) != EOF && N) {
36         G.assign(105, vector<int>());
37         memset(visited, false, 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;
52             while (token = strtok(NULL, " ")) {
53                 v = atoi(token);
54                 G[u].emplace_back(v);
55                 G[v].emplace_back(u);
56             }
57             tarjan(1, -1);
58             printf("%d\n", res);
59         }
60         return 0;
61     }

```

3.7 最小樹狀圖

```

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

```

```

18             if(e.cap < low) continue;
19             if(e.s!=e.t && e.cost<inEdge[e.t]) {
20                 inEdge[e.t] = e.cost;
21                 pre[e.t] = e.s;
22             }
23         }
24         for(int i=0; i<N; i++) {
25             if(i!=root && inEdge[i]==inf)
26                 return -1; //除了root 還有沒有 in edge
27         }
28         int seq = inEdge[root] = 0;
29         memset(idx, -1, sizeof(idx));
30         memset(vis, -1, sizeof(vis));
31         // 找所有的 cycle，一起編號為 seq
32         for(int i=0; i<N; i++) {
33             result += inEdge[i];
34             int cur = i;
35             while(vis[cur]!=i && idx[cur]==-1) {
36                 if(cur == root) break;
37                 vis[cur] = i;
38                 cur = pre[cur];
39             }
40             if(cur!=root && idx[cur]==-1) {
41                 for(int j=pre[cur]; j!=cur; j=pre[j])
42                     idx[j] = seq;
43                 idx[cur] = seq++;
44             }
45         }
46         if(seq == 0) return result; // 沒有 cycle
47         for(int i=0; i<N; i++)
48             // 沒有被縮點的點
49             if(idx[i] == -1) idx[i] = seq++;
50         // 縮點並重新編號
51         for(Edge& e : edges) {
52             if(idx[e.s] != idx[e.t])
53                 e.cost -= inEdge[e.t];
54             e.s = idx[e.s];
55             e.t = idx[e.t];
56         }
57         N = seq;
58         root = idx[root];
59     }
60 }

```

3.8 二分圖最大匹配

```

1 /* 核心：最大點獨立集 = |V| - /最大匹配數/，用匈牙利演算法找出最大匹配數 */
2 vector<Student> boys;
3 vector<Student> girls;
4 vector<vector<int>>> G;
5 bool used[505];
6 int p[505];
7 bool match(int i) {
8     for (int j: G[i]) {
9         if (!used[j]) {
10             used[j] = true;
11             if (p[j] == -1 || match(p[j])) {
12                 p[j] = i;
13                 return true;
14             }
15         }
16     }
17     return false;
18 }
19 void maxMatch(int n) {
20     memset(p, -1, sizeof(p));
21     int res = 0;
22     for (int i = 0; i < boys.size(); ++i) {
23         memset(used, false, sizeof(used));
24         if (match(i))
25             ++res;

```

```

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

```

3.9 JosephusProblem

```

1 //JosephusProblem, 只是規定要先砍1號
2 //所以當作有 n - 1個人, 目標的13順移成12
3 //再者從0開始比較好算, 所以目標12順移成11
4 int getWinner(int n, int k) {
5     int winner = 0;
6     for (int i = 1; i <= n; ++i)
7         winner = (winner + k) % i;
8     return winner;
9 }
10 int main() {
11     int n;
12     while (scanf("%d", &n) != EOF && n){
13         --n;
14         for (int k = 1; k <= n; ++k){
15             if (getWinner(n, k) == 11){
16                 printf("%d\n", k);
17                 break;
18             }
19         }
20     }
21     return 0;
22 }

```

3.10 KM

```

1 #define maxn 505
2 int W[maxn][maxn];
3 int Lx[maxn], Ly[maxn];
4 bool S[maxn], T[maxn];
5 //L[i] = j -> S_i配給T_j, -1 for 還沒匹配
6 int L[maxn];
7 int n;
8 bool match(int i) {
9     S[i] = true;
10    for (int j = 0; j < n; ++j) {
11        // KM重點
12        // Lx + Ly >= selected_edge(x, y)
13        // 要想辦法降低Lx + Ly
14        // 所以選Lx + Ly == selected_edge(x, y)
15        if (Lx[i] + Ly[j] == W[i][j] &&
16            !T[j]) {
17            T[j] = true;
18            if ((L[j] == -1) || match(L[j])) {
19                L[j] = i;
20                return true;
21            }
22        }
23    }
24    return false;
25 }
26 //修改二分圖上的交錯路徑上點的權重
27 //此舉是在通過調整vertex labeling看看
28 //能不能產生出新的增廣路
29 //((KM的增廣路要求Lx[i] + Ly[j] == W[i][j]))
30 //在這裡優先從最小的diff調調看, 才能保證最大權重匹配
31 void update() {
32     {
33         int diff = 0x3f3f3f3f;
34         for (int i = 0; i < n; ++i) {
35             if (S[i]) {
36                 for (int j = 0; j < n; ++j) {
37                     if (!T[j])
38                         diff = min(diff, Lx[i] +
39                                     Ly[j] - W[i][j]);
40                 }
41             }
42         }
43         for (int i = 0; i < n; ++i) {

```

```

42         if (S[i]) Lx[i] -= diff;
43         if (T[i]) Ly[i] += diff;
44     }
45 }
46 void KM()
47 {
48     for (int i = 0; i < n; ++i) {
49         L[i] = -1;
50         Lx[i] = Ly[i] = 0;
51         for (int j = 0; j < n; ++j)
52             Lx[i] = max(Lx[i], W[i][j]);
53     }
54     for (int i = 0; i < n; ++i) {
55         while(1) {
56             memset(S, false, sizeof(S));
57             memset(T, false, sizeof(T));
58             if (match(i))
59                 break;
60             else
61                 update(); //去調整vertex
62                             //labeling以增加增廣路徑
63         }
64     }
65 }
66 int main() {
67     while (scanf("%d", &n) != EOF) {
68         for (int i = 0; i < n; ++i)
69             for (int j = 0; j < n; ++j)
70                 scanf("%d", &W[i][j]);
71         KM();
72         int res = 0;
73         for (int i = 0; i < n; ++i) {
74             if (i != 0)
75                 printf(" %d", Lx[i]);
76             else
77                 printf("%d", Lx[i]);
78             res += Lx[i];
79         }
80         puts("");
81         for (int i = 0; i < n; ++i) {
82             if (i != 0)
83                 printf(" %d", Ly[i]);
84             else
85                 printf("%d", Ly[i]);
86             res += Ly[i];
87         }
88         puts("");
89         printf("%d\n", res);
90     }
91     return 0;
92 }

```

3.11 LCA 倍增法

```

1 //倍增法預處理O(nlogn), 查詢O(logn),
2 //利用lca找樹上任兩點距離
3 #define maxn 100005
4 struct Edge {
5     int u, v, w;
6 };
7 vector<vector<Edge>> G; // tree
8 int fa[maxn][31]; //fa[u][i] -> u的第2^i個祖先
9 long long dis[maxn][31];
10 int dep[maxn]; //深度
11 void dfs(int u, int p) { //預處理fa
12     fa[u][0] = p; //因為u的第2^0 = 1的祖先就是p
13     dep[u] = dep[p] + 1;
14     //第2^i的祖先是(第2^(i-1)個祖先)的
15     //第2^(i-1)的祖先
16     //ex: 第8個祖先是 (第4個祖先)的第4個祖先
17     for (int i = 1; i < 31; ++i) {
18         fa[u][i] = fa[fa[u][i-1]][i-1];
19         dis[u][i] = dis[fa[u][i-1]][i-1]
20             + dis[u][i-1];
21     }
22     //遍歷子節點
23     for (Edge& edge: G[u]) {

```

```

23         if (edge.v == p)
24             continue;
25         dis[edge.v][0] = edge.w;
26         dfs(edge.v, u);
27     }
28 }
29 long long lca(int x, int y) {
30     //此函數是找lca同時計算x、y的距離 -> dis(x,
31     //lca) + dis(lca, y)
32     //讓y比x深
33     if (dep[x] > dep[y])
34         swap(x, y);
35     int deltaDep = dep[y] - dep[x];
36     long long res = 0;
37     //讓y與x在同一個深度
38     for (int i = 0; deltaDep != 0; ++i,
39         deltaDep >>= 1)
40         if (deltaDep & 1)
41             res += dis[y][i], y = fa[y][i];
42     if (y == x) //x = y -> x、y彼此是彼此的祖先
43         return res;
44     //往上找, 一起跳, 但x、y不能重疊
45     for (int i = 30; i >= 0 && y != x; --i) {
46         if (fa[x][i] != fa[y][i]) {
47             res += dis[x][i] + dis[y][i];
48             x = fa[x][i];
49             y = fa[y][i];
50         }
51     }
52     //最後發現不能跳了, 此時x的第2^0 =
53     //1個祖先(或說y的第2^0 =
54     //1的祖先)即為x、y的lca
55     res += dis[x][0] + dis[y][0];
56     return res;
57 }
58 int main() {
59     int n, q;
60     while (~scanf("%d", &n) && n) {
61         G.assign(n + 5, vector<Edge>());
62         for (int i = 1; i <= n - 1; ++i) {
63             scanf("%d %d", &v, &w);
64             G[i + 1].push_back({i + 1, v + 1, w});
65             G[v + 1].push_back({v + 1, i + 1, w});
66         }
67         dfs(1, 0);
68         scanf("%d", &q);
69         int u;
70         while (q--) {
71             scanf("%d %d", &u, &v);
72             printf("%lld%c", lca(u + 1, v +
73                 1), (q ? ' ': '\n'));
74         }
75     }
76     return 0;
77 }

```

3.12 MCMF

```

1 #define maxn 225
2 #define INF 0x3f3f3f3f
3 struct Edge {
4     int u, v, cap, flow, cost;
5 };
6 //node size, edge size, source, target
7 int n, m, s, t;
8 vector<vector<int>> G;
9 vector<Edge> edges;
10 bool inqueue[maxn];
11 long long dis[maxn];
12 int parent[maxn];
13 long long outFlow[maxn];
14 void addEdge(int u, int v, int cap, int
15     cost) {
16     edges.emplace_back(Edge(u, v, cap, 0,
17         cost));

```



```

16 edges.emplace_back(Edge{v, u, 0, 0,
17   -cost});
18 m = edges.size();
19 G[u].emplace_back(m - 2);
20 G[v].emplace_back(m - 1);
21 }
22 //一邊求最短路的同時一邊MaxFlow
23 bool SPFA(long long& maxFlow, long long&
24   minCost) {
25   // memset(outFlow, 0x3f,
26     sizeof(outFlow));
27   memset(dis, 0x3f, sizeof(dis));
28   memset(inqueue, false, sizeof(inqueue));
29   queue<int> q;
30   q.push(s);
31   dis[s] = 0;
32   inqueue[s] = true;
33   outFlow[s] = INF;
34   while (!q.empty()) {
35     int u = q.front();
36     q.pop();
37     inqueue[u] = false;
38     for (const int edgeIndex: G[u]) {
39       const Edge& edge =
40         edges[edgeIndex];
41       if ((edge.cap > edge.flow) &&
42         (dis[edge.v] > dis[u] +
43         edge.cost)) {
44         dis[edge.v] = dis[u] +
45         edge.cost;
46         parent[edge.v] = edgeIndex;
47         outFlow[edge.v] =
48         min(outFlow[u], (long
49         long)(edge.cap -
50         edge.flow));
51         if (!inqueue[edge.v]) {
52           q.push(edge.v);
53           inqueue[edge.v] = true;
54         }
55       }
56     }
57   }
58   //如果dis[t] > 0代表根本不賺還倒賠
59   if (dis[t] > 0)
60     return false;
61   maxFlow += outFlow[t];
62   minCost += dis[t] * outFlow[t];
63   //一路更新回去這次最短路流完後要維護的
64   //MaxFlow演算法相關(如反向邊等)
65   int curr = t;
66   while (curr != s) {
67     edges[parent[curr]].flow +=
68     outFlow[t];
69     edges[parent[curr] ^ 1].flow -=
70     outFlow[t];
71     curr = edges[parent[curr]].u;
72   }
73   return true;
74 }
75 long long MCMF() {
76   long long maxFlow = 0;
77   long long minCost = 0;
78   while (SPFA(maxFlow, minCost))
79     ;
80   return minCost;
81 }
82 int main() {
83   int T;
84   scanf("%d", &T);
85   for (int Case = 1; Case <= T; ++Case){
86     //總共幾個月，囤貨成本
87     int M, I;
88     scanf("%d %d", &M, &I);
89     //node size
90     n = M + M + 2;
91     G.assign(n + 5, vector<int>());
92     edges.clear();
93     s = 0;

```

```

82 t = M + M + 1;
83 for (int i = 1; i <= M; ++i) {
84   int produceCost, produceMax,
85     sellPrice, sellMax,
86     inventoryMonth;
87   scanf("%d %d %d %d %d",
88     &produceCost, &produceMax,
89     &sellPrice, &sellMax,
90     &inventoryMonth);
91   addEdge(s, i, produceMax,
92     produceCost);
93   addEdge(M + i, t, sellMax,
94     -sellPrice);
95   for (int j = 0; j <=
96     inventoryMonth; ++j) {
97     if (i + j <= M)
98       addEdge(i, M + i + j, INF,
99         I * j);
100   }
101 }
102 printf("Case %d: %lld\n", Case,
103   -MCMF());
104 }
105 return 0;
106 }

```

3.13 Dancing Links

```

1 struct DLX {
2   int seq, resSize;
3   int col[maxn], row[maxn];
4   int U[maxn], D[maxn], R[maxn], L[maxn];
5   int rowHead[maxn], colSize[maxn];
6   int result[maxn];
7   DLX(int r, int c) {
8     for(int i=0; i<=c; i++) {
9       L[i] = i-1, R[i] = i+1;
10      U[i] = D[i] = i;
11    }
12    L[R[seq=c]=0]=c;
13    resSize = -1;
14    memset(rowHead, 0, sizeof(rowHead));
15    memset(colSize, 0, sizeof(colSize));
16  }
17  void insert(int r, int c) {
18    row[++seq]=r, col[seq]=c,
19    ++colSize[c];
20    U[seq]=c, D[seq]=D[c], U[D[c]]=seq,
21    D[c]=seq;
22    if(rowHead[r]) {
23      L[seq]=rowHead[r],
24      R[seq]=R[rowHead[r]];
25      L[R[rowHead[r]]]=seq,
26      R[rowHead[r]]=seq;
27    } else {
28      rowHead[r] = L[seq] = R[seq] =
29      seq;
30    }
31  }
32  void remove(int c) {
33    L[R[c]] = L[c], R[L[c]] = R[c];
34    for(int i=D[c]; i!=c; i=D[i]) {
35      for(int j=R[i]; j!=i; j=R[j]) {
36        U[D[j]] = U[j];
37        D[U[j]] = D[j];
38        --colSize[col[j]];
39      }
40    }
41  }
42  void recover(int c) {
43    for(int i=U[c]; i!=c; i=U[i]) {
44      for(int j=L[i]; j!=i; j=L[j]) {
45        U[D[j]] = D[U[j]] = j;
46        ++colSize[col[j]];
47      }
48    }
49    L[R[c]] = R[L[c]] = c;

```

```

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);

```

```

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

```

4.2 線段樹 2D

```

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

```

```

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

```

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 int main() {
36     int t;
37     cin >> t;
38     bool first = true;
39     while (t--) {
40         if (first)
41             first = false;
42         else
43             puts("");
44         memset(st, 0, sizeof(st));
45         int m, n;
46         cin >> m >> n;
47         for (int i = 1; i <= m; ++i) {
48             cin >> nums[i];
49             id[i] = nums[i];
50         }
51         for (int i = 0; i < n; ++i)
52             cin >> getArr[i];
53         //離散化
54         //防止m == 0
55         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; r < len; r++) {
27             int c = str[idx] - 'a';
28             if (!mp[r][c]) return result;
29             if (val[r])
30                 result = (result + dp[idx + 1]) % mod;
31         }
32         return result;
33     }
34 };
35 int main() {
36     int n, tc = 1;
37     while (~scanf("%s", 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 ****Output****
58 * Case 1: 2
59 ****
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。"
3
4 example
5
6 給出一個長度為 n 的數組，
7 輸出每 k 個連續的數中的最大值和最小值。
8
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14 //得到這個隊列裡的最小值，直接找到最後的就行了
15 void getmin() {
16     int head = 0, tail = 0;
17     for (int i = 1; i <= n; i++) {
18         while (head <= tail && a[q[tail]] >= a[i])
19             tail--;
20         q[++tail] = i;
21     }
22     for (int i = k; i <= n; i++) {
23         while (head <= tail && a[q[tail]] >= a[i])
24             tail--;
25         q[++tail] = i;
26         while (q[head] <= i - k) head++;
27         cout << a[q[head]] << " ";
28     }
29     // 和上面同理
30 void getmax() {
31     int head = 0, tail = 0;
32     for (int i = 1; i <= n; i++) {
33         while (head <= tail && a[q[tail]] <= a[i]) tail--;
34         q[++tail] = i;
35     }
36     for (int i = k; i <= n; i++) {
37         while (head <= tail && a[q[tail]] <= a[i]) tail--;
38         q[++tail] = i;
39         while (q[head] <= i - k) head++;
40         cout << a[q[head]] << " ";
41     }
42     cout << endl;
43 }
44
45 int main() {
46     cin >> n >> k; //每k個連續的數
47     for (int i = 1; i <= n; i++) cin >> a[i];
48     getmin();
49     getmax();
50     return 0;
51 }

```

5 geometry

5.1 intersection

```

1 using LL = long long;
2

```

```

3 struct Point2D {
4     LL x, y;
5 };
6
7 struct Line2D {
8     Point2D s, e;
9     LL a, b, c; // L: ax + by = c
10     Line2D(Point2D s, Point2D e): s(s), e(e) {
11         a = e.y - s.y;
12         b = s.x - e.x;
13         c = a * s.x + b * s.y;
14     }
15 };
16
17 // 用克拉馬公式求二元一次解
18 Point2D intersection2D(Line2D l1, Line2D l2) {
19     {
20         LL D = l1.a * l2.b - l2.a * l1.b;
21         LL Dx = l1.c * l2.b - l2.c * l1.b;
22         LL Dy = l1.a * l2.c - l2.a * l1.c;
23     }
24     if (D) { // intersection
25         double x = 1.0 * Dx / D;
26         double y = 1.0 * Dy / D;
27     } else {
28         if (Dx || Dy) // Parallel lines
29             else // Same line
30     }
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, y+v.y);
13     }
14     Vector operator-(Vector v) {
15         return Vector(x-v.x, y-v.y);
16     }
17     Vector operator*(double val) {
18         return Vector(x*val, y*val);
19     }
20     double dot(Vector v) { return x*v.x + y*v.y; }
21     double cross(Vector v) { return x*v.y - y*v.x; }
22     double length() { return sqrt(dot(*this)); }
23     Vector unit_normal_vector() {
24         double len = length();
25         return Vector(-y/len, x/len);
26     }
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), v(v) {
36         ang = atan2(v.y, v.x);
37     }
38     bool operator< (const Line& l) const {
39         return ang < l.ang;

```



```

40 }
41 Point intersection(Line l) {
42     Vector u = p - l.p;
43     double t = l.v.cross(u) /
44         v.cross(l.v);
45     return p + v*t;
46 };
47
48 int n, m;
49 Line narrow[maxn]; // 要判斷的直線
50 Point poly[maxn]; //
    能形成半平面交的凸包邊界點
51
52 // return true if point p is on the left of
    line l
53 bool onLeft(Point p, Line l) {
54     return l.v.cross(p-l.p) > 0;
55 }
56
57 int halfplaneIntersection() {
58     int l, r;
59     Line L[maxn]; // 排序後的向量隊列
60     Point P[maxn]; // s[i] 跟 s[i-1]
        的交點
61
62     L[l=r=0] = narrow[0]; // notice: narrow
        is sorted
63     for(int i=1; i<n; i++) {
64         while(l<r && !onLeft(P[r-1],
            narrow[i])) r--;
65         while(l<r && !onLeft(P[l],
            narrow[i])) l++;
66
67         L[++r] = narrow[i];
68         if(l < r) P[r-1] =
            L[r-1].intersection(L[r]);
69     }
70
71     while(l<r && !onLeft(P[r-1], L[l])) r--;
72     if(r-l <= 1) return 0;
73
74     P[r] = L[r].intersection(L[l]);
75
76     int m=0;
77     for(int i=l; i<=r; i++) {
78         poly[m++] = P[i];
79     }
80
81     return m;
82 }
83
84 Point pt[maxn];
85 Vector vec[maxn];
86 Vector normal[maxn]; // normal[i] = vec[i]
        的單位法向量
87
88 double bsearch(double l=0.0, double r=1e4) {
89     if(abs(r-l) < eps) return l;
90
91     double mid = (l + r) / 2;
92
93     for(int i=0; i<n; i++) {
94         narrow[i] = Line(pt[i]+normal[i]*mid,
            vec[i]);
95     }
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 //Q: 平面上給定多個區域，由多個座標點所形成，再給定
2 //多點(x,y)，判斷有落點的區域(destroyed)的面積總和。
3 const int maxn = 500 + 10;
4 const int maxCoordinate = 500 + 10;
5 struct Point {
6     int x, y;
7 };
8 int n;
9 bool destroyed[maxn];
10 Point arr[maxn];
11 vector<Point> polygons[maxn];
12 void scanAndSortPoints() {
13     int minX = maxCoordinate, minY =
        maxCoordinate;
14     for(int i=0; i<n; i++) {
15         int x, y;
16         scanf("%d%d", &x, &y);
17         arr[i] = (Point){x, y};
18         if(y < minY || (y == minY && x <
            minX)) {
19             // If there are floating points, use:
20             // if(y<minY || (abs(y-minY)<eps &&
                x<minX)) {
21                 minX = x, minY = y;
22             }
23         }
24         sort(arr, arr+n, [minX, minY](Point& a,
            Point& b){
25             double theta1 = atan2(a.y - minY, a.x
                - minX);
26             double theta2 = atan2(b.y - minY, b.x
                - minX);
27             return theta1 < theta2;
28         });
29         return;
30 }
31
32 // returns cross product of u(AB) x v(AC)
33 int cross(Point& A, Point& B, Point& C) {
34     int u[2] = {B.x - A.x, B.y - A.y};
35     int v[2] = {C.x - A.x, C.y - A.y};
36     return (u[0] * v[1]) - (u[1] * v[0]);
37 }
38
39 // size of arr = n >= 3
40 // st = the stack using vector, m = index of
    the top
41 vector<Point> convex_hull() {
42     vector<Point> st(arr, arr+3);
43     for(int i=3, m=2; i<n; i++, m++) {
44         while(m >= 2) {
45             if(cross(st[m], st[m-1], arr[i])
                < 0)
46                 break;
47             st.pop_back();
48             m--;
49         }
50         st.push_back(arr[i]);
51     }
52     return st;
53 }
54
55 bool inPolygon(vector<Point>& vec, Point p) {

```

```

56     vec.push_back(vec[0]);
57     for(int i=1; i<vec.size(); i++) {
58         if(cross(vec[i-1], vec[i], p) < 0) {
59             vec.pop_back();
60             return false;
61         }
62     }
63     vec.pop_back();
64     return true;
65 }
66
67     1 | x1  x2  x3  x4  x5      xn |
68 A = -- | x   x   x   x   x ... x |
69     2 | y1  y2  y3  y4  y5      yn |
70 double calculateArea(vector<Point>& v) {
71     v.push_back(v[0]); // make v[n] = v[0]
72     double result = 0.0;
73     for(int i=1; i<v.size(); i++)
74         result +=
75             v[i-1].x*v[i].y - v[i-1].y*v[i].x;
76     v.pop_back();
77     return result / 2.0;
78 }
79
80 int main() {
81     int p = 0;
82     while(~scanf("%d", &n) && (n != -1)) {
83         scanAndSortPoints();
84         polygons[p++] = convex_hull();
85     }
86     int x, y;
87     double result = 0.0;
88     while(~scanf("%d%d", &x, &y))
89         for(int i=0; i<p; i++)
90             if(inPolygon(polygons[i],
                (Point){x, y}))
91                 destroyed[i] = true;
92     for(int i=0; i<p; i++)
93         if(destroyed[i])
94             result +=
95                 calculateArea(polygons[i]);
96     printf("%.2lf\n", result);
97     return 0;
98 }

```

6 DP

6.1 抽屜

```

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

```

6.2 Deque 最大差距

```

1 /*定義dp[l][r]是l ~ r時與先手最大差異值
2 轉移式: dp[l][r] = max{a[l] - solve(l + 1,
    r), a[r] - solve(l, r - 1)}
3 裡面用減的主要是因為求的是相減且會一直換手，
4 所以正負正負...*/
5 #define maxn 3005
6 bool vis[maxn][maxn];
7 long long dp[maxn][maxn];
8 long long a[maxn];
9 long long solve(int l, int r) {

```

6.6 樹 DP 有幾個 path 長度為 k

```

10 if (l > r) return 0;
11 if (vis[l][r]) return dp[l][r];
12 vis[l][r] = true;
13 long long res = a[l] - solve(l + 1, r);
14 res = max(res, a[r] - solve(l, r - 1));
15 return dp[l][r] = res;
16 }
17 int main() {
18     ...
19     printf("%lld\n", solve(1, n));
20 }

```

6.3 LCS 和 LIS

```

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

```

6.4 RangeDP

```

1 //區間dp
2 int dp[55][55];
3 // dp[i][j] -> [i,j] 切割區間中最小的 cost
4 int cuts[55];
5 int solve(int i, int j) {
6     if (dp[i][j] != -1)
7         return dp[i][j];
8     //代表沒有其他切法, 只能是 cuts[j] - cuts[i]
9     if (i == j - 1)
10        return dp[i][j] = 0;
11    int cost = 0x3f3f3f3f;
12    for (int m = i + 1; m < j; ++m) {
13        //枚舉區間中間切點
14        cost = min(cost, solve(i, m) +
15            solve(m, j) + cuts[j] - cuts[i]);
16    }
17    return dp[i][j] = cost;
18 }
19 int main() {
20     int l, n;
21     while (scanf("%d", &l) != EOF && l) {
22         scanf("%d", &n);
23         for (int i = 1; i <= n; ++i)
24             scanf("%d", &cuts[i]);
25         cuts[0] = 0;
26         cuts[n + 1] = 1;
27         memset(dp, -1, sizeof(dp));
28         printf("ans = %d.\n", solve(0, n+1));
29     }
30     return 0;
31 }

```

6.5 stringDP

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

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

Longest Palindromic Subsequence

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

```

1 #define maxn 50005
2 #define maxk 505
3 //dp[u][u的child且距離u長度k的數量]
4 long long dp[maxn][maxk];
5 vector<vector<int>> G;
6 int n, k;
7 long long res = 0;
8 void dfs(int u, int p) {
9     //u自己
10    dp[u][0] = 1;
11    for (int v: G[u]) {
12        if (v == p)
13            continue;
14        dfs(v, u);
15        for (int i = 1; i <= k; ++i) {
16            //子樹v距離i-1的等於對於u來說距離i的
17            dp[u][i] += dp[v][i-1];
18        }
19    }
20    //統計在u子樹中距離u為k的數量
21    res += dp[u][k];
22    long long cnt = 0;
23    for (int v: G[u]) {
24        if (v == p)
25            continue; //重點算法
26        for (int x = 0; x <= k-2; ++x) {
27            cnt +=
28                dp[v][x]*(dp[u][k-x-1]-dp[v][k-x-2]);
29        }
30    }
31    res += cnt / 2;
32 }
33 int main() {
34     ...
35     dfs(1, -1);
36     printf("%lld\n", res);
37     return 0;
38 }

```

6.7 TreeDP reroot

```

1 /*re-root dp on tree 0(n + n + n) -> 0(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])

```

```

29        this->subtreeSize[u] +=
30            this->dfs(v);
31        //自己
32        this->subtreeSize[u] += 1;
33        return this->subtreeSize[u];
34    }
35    //求res[0], 0到所有點的距離
36    int dfs2(int u, int dis) {
37        this->visited[u] = true;
38        int sum = 0;
39        for (int v: this->G[u])
40            if (!visited[v])
41                sum += this->dfs2(v, dis + 1);
42        //要加上自己的距離
43        return sum + dis;
44    }
45    //算出所有的res
46    void dfs3(int u, int n) {
47        this->visited[u] = true;
48        for (int v: this->G[u]) {
49            if (!visited[v]) {
50                this->res[v] = this->res[u] +
51                    n - 2 *
52                    this->subtreeSize[v];
53                this->dfs3(v, n);
54            }
55        }
56    }
57 }

```

6.8 WeightedLIS

```

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

```

```
44         temp =
45             B[i]+query(1,1,n,1,height[i]-1);
46     else
47         temp = B[i];
48     update(height[i], 1, 1, n, temp);
49     res = max(res, temp);
50 }
51 printf("%lld\n", res);
52 return 0;
53 }
```