

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	1
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	二分圖最大匹配	4
3.9	JosephusProblem	4
3.10	KM	4
3.11	LCA 倍增法	5
3.12	MCMF	5
3.13	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	6
4.4	Trie	7
4.5	單調隊列	7
5	geometry	8
5.1	intersection	8
5.2	半平面相交	8
5.3	凸包	9
6	DP	9
6.1	抽屜	9
6.2	Deque 最大差距	9
6.3	LCS 和 LIS	9
6.4	RangeDP	10
6.5	stringDP	10
6.6	TreeDP 有幾個 path 長度為 k	10
6.7	TreeDP reroot	10
6.8	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& 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 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<p.size(); ++i) {
27            if(p[i]*p[i] > n) break;
28            if(n % p[i]) continue;
29            cout << p[i] << ' ';
30            while(n%p[i] == 0) n /= p[i];
31        }
32        if(n != 1) cout << n << ' ';
33        cout << '\n';
34    }
35
36    擴展歐幾里得算法 ax + by = GCD(a, b)

```

```

37 int ext_euc(int a, int b, int &x, int &y) {
38     if(b == 0){
39         x = 1, y = 0;
40         return a;
41     }
42     int d = ext_euc(b, a%b, y, x);
43     y -= a/b*x;
44     return d;
45 }
46 int main(){
47     int a, b, x, y;
48     cin >> a >> b;
49     ext_euc(a, b, x, y);
50     cout << x << ' ' << y << endl;
51     return 0;
52 }
53
54 歌德巴赫猜想
55 解：把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
56 #define N 20000000
57 int ox[N], p[N], pr;
58 void PrimeTable(){
59     ox[0] = ox[1] = 1;
60     pr = 0;
61     for(int i=2;i<N;i++){
62         if(!ox[i]) p[pr++] = i;
63         for(int j=0; i*p[j]<N&&j<pr; j++)
64             ox[i*p[j]] = 1;
65     }
66 }
67
68 int main(){
69     PrimeTable();
70     int n;
71     while(cin>>n, n){
72         int x;
73         for(x=1;; x+=2)
74             if(!ox[x] && !ox[n-x]) break;
75         printf("%d = %d + %d\n", n, x, n-x);
76     }
77 }
78
79 problem :
80 給定整數 N，求 N 最少可以拆成多少個質數的和。
81 如果 N 是質數，則答案為 1。
82 如果 N 是偶數 (N!=2)，則答案為 2 (強歌德巴赫猜想)。
83 如果 N 是奇數且 N-2 是質數，則答案為 2 (2+質數)。
84 其他狀況答案為 3 (弱歌德巴赫猜想)。
85
86 bool isPrime(int n){
87     for(int i=2;i<n;i++){
88         if(i>n) return true;
89         if(n%i==0) return false;
90     }
91     return true;
92 }
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 }

```

2.3 歐拉函數

```

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

```

```

11 return ans;
12 }

```

3 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{
17         return Point(x+rhs.x, y+rhs.y, z+rhs.z);
18     }
19     Point operator-(const Point &rhs) const{
20         return Point(x-rhs.x, y-rhs.y, z-rhs.z);
21     }
22     Point operator*(const double &d) const{
23         return Point(x*d, y*d, z*d);
24     }
25     Point operator/(const double &d) const{
26         return Point(x/d, y/d, z/d);
27     }
28     double dist(const Point &rhs) const{
29         double res = 0;
30         res+=(x-rhs.x)*(x-rhs.x);
31         res+=(y-rhs.y)*(y-rhs.y);
32         res+=(z-rhs.z)*(z-rhs.z);
33         return res;
34     };
35 int main(){
36     IOS; //輸入優化
37     int T;
38     cin>>T;
39     for(int ti=1; ti<=T; ++ti){
40         double time;
41         Point x1, y1, d1, x2, y2, d2;
42         cin>>time>>x1>>y1>>x2>>y2;
43         d1=(y1-x1)/time;
44         d2=(y2-x2)/time;
45         double L=0, R=1e8, m1, m2, f1, f2;
46         double ans = x1.dist(x2);
47         while(abs(L-R)>1e-10){
48             m1=(L+R)/2;
49             m2=(m1+R)/2;
50             f1=((d1*m1)+x1).dist((d2*m1)+x2);
51             f2=((d1*m2)+x1).dist((d2*m2)+x2);
52             ans = min(ans, min(f1, f2));
53             if(f1<f2) R=m2;
54             else L=m1;
55         }
56         cout<<"Case "<<ti<<": ";
57         cout << fixed << setprecision(4) <<
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  $a[i] = b[0] + b[1] + b[2] + \dots + b[i]$ ，

```

```

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  $N(\leq 10^4)$ ，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 //problem
38 長度  $n$  的直線中有數個加熱器，
39 在  $x$  的加熱器可以讓  $[x-r, x+r]$  內的物品加熱，
40 問最少要幾個加熱器可以把  $[0, n]$  的範圍加熱。
41 //solution
42 對於最左邊沒加熱的點  $a$ ，選擇最遠可以加熱  $a$  的加熱器，
43 更新已加熱範圍，重複上述動作繼續尋找加熱器。
44 //code
45 int main(){
46     int n, r;
47     int a[1005];
48     cin>>n>>r;
49     for(int i=1; i<=n; ++i) cin>>a[i];
50     int i=1, ans=0;
51     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 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  $\max Li$  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     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 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
143 //problem
144 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？
145 //solution
146 和最少延遲數量問題是相同的問題，只要將題敘做轉換。
147 工作處理時長 → 烏龜重量
148 工作期限 → 烏龜可承受重量
149 多少工作不延期 → 可以疊幾隻烏龜
150 //code

```

```

151 struct Work{
152     int t, d;
153     bool operator<(const Work &rhs)const{
154         return d<rhs.d;
155     }
156 };
157 int main(){
158     int n=0;
159     Work a[10000];
160     priority_queue<int> pq;
161     while(cin>>a[n].t>>a[n].d)
162         ++n;
163     sort(a,a+n);
164     int sumT=0,ans=n;
165     for(int i=0;i<n;++i){
166         pq.push(a[i].t);
167         sumT+=a[i].t;
168         if(a[i].d<sumT){
169             int x=pq.top();
170             pq.pop();
171             sumT-=x;
172             --ans;
173         }
174     }
175     cout<<ans<<'\n';
176 }

```

```

177 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 Ti，
180 期限是 Di，如果第 i 項工作延遲需要受到 pi
181 單位懲罰，
182 請問最少會受到多少單位懲罰。
183 //solution
184 依照懲罰由大到小排序，
185 每項工作依序嘗試可不可以放在
186 Di-Ti+1, Di-Ti, ..., 1, 0，
187 如果有空間就放進去，否則延後執行。
188 //problem
189 給定 N 項工作，每項工作的需要處理時長為 Ti，
190 期限是 Di，如果第 i 項工作在期限內完成會獲得 ai
191 單位獎勵，
192 請問最多會獲得多少單位獎勵。
193 //solution
194 和上題相似，這題變成依照獎勵由大到小排序。
195 //code
196 struct Work{
197     int d,p;
198     bool operator<(const Work &rhs)const{
199         return p>rhs.p;
200     }
201 };
202 int main(){
203     int n;
204     Work a[100005];
205     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 }

```

3.4 dinic

```

1 const int maxn = 1e5 + 10;
2 const int inf = 0x3f3f3f3f;
3 struct Edge {
4     int s, t, cap, flow;
5 };
6 int n, m, S, T;
7 int level[maxn], dfs_idx[maxn];
8 vector<Edge> E;
9 vector<vector<int>> G;
10 void init() {
11     S = 0;
12     T = n + m;
13     E.clear();
14     G.assign(maxn, vector<int>());
15 }
16 void addEdge(int s, int t, int cap) {
17     E.push_back({s, t, cap, 0});
18     E.push_back({t, s, 0, 0});
19     G[s].push_back(E.size()-2);
20     G[t].push_back(E.size()-1);
21 }
22 bool bfs() {
23     queue<int> q({S});
24     memset(level, -1, sizeof(level));
25     level[S] = 0;
26     while(!q.empty()) {
27         int cur = q.front();
28         q.pop();
29         for(int i : G[cur]) {
30             Edge e = E[i];
31             if(level[e.t]==-1 &&
32                e.cap>e.flow) {
33                 level[e.t] = level[e.s] + 1;
34                 q.push(e.t);
35             }
36         }
37     }
38     return level[T];
39 }
40 int dfs(int cur, int lim) {
41     if(cur==T || lim==0) return lim;
42     int result = 0;
43     for(int& i=dfs_idx[cur]; i<G[cur].size()
44         && lim; i++) {
45         Edge& e = E[G[cur][i]];
46         if(level[e.s]+1 != level[e.t])
47             continue;
48         int flow = dfs(e.t, min(lim,
49             e.cap-e.flow));
50         if(flow <= 0) continue;
51         e.flow += flow;
52         result += flow;
53         E[G[cur][i]^1].flow -= flow;
54         lim -= flow;
55     }
56     return result;

```

```

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

```

```

66     G[u].emplace_back(v);
67 }
68 for (int i = 1; i <= n; ++i) {
69     if (dfn[i] == 0)
70         dfs(i);
71 }
72 printf("%lld %lld\n", totalCost, ways %
73     MOD);
74 return 0;

```

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] >=
21                 dfn[u])
22                 isCut = true;
23         }
24         else if (v != parent)
25             low[u] = min(low[u], dfn[v]);
26     }
27     //If u is root of DFS
28     //tree->有兩個以上的children
29     if (parent == -1 && child >= 2)
30         isCut = true;
31     if (isCut) ++res;
32 }
33 int main() {
34     char input[105];
35     char* token;
36     while (scanf("%d", &N) != EOF && N) {
37         G.assign(105, vector<int>());
38         memset(visited, false,
39             sizeof(visited));
40         memset(low, 0, sizeof(low));
41         memset(dfn, 0, sizeof(dfn));
42         timer = 0;
43         res = 0;
44         getchar(); // for \n
45         while (fgets(input, 105, stdin)) {
46             if (input[0] == '\0')
47                 break;
48             int size = strlen(input);
49             input[size - 1] = '\0';
50             --size;
51             token = strtok(input, " ");
52             int u = atoi(token);
53             int v;
54             while (token = strtok(NULL, " "))
55                 v = atoi(token);
56             G[u].emplace_back(v);
57             G[v].emplace_back(u);
58         }
59         tarjan(1, -1);
60         printf("%d\n", res);
61     }
62     return 0;

```

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                 if(cur == root) break;
41                 vis[cur] = i;
42                 cur = pre[cur];
43             }
44             if(cur!=root && idx[cur]==-1) {
45                 for(int j=pre[cur]; j!=cur;
46                     j=pre[j])
47                     idx[j] = seq;
48                 idx[cur] = seq++;
49             }
50         }
51         if(seq == 0) return result; // 沒有
52                             // cycle
53         for(int i=0; i<N; i++)
54             // 沒有被縮點的點
55             if(idx[i] == -1) idx[i] = seq++;
56         // 縮點並重新編號
57         for(Edge& e : edges) {
58             if(idx[e.s] != idx[e.t])
59                 e.cost -= inEdge[e.t];
60             e.s = idx[e.s];
61             e.t = idx[e.t];
62         }
63         N = seq;
64         root = idx[root];
65     }
66 }

```

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;

```


3.11 LCA 倍增法

```

19         return true;
20     }
21 }
22 }
23 return false;
24 }
25 //修改二分圖上的交錯路徑上點的權重
26 //此舉是在通過調整vertex labeling看看
27 //能不能產生出新的增廣路
28 //(KM的增廣路要求Lx[i] + Ly[j] == W[i][j])
29 //在這裡優先從最小的diff調看，才能保證最大權重匹配
30 void update()
31 {
32     int diff = 0x3f3f3f3f;
33     for (int i = 0; i < n; ++i) {
34         if (S[i]) {
35             for (int j = 0; j < n; ++j) {
36                 if (!T[j])
37                     diff = min(diff, Lx[i] +
38                               Ly[j] - W[i][j]);
39             }
40         }
41     }
42     for (int i = 0; i < n; ++i) {
43         if (S[i]) Lx[i] -= diff;
44         if (T[i]) Ly[i] += diff;
45     }
46 }
47 void KM()
48 {
49     for (int i = 0; i < n; ++i) {
50         L[i] = -1;
51         Lx[i] = Ly[i] = 0;
52         for (int j = 0; j < n; ++j)
53             Lx[i] = max(Lx[i], W[i][j]);
54     }
55     for (int i = 0; i < n; ++i) {
56         while(1) {
57             memset(S, false, sizeof(S));
58             memset(T, false, sizeof(T));
59             if (match(i))
60                 break;
61             else
62                 update(); //去調整vertex
63                             //labeling以增加增廣路徑
64         }
65     }
66 }
67 int main() {
68     while (scanf("%d", &n) != EOF) {
69         for (int i = 0; i < n; ++i)
70             for (int j = 0; j < n; ++j)
71                 scanf("%d", &W[i][j]);
72         KM();
73         int res = 0;
74         for (int i = 0; i < n; ++i) {
75             if (i != 0)
76                 printf(" %d", Lx[i]);
77             else
78                 printf("%d", Lx[i]);
79             res += Lx[i];
80         }
81         puts("");
82         for (int i = 0; i < n; ++i) {
83             if (i != 0)
84                 printf(" %d", Ly[i]);
85             else
86                 printf("%d", Ly[i]);
87             res += Ly[i];
88         }
89         puts("");
90         printf("%d\n", res);
91     }
92 }

```

```

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]) {
24         if (edge.v == p)
25             continue;
26         dis[edge.v][0] = edge.w;
27         dfs(edge.v, u);
28     }
29 }
30 long long lca(int x, int y) {
31     //此函數是找lca同時計算x、y的距離 -> dis(x,
32     //lca) + dis(lca, y)
33     //讓y比x深
34     if (dep[x] > dep[y])
35         swap(x, y);
36     int deltaDep = dep[y] - dep[x];
37     long long res = 0;
38     //讓y與x在同一個深度
39     for (int i = 0; deltaDep != 0; ++i,
40         deltaDep >>= 1)
41         if (deltaDep & 1)
42             res += dis[y][i], y = fa[y][i];
43     if (y == x) //x = y -> x、y彼此是彼此的祖先
44         return res;
45     //往上找，一起跳，但x、y不能重疊
46     for (int i = 30; i >= 0 && y != x; --i) {
47         if (fa[x][i] != fa[y][i]) {
48             res += dis[x][i] + dis[y][i];
49             x = fa[x][i];
50             y = fa[y][i];
51         }
52     }
53     //最後發現不能跳了，此時x的第2^0 =
54     //1個祖先(或說y的第2^0 =
55     //1的祖先)即為x、y的lca
56     res += dis[x][0] + dis[y][0];
57     return res;
58 }
59 int main() {
60     int n, q;
61     while (~scanf("%d", &n) && n) {
62         int v, w;
63         G.assign(n + 5, vector<Edge>());
64         for (int i = 1; i <= n - 1; ++i) {
65             scanf("%d %d", &v, &w);
66             G[i + 1].push_back({i + 1, v + 1, w});
67             G[v + 1].push_back({v + 1, i + 1, w});
68         }
69         dfs(1, 0);
70         scanf("%d", &q);
71         int u;
72         while (q--) {
73             scanf("%d %d", &u, &v);
74             printf("%lldc", lca(u + 1, v +
75                 1), (q) ? ' ': '\n');
76         }
77     }
78 }

```

```

71 }
72 return 0;
73 }

```

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});
18     edges.emplace_back(Edge{v, u, 0, 0,
19         -cost});
20     m = edges.size();
21     G[u].emplace_back(m - 2);
22     G[v].emplace_back(m - 1);
23 }
24 //一邊求最短路的同時一邊MaxFlow
25 bool SPFA(long long& maxFlow, long long&
26     minCost) {
27     // memset(outFlow, 0x3f,
28     // sizeof(outFlow));
29     memset(dis, 0x3f, sizeof(dis));
30     memset(inqueue, false, sizeof(inqueue));
31     queue<int> q;
32     q.push(s);
33     dis[s] = 0;
34     inqueue[s] = true;
35     outFlow[s] = INF;
36     while (!q.empty()) {
37         int u = q.front();
38         q.pop();
39         inqueue[u] = false;
40         for (const int edgeIndex: G[u]) {
41             const Edge& edge =
42                 edges[edgeIndex];
43             if ((edge.cap > edge.flow) &&
44                 (dis[edge.v] > dis[u] +
45                     edge.cost)) {
46                 dis[edge.v] = dis[u] +
47                     edge.cost;
48                 parent[edge.v] = edgeIndex;
49                 outFlow[edge.v] =
50                     min(outFlow[u], (long
51                         long)(edge.cap -
52                             edge.flow));
53                 if (!inqueue[edge.v]) {
54                     q.push(edge.v);
55                     inqueue[edge.v] = true;
56                 }
57             }
58         }
59     }
60     //如果dis[t] > 0代表根本不賺還倒賠
61     if (dis[t] > 0)
62         return false;
63     maxFlow += outFlow[t];
64     minCost += dis[t] * outFlow[t];
65     //一路更新回去這次最短路流完後要維護的
66     //MaxFlow演算法相關(如反向邊等)
67     int curr = t;
68     while (curr != s) {
69         edges[parent[curr]].flow +=
70             outFlow[t];
71     }
72 }

```

```

58     edges[parent[curr] ^ 1].flow -=
        outFlow[t];
59     curr = edges[parent[curr]].u;
60 }
61 return true;
62 }
63 long long MCMF() {
64     long long maxFlow = 0;
65     long long minCost = 0;
66     while (SPFA(maxFlow, minCost))
67         ;
68     return minCost;
69 }
70 int main() {
71     int T;
72     scanf("%d", &T);
73     for (int Case = 1; Case <= T; ++Case){
74         //總共幾個月，囤貨成本
75         int M, I;
76         scanf("%d %d", &M, &I);
77         //node size
78         n = M + M + 2;
79         G.assign(n + 5, vector<int>());
80         edges.clear();
81         s = 0;
82         t = M + M + 1;
83         for (int i = 1; i <= M; ++i) {
84             int produceCost, produceMax,
                sellPrice, sellMax,
                inventoryMonth;
85             scanf("%d %d %d %d %d",
                &produceCost, &produceMax,
                &sellPrice, &sellMax,
                &inventoryMonth);
86             addEdge(s, i, produceMax,
                produceCost);
87             addEdge(M + i, t, sellMax,
                -sellPrice);
88             for (int j = 0; j <=
                inventoryMonth; ++j) {
89                 if (i + j <= M)
90                     addEdge(i, M + i + j, INF,
                        I * j);
91             }
92         }
93         printf("Case %d: %lld\n", Case,
            -MCMF());
94     }
95     return 0;
96 }

```

3.13 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]=0]=c;
13        resSize = -1;
14        memset(rowHead, 0, sizeof(rowHead));
15        memset(colSize, 0, sizeof(colSize));
16    }
17    void insert(int r, int c) {
18        row[++seq]=r, col[seq]=c,
            ++colSize[c];
19        U[seq]=c, D[seq]=D[c], U[D[c]]=seq,
            D[c]=seq;
20        if(rowHead[r]) {
21            L[seq]=rowHead[r],
                R[seq]=R[rowHead[r]];

```

```

22        L[R[rowHead[r]]]=seq,
            R[rowHead[r]]=seq;
23    } else {
24        rowHead[r] = L[seq] = R[seq] =
            seq;
25    }
26    }
27    void remove(int c) {
28        L[R[c]] = L[c], R[L[c]] = R[c];
29        for(int i=D[c]; i!=c; i=D[i]) {
30            for(int j=R[i]; j!=i; j=R[j]) {
31                U[D[j]] = U[j];
32                D[U[j]] = D[j];
33                --colSize[col[j]];
34            }
35        }
36    }
37    void recover(int c) {
38        for(int i=U[c]; i!=c; i=U[i]) {
39            for(int j=L[i]; j!=i; j=L[j]) {
40                U[D[j]] = D[U[j]] = j;
41                ++colSize[col[j]];
42            }
43        }
44        L[R[c]] = R[L[c]] = c;
45    }
46    bool dfs(int idx=0) { // 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
        depth 版
68        if(R[0] == 0) {
69            resSize = min(resSize, idx); //
                注意init值
70            return;
71        }
72        int c = R[0];
73        for(int i=R[0]; i; i=R[i]) {
74            if(colSize[i] < colSize[c]) c = i;
75        }
76        remove(c);
77        for(int i=D[c]; i!=c; i=D[i]) {
78            for(int j=R[i]; j!=i; j=R[j])
79                remove(col[j]);
80            dfs(idx+1);
81            for(int j=L[i]; j!=i; j=L[j])
82                recover(col[j]);
83        }
84        recover(c);
85    }
86 };

```

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

```

4.2 線段樹 2D

```

1 //純2D segment tree 區間查詢單點修改最大最小值
2 #define maxn 2005 //500 * 4 + 5
3 int maxST[maxn][maxn], minST[maxn][maxn];
4 int N;
5 void modifyY(int index, int l, int r, int
    val, int yPos, int xIndex, bool
    xIsLeaf) {
6     if (l == r) {
7         if (xIsLeaf) {

```

```

8      maxST[xIndex][index] =
9          minST[xIndex][index] = val;
10     }
11     return;
12     maxST[xIndex][index] =
13         max(maxST[xIndex * 2][index],
14             maxST[xIndex * 2 + 1][index]);
15     minST[xIndex][index] =
16         min(minST[xIndex * 2][index],
17             minST[xIndex * 2 + 1][index]);
18 }
19 else {
20     int mid = (1 + r) / 2;
21     if (yPos <= mid)
22         modifyY(index * 2, 1, mid, val,
23                 yPos, xIndex, xIsLeaf);
24     else
25         modifyY(index * 2 + 1, mid + 1,
26                 r, val, yPos, xIndex,
27                 xIsLeaf);
28     maxST[xIndex][index] =
29         max(maxST[xIndex][index * 2],
30             maxST[xIndex][index * 2 + 1]);
31     minST[xIndex][index] =
32         min(minST[xIndex][index * 2],
33             minST[xIndex][index * 2 + 1]);
34 }
35 }
36 void modifyX(int index, int l, int r, int
37 val, int xPos, int yPos) {
38     if (l == r) {
39         modifyY(1, 1, N, val, yPos, index,
40                 true);
41     }
42     else {
43         int mid = (1 + r) / 2;
44         if (xPos <= mid)
45             modifyX(index * 2, 1, mid, val,
46                     xPos, yPos);
47         else
48             modifyX(index * 2 + 1, mid + 1,
49                     r, val, xPos, yPos);
50         modifyY(1, 1, N, val, yPos, index,
51                 false);
52     }
53 }
54 void queryY(int index, int l, int r, int
55 yql, int yqr, int xIndex, int& vmax,
56 int& vmin) {
57     if (yql <= l && r <= yqr) {
58         vmax = max(vmax,
59                     maxST[xIndex][index]);
60         vmin = min(vmin,
61                     minST[xIndex][index]);
62     }
63     else
64     {
65         int mid = (1 + r) / 2;
66         if (yql <= mid)
67             queryY(index * 2, 1, mid, yql,
68                     yqr, xIndex, vmax, vmin);
69         if (mid < yqr)
70             queryY(index * 2 + 1, mid + 1, r,
71                     yql, yqr, xIndex, vmax,
72                     vmin);
73     }
74 }
75 void queryX(int index, int l, int r, int
76 xql, int xqr, int yql, int yqr, int&
77 vmax, int& vmin) {
78     if (xql <= l && r <= xqr) {
79         queryY(1, 1, N, yql, yqr, index,
80                 vmax, vmin);
81     }
82     else {
83         int mid = (1 + r) / 2;
84         if (xql <= mid)
85             queryX(index * 2, 1, mid, xql,
86                     xqr, yql, yqr, int&
87                     vmax, int& vmin);
88         if (mid < xqr)
89             queryX(index * 2 + 1, mid + 1, r,
90                     xql, xqr, yql, yqr, int&
91                     vmax, int& vmin);
92     }
93 }
94 }
95 }

```

```

59     queryX(index * 2, 1, mid, xql,
60             xqr, yql, yqr, vmax, vmin);
61     if (mid < xqr)
62         queryX(index * 2 + 1, mid + 1, r,
63                 xql, xqr, yql, yqr, vmax,
64                 vmin);
65 }
66 }
67 int main() {
68     while (scanf("%d", &N) != EOF) {
69         int val;
70         for (int i = 1; i <= N; ++i) {
71             for (int j = 1; j <= N; ++j) {
72                 scanf("%d", &val);
73                 modifyX(1, 1, N, val, i, j);
74             }
75         }
76         int q;
77         int vmax, vmin;
78         int xql, xqr, yql, yqr;
79         char op;
80         scanf("%d", &q);
81         while (q--) {
82             getchar(); //for \n
83             scanf("%c", &op);
84             if (op == 'q') {
85                 scanf("%d %d %d %d", &xql,
86                     &yql, &xqr, &yqr);
87                 vmax = -0x3f3f3f3f;
88                 vmin = 0x3f3f3f3f;
89                 queryX(1, 1, N, xql, xqr,
90                     yql, yqr, vmax, vmin);
91                 printf("%d %d\n", vmax, vmin);
92             }
93             else {
94                 scanf("%d %d %d", &xql, &yql,
95                     &val);
96                 modifyX(1, 1, N, val, xql,
97                     yql);
98             }
99         }
100     }
101     return 0;
102 }

```

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 = (1 + r) / 2;
16     if (qx <= mid)
17         update(index * 2, 1, 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 = (1 + r) / 2;
28     //k比左子樹小
29     if (k <= st[index * 2])

```

```

29     return query(index * 2, 1, mid, k);
30 }
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)
56             sort(id + 1, id + m + 1);
57         int stSize = unique(id + 1, id + m +
58             1) - (id + 1);
59         for (int i = 1; i <= m; ++i) {
60             nums[i] = lower_bound(id + 1, id
61                 + stSize + 1, nums[i]) - id;
62         }
63         int addCount = 0;
64         int getCount = 0;
65         int k = 1;
66         while (getCount < n) {
67             if (getArr[getCount] == addCount)
68                 {
69                     printf("%d\n", query(1, 1,
70                         stSize, k));
71                     ++k;
72                     ++getCount;
73                 }
74             else {
75                 update(1, 1, stSize,
76                     nums[addCount + 1]);
77                 ++addCount;
78             }
79         }
80     }
81     return 0;
82 }

```

4.4 Trie

```

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

```

```

22     return;
23 }
24 int find(int idx, int len) {
25     int result = 0;
26     for(int r=0; idx<len; idx++) {
27         int c = str[idx] - 'a';
28         if(!(r = mp[r][c])) return result;
29         if(val[r])
30             result = (result + dp[idx +
31                             1]) % mod;
32     }
33     return result;
34 }
35 int main() {
36     int n, tc = 1;
37     while(~scanf("%s%d", str, &n)) {
38         Trie tr;
39         int len = strlen(str);
40         char word[100+10];
41         memset(dp, 0, sizeof(dp));
42         dp[len] = 1;
43         while(n--) {
44             scanf("%s", word);
45             tr.insert(word, strlen(word));
46         }
47         for(int i=len-1; i>=0; i--)
48             dp[i] = tr.find(i, len);
49         printf("Case %d: %d\n", tc++, dp[0]);
50     }
51     return 0;
52 }
53 ****Input****
54 * abcd
55 * 4
56 * a b c d 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。"
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     cout<<endl;
30 // 和上面同理
31 void getmax() {
32     int head=0, tail=0;
33     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    {
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):
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) {

```

```

18         return Vector(x*val, y*val);
19     }
20     double dot(Vector v) { return x*v.x +
21         y*v.y; }
22     double cross(Vector v) { return x*v.y -
23         y*v.x; }
24     double length() { return
25         sqrt(dot(*this)); }
26     Vector unit_normal_vector() {
27         double len = length();
28         return Vector(-y/len, x/len);
29     }
30 };
31 using Point = Vector;
32
33 struct Line {
34     Point p;
35     Vector v;
36     double ang;
37     Line(Point p={}, Vector v={}): p(p),
38         v(v) {
39         ang = atan2(v.y, v.x);
40     }
41     bool operator<(const Line& l) const {
42         return ang < l.ang;
43     }
44     Point intersection(Line l) {
45         Vector u = p - l.p;
46         double t = l.v.cross(u) /
47             v.cross(l.v);
48         return p + v*t;
49     }
50 };
51
52 // return true if point p is on the left of
53 // line l
54 bool onLeft(Point p, Line l) {
55     return l.v.cross(p-l.p) > 0;
56 }
57
58 int halfplaneIntersection() {
59     int l, r;
60     Line L[maxn]; // 排序後的向量隊列
61     Point P[maxn]; // s[i] 跟 s[i-1]
62     // 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=1; i<=r; i++) {
84         poly[m++] = P[i];
85     }
86     return m;
87 }

```



```

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,
95             vec[i]);
96     }
97
98     if(halfplaneIntersection())
99         return bsearch(mid, r);
100     else return bsearch(l, mid);
101 }
102
103 int main() {
104     while(~scanf("%d", &n) && n) {
105         for(int i=0; i<n; i++) {
106             double x, y;
107             scanf("%lf%lf", &x, &y);
108             pt[i] = {x, y};
109         }
110         for(int i=0; i<n; i++) {
111             vec[i] = pt[(i+1)%n] - pt[i];
112             normal[i] =
113                 vec[i].unit_normal_vector();
114         }
115         printf("%.6lf\n", bsearch());
116     }
117     return 0;
118 }

```

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 #include <bits/stdc++.h>
4 using namespace std;
5
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 =
20         maxCoordinate;
21     for(int i=0; i<n; i++) {
22         int x, y;
23         scanf("%d%d", &x, &y);
24         arr[i] = (Point){x, y};
25         if(y < minY || (y == minY && x <
26             minX)) {
27             // If there are floating points, use:
28             // if(y<minY || (abs(y-minY)<eps &&
29                 x<minX)) {
30                 minX = x, minY = y;
31             }
32         }
33     }
34     sort(arr, arr+n, [minX, minY](Point& a,
35         Point& b){
36         double theta1 = atan2(a.y - minY, a.x
37             - minX);
38     }

```

```

32     double theta2 = atan2(b.y - minY, b.x
33         - minX);
34     return theta1 < theta2;
35 });
36 return;
37
38 // returns cross product of u(AB) x v(AC)
39 int cross(Point& A, Point& B, Point& C) {
40     int u[2] = {B.x - A.x, B.y - A.y};
41     int v[2] = {C.x - A.x, C.y - A.y};
42     return (u[0] * v[1]) - (u[1] * v[0]);
43 }
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])
52                 < 0)
53                 break;
54             st.pop_back();
55             m--;
56         }
57         st.push_back(arr[i]);
58     }
59     return st;
60 }
61
62 bool inPolygon(vector<Point>& vec, Point p) {
63     vec.push_back(vec[0]);
64     for(int i=1; i<vec.size(); i++) {
65         if(cross(vec[i-1], vec[i], p) < 0) {
66             vec.pop_back();
67             return false;
68         }
69     }
70     vec.pop_back();
71     return true;
72 }
73
74 1 | x1 x2 x3 x4 x5 ... xn |
75 A = - | x x x x x ... x |
76 2 | y1 y2 y3 y4 y5 yn |
77 double calculateArea(vector<Point>& v) {
78     v.push_back(v[0]); // make v[n] =
79     v[0]
80     double result = 0.0;
81     for(int i=1; i<v.size(); i++)
82         result += v[i-1].x*v[i].y -
83             v[i].x*v[i-1].y;
84     v.pop_back();
85     return result / 2.0;
86 }
87
88 int main() {
89     int p = 0;
90     while(~scanf("%d", &n) && (n != -1)) {
91         scanAndSortPoints();
92         polygons[p++] = convex_hull();
93     }
94
95     int x, y;
96     double result = 0.0;
97     while(~scanf("%d%d", &x, &y)) {
98         for(int i=0; i<p; i++) {
99             if(inPolygon(polygons[i],
100                 (Point){x, y}))
101                 destroyed[i] = true;
102         }
103     }
104     for(int i=0; i<p; i++) {
105         if(destroyed[i])
106             result +=
107                 calculateArea(polygons[i]);
108     }

```

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 -
6     // 1個抽屜且1個安全且最上面L) + (底下n -
7     // 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 -
9     // 1][0][0];
10     for (int j = 1; j <= i; ++j) {
11         dp[i][j][0] = dp[i - 1][j + 1][1] +
12             dp[i - 1][j][0];
13         dp[i][j][1] = dp[i - 1][j - 1][1] +
14             dp[i - 1][j - 1][0];
15     }
16 } //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:
3 dp[l][r] = max{a[l] - solve(l + 1, r),
4     a[r] - solve(l, r - 1)}
5 裡面用減的主要是因為求的是相減且會一直換手，
6 所以正負正負...*/
7 #define maxn 3005
8 bool vis[maxn][maxn];
9 long long dp[maxn][maxn];
10 long long solve(int l, int r) {
11     if (l > r)
12         return 0;
13     if (vis[l][r])
14         return dp[l][r];
15     vis[l][r] = true;
16     long long res = a[l] - solve(l + 1, r);
17     res = max(res, a[r] - solve(l, r - 1));
18     return dp[l][r] = res;
19 }
20 int main() {
21     ...
22     printf("%lld\n", solve(1, n));
23 }

```

6.3 LCS 和 LIS

```

1 //最長共同子序列(LCS)
2 給定兩序列 A,B，求最長的序列 C，
3 C 同時為 A,B 的子序列。
4 //最長遞增子序列 (LIS)
5 給你一個序列 A，求最長的序列 B，
6 B 是一個 (非) 嚴格遞增序列，且為 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

```

6.4 RangeDP

```
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 }
```

6.5 stringDP

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

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

Longest Palindromic Subsequence

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

6.6 TreeDP 有幾個 path 長度為 k

```
1 #define maxn 50005
2 #define maxk 505
3 //dp[u][u的child且距離u長度k的數量]
4 long long dp[maxn][maxk];
5 vector<vector<int>> G;
6 int n, k;
7 long long res = 0;
8 void dfs(int u, int p) {
9     //u自己
10    dp[u][0] = 1;
11    for (int v: G[u]) {
12        if (v == p)
13            continue;
14        dfs(v, u);
15        for (int i = 1; i <= k; ++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    }
```

```
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 }
```

6.7 TreeDP reroot

```
1 /*re-root dp on tree  $O(n + n + n) \rightarrow O(n)*$ */
2 class Solution {
3 public:
4     vector<int> sumOfDistancesInTree(int n,
5         vector<vector<int>>& edges) {
6         this->res.assign(n, 0);
7         G.assign(n + 5, vector<int>());
8         for (vector<int>& edge: edges) {
9             G[edge[0]].emplace_back(edge[1]);
10            G[edge[1]].emplace_back(edge[0]);
11        }
12        memset(this->visited, 0,
13            sizeof(this->visited));
14        this->dfs(0);
15        memset(this->visited, 0,
16            sizeof(this->visited));
17        this->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     }
```

```
47 }
48 //算出所有的res
49 void dfs3(int u, int n) {
50     this->visited[u] = true;
51     for (int v: this->G[u]) {
52         if (!visited[v]) {
53             this->res[v] = this->res[u] +
54                 n - 2 *
55                 this->subtreeSize[v];
56             this->dfs3(v, n);
57         }
58     }
```

6.8 Weighted LIS

```
1 #define maxn 200005
2 long long dp[maxn];
3 long long height[maxn];
4 long long B[maxn];
5 long long st[maxn << 2];
6 void update(int p, int index, int 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 }
```