

# Contents

1	字串	
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	7
4.4	Trie	7
4.5	單調隊列	8
5	geometry	8
5.1	intersection	8
5.2	半平面相交	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	9
6.5	stringDP	10
6.6	樹 DP 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 p[MAXN];
5 int pSize=0;
6 void getPrimes(){
7     memset(isPrime,true,sizeof(isPrime));
8     isPrime[0]=isPrime[1]=false;
9     for(int i=2;i<MAXN;i++){
10        if(isPrime[i]) p[pSize++]=i;
11        for(int j=0;j<pSize&&i*p[j]<=MAXN;++j){
12            isPrime[i*p[j]]=false;
13            if(i%p[j]==0) break;
14        }
15    }
16 }
17
18 最大公因數 O(log(min(a,b)))
19 int GCD(int a, int b){
20     if(b == 0) return a;
21     return GCD(b, a%b);
22 }
23
24 質因數分解
25 void primeFactorization(int n){
26     for(int i=0; i<p.size(); ++i) {
27         if(p[i]*p[i] > n) break;
28         if(n % p[i]) continue;
29         cout << p[i] << ' ';
30         while(n%p[i] == 0) n /= p[i];
31     }
32     if(n != 1) cout << n << ' ';
33     cout << '\n';
34 }
35
36 擴展歐幾里得算法 ax + by = GCD(a, b)
37 int ext_euc(int a, int b, int &x, int &y) {

```

```

38     if(b == 0){
39         x = 1, y = 0;
40         return a;
41     }
42     int d = ext_euc(b, a%b, y, x);
43     y -= a/b*x;
44     return d;
45 }
46 int main(){
47     int a, b, x, y;
48     cin >> a >> b;
49     ext_euc(a, b, x, y);
50     cout << x << ' ' << y << endl;
51     return 0;
52 }
53
54 歌德巴赫猜想
55 解：把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
56 #define N 2000000
57 int ox[N], p[N], pr;
58 void PrimeTable(){
59     ox[0] = ox[1] = 1;
60     pr = 0;
61     for(int i=2;i<N;i++){
62         if(!ox[i]) p[pr++] = i;
63         for(int j=0; i*p[j]<N&&j<pr; j++)
64             ox[i*p[j]] = 1;
65     }
66 }
67
68 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*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*i<=n;i++){
6         if(n%i==0){
7             ans=ans-i;
8             while(n%i==0) n/=i;
9         }
10    }
11    if(n>1) ans=ans-ans/n;
12    return ans;

```

## 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):
        x(_x), y(_y), z(_z){}
10     friend istream& operator>>(istream& is,
        Point& p) {
11         is >> p.x >> p.y >> p.z;
12         return is;
13     }
14 }
15 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)<<
            sqrt(ans)<<"\n";
58     }
59 }

```

### 3.2 差分

```

1 用途：在區間  $[l, r]$  加上一個數字  $v$ 。
2  $b[l] += v$ ; ( $b[0 \sim l]$  加上  $v$ )
3  $b[r+1] -= v$ ; ( $b[r+1 \sim n]$  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  $N(≤10^4)$ ，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  $>Li$  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){
52         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         sort(a, a+n);
93         int ans=1, R=a[0].R;
94         for(int i=1; i<n; ++i){
95             if(a[i].L>=R){
96                 ++ans;
97                 R=a[i].R;
98             }
99         }
100         cout<<ans<<"\n";
101     }
102 }
103 最小化最大延遲問題
104 //problem
105 給定  $N$  項工作，每項工作的需要處理時長為  $Ti$ ，
106 期限是  $Di$ ，第  $i$  項工作延遲的時間為
     $Li=\max(0, Fi-Di)$ ，
107 原本  $Fi$  為第  $i$  項工作的完成時間，
108 求一種工作排序使  $\max Li$  最小。
109 //solution
110 按照到期時間從早到晚處理。
111 //code
112 struct Work{
113     int t, d;
114     bool operator<(const Work &rhs) const{
115         return d<rhs.d;
116     }
117 };
118 int main(){
119     int n;
120     Work a[10000];
121     cin>>n;
122     for(int i=0; i<n; ++i)
123         cin>>a[i].t>>a[i].d;
124     sort(a, a+n);
125     int maxL=0, sumT=0;
126     for(int i=0; i<n; ++i){
127         sumT+=a[i].t;
128         maxL=max(maxL, sumT-a[i].d);
129     }
130     cout<<maxL<<"\n";

```

```

131 }
132 最少延遲數量問題
133 //problem
134 給定 N 個工作，每個工作的需要處理時長為 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
178 任務調度問題
179 //problem
180 給定 N 項工作，每項工作的需要處理時長為 Ti，
181 期限是 Di，如果第 i 項工作延遲需要受到 pi
182 單位懲罰，
183 請問最少會受到多少單位懲罰。
184 //solution
185 依照懲罰由大到小排序，
186 每項工作依序嘗試可不可以放在
187 Di-Ti+1, Di-Ti, ..., 1, 0，
188 如果有空間就放進去，否則延後執行。
189 //problem
190 給定 N 項工作，每項工作的需要處理時長為 Ti，
191 期限是 Di，如果第 i 項工作在期限內完成會獲得 ai
192 單位獎勵，
193 請問最多會獲得多少單位獎勵。
194 //solution
195 和上題相似，這題變成依照獎勵由大到小排序。
196 //code
197 struct Work{
198     int d,p;
199     bool operator<(const Work &rhs)const{
200         return p>rhs.p;
201     }
202 };
203 int main(){
204     int n;
205     Work a[100005];
206     bitset<100005> ok;
207     while(cin>>n){

```

```

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;
57 }

```

```

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);

```

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

```

### 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] &&
            !T[j]) {
16             T[j] = true;
17             if ((L[j] == -1) || match(L[j])) {
18                 L[j] = i;
19                 return true;
20             }
21         }
22     }
23 }

```

### 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                 {
54                     v = atoi(token);
55                     G[u].emplace_back(v);
56                     G[v].emplace_back(u);
57                 }
58             tarjan(1, -1);
59             printf("%d\n", res);
60         }
61     }
62 }

```



```

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 }

```

### 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]) {
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("%lld\n", lca(u + 1, v +
75                 1), (q ? ' ': '\n'));
76         }
77     }
78     return 0;
79 }

```

### 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         edges[parent[curr] ^ 1].flow -=
72             outFlow[t];
73         curr = edges[parent[curr]].u;
74     }
75     return true;
76 }

```

```

63 long long MCMF() {
64     long long maxFlow = 0;
65     long long minCost = 0;
66     while (SPFA(maxFlow, minCost))
67         ;
68     return minCost;
69 }
70 int main() {
71     int T;
72     scanf("%d", &T);
73     for (int Case = 1; Case <= T; ++Case){
74         //總共幾個月，囤貨成本
75         int M, I;
76         scanf("%d %d", &M, &I);
77         //node size
78         n = M + M + 2;
79         G.assign(n + 5, vector<int>());
80         edges.clear();
81         s = 0;
82         t = M + M + 1;
83         for (int i = 1; i <= M; ++i) {
84             int produceCost, produceMax,
85                 sellPrice, sellMax,
86                 inventoryMonth;
87             scanf("%d %d %d %d %d",
88                 &produceCost, &produceMax,
89                 &sellPrice, &sellMax,
90                 &inventoryMonth);
91             addEdge(s, i, produceMax,
92                 produceCost);
93             addEdge(M + i, t, sellMax,
94                 -sellPrice);
95             for (int j = 0; j <=
96                 inventoryMonth; ++j) {
97                 if (i + j <= M)
98                     addEdge(i, M + i + j, INF,
99                         I * j);
100             }
101         }
102         printf("Case %d: %lld\n", Case,
103             -MCMF());
104     }
105     return 0;
106 }

```

### 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 }

```

```

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
68     //depth 版
69     if(R[0] == 0) {
70         resSize = min(resSize, idx); //
71         //注意init值
72         return;
73     }
74     int c = R[0];
75     for(int i=R[0]; i; i=R[i]) {
76         if(colSize[i] < colSize[c]) c = i;
77     }
78     remove(c);
79     for(int i=D[c]; i!=c; i=D[i]) {
80         for(int j=R[i]; j!=i; j=R[j])
81             remove(col[j]);
82         dfs(idx+1);
83         for(int j=L[i]; j!=i; j=L[j])
84             recover(col[j]);
85     }
86     recover(c);
87 }

```

## 4 DataStructure

### 4.1 線段樹 1D

```

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

```

```

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

```

### 4.2 線段樹 2D

```

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

```

```

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

```

```

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

```

### 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;

```

```

35     cin >> t;
36     bool first = true;
37     while (t-- > 0) {
38         if (first)
39             first = false;
40         else
41             puts("");
42         memset(st, 0, sizeof(st));
43         int m, n;
44         cin >> m >> n;
45         for (int i = 1; i <= m; ++i) {
46             cin >> nums[i];
47             id[i] = nums[i];
48         }
49         for (int i = 0; i < n; ++i)
50             cin >> getArr[i];
51         //離散化
52         //防止m == 0
53         if (m)
54             sort(id + 1, id + m + 1);
55         int stSize = unique(id + 1, id + m +
56                             1) - (id + 1);
57         for (int i = 1; i <= m; ++i) {
58             nums[i] = lower_bound(id + 1, id
59                                     + stSize + 1, nums[i]) - id;
60         }
61         int addCount = 0;
62         int getCount = 0;
63         int k = 1;
64         while (getCount < n) {
65             if (getArr[getCount] == addCount)
66             {
67                 printf("%d\n", query(1, 1,
68                                         stSize, k));
69                 ++k;
70                 ++getCount;
71             }
72             else {
73                 update(1, 1, stSize,
74                         nums[addCount + 1]);
75                 ++addCount;
76             }
77         }
78     }
79     return 0;
80 }

```

### 4.4 Trie

```

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

```

```

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

```

## 4.5 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以退役了。"
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<k;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 }
31 // 和上面同理
32 void getmax() {
33     int head=0,tail=0;
34     for(int i=1;i<k;i++) {
35         while(head<=tail&&a[q[tail]]<=a[i])tail--;
36         q[++tail]=i;
37     }
38     for(int i=k; i<=n;i++) {
39         while(head<=tail&&a[q[tail]]<=a[i])tail--;
40         q[++tail]=i;
41         while(q[head]<=i-k) head++;
42     }
43 }

```

```

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    // 用克拉克馬公式求二元一次解
18    Point2D intersection2D(Line2D l1, Line2D l2)
19    {
20        LL D = l1.a * l2.b - l2.a * l1.b;
21        LL Dx = l1.c * l2.b - l2.c * l1.b;
22        LL Dy = l1.a * l2.c - l2.a * l1.c;
23
24        if(D) { // intersection
25            double x = 1.0 * Dx / D;
26            double y = 1.0 * Dy / D;
27        } else {
28            if(Dx || Dy) // Parallel lines
29                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) {
19         return Vector(x*val, y*val);
20     }
21     double dot(Vector v) { return x*v.x +
22         y*v.y; }
23     double cross(Vector v) { return x*v.y -
24         y*v.x; }

```

```

22 double length() { return
23     sqrt(dot(*this)); }
24 Vector unit_normal_vector() {
25     double len = length();
26     return Vector(-y/len, x/len);
27 };
28
29 using Point = Vector;
30
31 struct Line {
32     Point p;
33     Vector v;
34     double ang;
35     Line(Point p={}, Vector v={}): p(p),
36         v(v) {
37         ang = atan2(v.y, v.x);
38     }
39     bool operator<(const Line& l) const {
40         return ang < l.ang;
41     }
42     Point intersection(Line l) {
43         Vector u = p - l.p;
44         double t = l.v.cross(u) /
45             v.cross(l.v);
46         return p + v*t;
47     }
48 };
49 int n, m;
50 Line narrow[maxn]; // 要判斷的直線
51 Point poly[maxn]; // 能形成半平面交的凸包邊界點
52 // 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     L[l=r=0] = narrow[0]; // notice: narrow
64     // is sorted
65     for(int i=1; i<n; i++) {
66         while(l<r && !onLeft(P[r-1],
67             narrow[i])) r--;
68         while(l<r && !onLeft(P[l],
69             narrow[i])) l++;
70
71         L[++r] = narrow[i];
72         if(l < r) P[r-1] =
73             L[r-1].intersection(L[r]);
74     }
75
76     while(l<r && !onLeft(P[r-1], L[l])) r--;
77     if(r-l <= 1) return 0;
78
79     P[r] = L[r].intersection(L[l]);
80
81     int m=0;
82     for(int i=l; i<=r; i++) {
83         poly[m++] = P[i];
84     }
85     return m;
86 }
87
88 Point pt[maxn];
89 Vector vec[maxn];
90 Vector normal[maxn]; // normal[i] = vec[i]
91 // 的單位法向量
92 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         double theta2 = atan2(b.y - minY, b.x
39             - minX);
40         return theta1 < theta2;
41     });
42     return;
43 }
44 }
45 }

```

```

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
47 // the top
48 vector<Point> convex_hull() {
49     vector<Point> st(arr, arr+3);
50     for(int i=3, m=2; i<n; i++, m++) {
51         while(m >= 2) {
52             if(cross(st[m], st[m-1], arr[i])
53                 < 0)
54                 break;
55             st.pop_back();
56             m--;
57         }
58         st.push_back(arr[i]);
59     }
60     return st;
61 }
62
63 bool inPolygon(vector<Point>& vec, Point p) {
64     vec.push_back(vec[0]);
65     for(int i=1; i<vec.size(); i++) {
66         if(cross(vec[i-1], vec[i], p) < 0) {
67             vec.pop_back();
68             return false;
69         }
70     }
71     vec.pop_back();
72     return true;
73 }
74
75 double calculateArea(vector<Point>& v) {
76     v.push_back(v[0]); // make v[n] =
77         v[0]
78     double result = 0.0;
79     for(int i=1; i<v.size(); i++)
80         result += v[i-1].x*v[i].y -
81             v[i-1].y*v[i].x;
82     v.pop_back();
83     return result / 2.0;
84 }
85
86 int main() {
87     int p = 0;
88     while(~scanf("%d", &n) && (n != -1)) {
89         scanAndSortPoints();
90         polygons[p++] = convex_hull();
91     }
92     int x, y;
93     double result = 0.0;
94     while(~scanf("%d%d", &x, &y)) {
95         for(int i=0; i<p; i++) {
96             if(inPolygon(polygons[i],
97                 (Point){x, y}))
98                 destroyed[i] = true;
99         }
100     }
101     for(int i=0; i<p; i++) {
102         if(destroyed[i])
103             result +=
104                 calculateArea(polygons[i]);
105     }
106     printf("%.2lf\n", result);
107     return 0;
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 a[maxn];
11 long long solve(int l, int r) {
12     if (l > r)
13         return 0;
14     if (vis[l][r])
15         return dp[l][r];
16     vis[l][r] = true;
17     long long res = a[l] - solve(l + 1, r);
18     res = max(res, a[r] - solve(l, r - 1));
19     return dp[l][r] = res;
20 }
21 int main() {
22     ...
23     printf("%lld\n", solve(1, n));
24 }

```

### 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) +
14            solve(m, j) + cuts[j] - cuts[i]);
15    }
16    return dp[i][j] = cost;
17 }
18 int main() {
19     int l;
20     int 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] = l;
27         memset(dp, -1, sizeof(dp));
28         printf("The minimum cutting is
29             %d.\n", solve(0, n + 1));
30     }
31     return 0;
32 }

```

## 6.5 stringDP

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

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

Longest Palindromic Subsequence

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

## 6.6 樹 DP 有幾個 path 長度為 k

```

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

```

```

21    res += dp[u][k];
22    long long cnt = 0;
23    for (int v: G[u]) {
24        if (v == p)
25            continue; //重點算法
26        for (int x = 0; x <= k - 2; ++x) {
27            cnt += dp[v][x] * (dp[u][k - x -
28                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])
33                 this->subtreeSize[u] +=
34                     this->dfs(v);
35         //自己
36         this->subtreeSize[u] += 1;
37         return this->subtreeSize[u];
38     }
39     //求res[0], 0到所有點的距離
40     int dfs2(int u, int dis) {
41         this->visited[u] = true;
42         int sum = 0;
43         for (int v: this->G[u])
44             if (!visited[v])
45                 sum += this->dfs2(v, dis + 1);
46         //要加上自己的距離
47         return sum + dis;
48     }
49     //算出所有的res
50     void dfs3(int u, int n) {

```

```

46     this->visited[u] = true;
47     for (int v: this->G[u]) {
48         if (!visited[v]) {
49             this->res[v] = this->res[u] +
50                 n - 2 *
51                 this->subtreeSize[v];
52             this->dfs3(v, n);
53         }
54     }
55 }

```

## 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 }

```