

1 Setup

1.1 Setup

```
1 compile.sh
2   #!/bin/bash
3   g++ $1 && ./a.out < ipt.in > opt.out
4
5 sudo chmod u+x compile.sh
6
7 vim ~/.bashrc
8   alias run='~/compile.sh'
```

1.2 template

```
1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3","unroll-loops")
3 #define IO cin.tie(0), ios::sync_with_stdio(0)
4 #define All(x) x.begin(), x.end()
5 #define endl "\n"
6 #define sort_unique(x) sort(All(x)); x.erase(unique(All(x)),
7   x.end());
8 #define REP(i,n) for(int i = 0; i < n; i++)
9 #define print1d(a) {for(auto v: a) cout << v << " "; cout <<
10   endl;}
11 #define eb emplace_back
12 #define pb push_back
13 #define ne nth_element
14 #define lb lower_bound
15 #define ub upper_bound
16 #define int long long
17 typedef pair<int, int> pii;
18 typedef vector<int> vi;
19 using namespace std;
20
21 #include <bits/extc++.h>
22 #include <ext/pb_ds/assoc_container.hpp>
23 #include <ext/pb_ds/tree_policy.hpp>
24 using namespace __gnu_pbds;
25 #define multiset tree<ll, null_type, less_equal<ll>,
26   rb_tree_tag, tree_order_statistics_node_update>
27 order_of_key(k) : nums strictly smaller than k
28 find_by_order(k): index from 0
29 /*
30 | Tag          | push | pop | join | modify |
31 | paring_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
32 | thin_heap_tag   | 0(lgN) | 0(lgN) | 慢 | 慢 |
33 | binomial_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
34 | rc_binomial_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
35 | binary_heap_tag | 0(1) | 0(lgN) | 慢 | 0(lgN) |
36 */
37 typedef __gnu_pbds::priority_queue<pair<int,ii>,less<pair<int
38   ,ii>>,rc_binomial_heap_tag> heap;
39
40 int32_t main(){
41   IO;
42   return 0;
43 }
```

2 IO

2.1 IO

```
1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3","unroll-loops")
3 #define IO cin.tie(0), ios::sync_with_stdio(0)
4 #define All(x) x.begin(), x.end()
5 #define sort_unique(x) sort(All(x)); x.erase(unique(All(x)),
6   x.end());
7 #define ne nth_element
8 using namespace std;
9 #include <bits/extc++.h>
10 #include <ext/pb_ds/assoc_container.hpp>
11 #include <ext/pb_ds/tree_policy.hpp>
12 using namespace __gnu_pbds;
13 #define multiset tree<ll, null_type, less_equal<ll>,
14   rb_tree_tag, tree_order_statistics_node_update>
15 order_of_key(k) : nums strictly smaller than k
16 find_by_order(k): index from 0
17 /*
18 | Tag          | push | pop | join | modify |
19 | paring_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
20 | thin_heap_tag   | 0(lgN) | 0(lgN) | 慢 | 慢 |
21 | binomial_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
22 | rc_binomial_heap_tag | 0(1) | 0(lgN) | 0(1) | 0(lgN) |
23 | binary_heap_tag | 0(1) | 0(lgN) | 慢 | 0(lgN) |
24 */
25 typedef __gnu_pbds::priority_queue<pair<int,ii>,less<pair<int
26   ,ii>>,rc_binomial_heap_tag> heap;
```

3 Data_Structure

3.1 線段樹

```
1 ll v[1e5] , seg[4e5+7] ;
2 void StructSEG(int l , int r , ll node){
3   if(l==r) seg[node] = v[r] ;
4   else{
5     int mid = (l+r) >> 1 , tmp = node << 1;
6     StructSEG(l,mid,tmp) ;
7     StructSEG(mid+1,r,tmp+1) ;
8     seg[node] = seg[tmp] + seg[tmp+1] ;
9   }
10 }
11 int query(int tL , int tR , int nL , int nR , int node){
12   int mid = (nL+nR)>>1 , ans = 0 ;
13   if(tL <= nL && nR <= tR) return seg[node] ;
14   if(tL <= mid) ans += query(tL,tR,nL ,mid,(node<<1) ) ;
15   if(tR > mid) ans += query(tL,tR,mid+1,nR ,(node<<1)+1);
16   return ans ;
17 }
18 void updateNode(int idx , int val , int l , int r , int node){
19   if(l == r) seg[node] = val , v[idx] = val ;
20   else{
21     int m = (l + r) >> 1 ;
22     int leftNode = (node << 1) ;
```

```
23   int rightNode = (node << 1) + 1 ;
24   if(idx <= m && idx >= l) updateNode(idx , val , l , m
25     , leftNode ) ;
26   else updateNode(idx , val , m+1,r , rightNode ) ;
27   seg[node] = seg[leftNode] + seg[rightNode] ;
28 }
29
30 int main(){
31   IO;
32   int n, q, l, r;
33   cin >> n >> q;
34   for(int i = 1 ; i <= n ; i++) cin >> v[i] ;
35   StructSEG(1,n,1) ;
36   while(q--){
37     cin >> l >> r ;
38     cout << query(1,r,1,n,1) << '\n' ;
39   }
40 }
```

3.2 區間修改線段樹

```
1 #define Ls(x) x << 1
2 #define Rs(x) x << 1 | 1
3 int N = 1e5+5;
4 vector<int> segTree(4*N), lazy(4*N) , arr(N);
5 void build(int l, int r, int idx = 1){
6   if(l == r){
7     segTree[idx] = arr[l];
8     return;
9   }
10   int mid = (l+r) >> 1;
11   build(l, mid, Ls(idx));
12   build(mid+1, r, Rs(idx));
13   segTree[idx] = segTree[Ls(idx)] + segTree[Rs(idx)];
14 }
15
16 void pushDown(int l, int r, int idx){
17   if(lazy[idx] == 0) return;
18   segTree[idx] += (r-l+1)*lazy[idx];
19   if(l != r) lazy[Ls(idx)] += lazy[idx], lazy[Rs(idx)] +=
20     lazy[idx];
21   lazy[idx] = 0;
22 }
23
24 void update(int l, int r, int ql, int qr, int val, int idx =
25   1){
26   pushDown(l, r, idx);
27   if(l > qr || r < ql) return;
28   if(l >= ql && r <= qr){
29     segTree[idx] += (r-l+1)*val;
30     if(l != r) lazy[Ls(idx)] += val, lazy[Rs(idx)] += val
31     ;
32     return;
33   }
34   int mid = (l+r) >> 1;
35   update(l, mid, ql, qr, val, Ls(idx));
36   update(mid+1, r, ql, qr, val, Rs(idx));
37   segTree[idx] = segTree[Ls(idx)] + segTree[Rs(idx)];
38 }
39
40 int query(int l, int r, int ql, int qr, int idx = 1){
41   pushDown(l, r, idx);
```

```

39 if(l > qr || r < ql) return 0;
40 if(l >= ql && r <= qr) return segTree[idx];
41 int mid = (l+r) >> 1, sum = 0;
42 if(ql <= mid) sum += query(l, mid, ql, qr, Ls(idx));
43 if(qr > mid) sum += query(mid+1, r, ql, qr, Rs(idx));
44 return sum;
45 }
46 int main(){
47     int n, t;
48     cin >> n >> t;
49     for(int i = 1; i <= n; i++) cin >> arr[i];
50     build(1,n,1);
51     for(int i = 0; i < t; i++){
52         int j, l, r, v;
53         cin >> j;
54         if(j == 1){
55             cin >> l >> r >> v;
56             update(1,n,l,r,v);
57         }else{
58             cin >> r;
59             cout << query(r,1,n) << endl;
60         }
61     }
62 }
63
64 // prefix sum
65 #pragma GCC optimize("O3","unroll-loops")
66 #include <bits/stdc++.h>
67 #define IO ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0)
68 #define endl '\n'
69 #define Ls(x) x << 1
70 #define Rs(x) (x << 1) | 1
71 using namespace std;
72 typedef long long ll;
73
74 int N = 2e5 + 5;
75 vector<ll> segTree(4 * N, -2e18), lazy(4 * N), arr(N), tmp(N);
76
77 void build(int l, int r, int idx = 1){
78     if(l == r){
79         segTree[idx] = arr[l];
80         return;
81     }
82     int mid = (l+r) >> 1;
83     build(l, mid, Ls(idx));
84     build(mid+1, r, Rs(idx));
85     segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
86 }
87
88 void pushDown(int l, int r, int idx, int x = 0){
89     if(lazy[idx] == 0) return;
90     segTree[idx] += lazy[idx];
91     int mid = (l + r) / 2;
92     if(l != r) lazy[Ls(idx)] += lazy[idx], lazy[Rs(idx)] += lazy[idx];
93     lazy[idx] = 0;
94 }
95
96 void update(int l, int r, int ql, int qr, ll val, int idx = 1){
97     pushDown(l, r, idx);

```

```

101 if(l > qr || r < ql) return;
102 if(l >= ql && r <= qr){
103     /* segTree[idx] += val; */
104     lazy[idx] += val;
105     pushDown(l, r, idx);
106     return;
107 }
108 int mid = (l+r) >> 1;
109 update(l, mid, ql, qr, val, Ls(idx));
110 update(mid+1, r, ql, qr, val, Rs(idx));
111 segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
112 }
113
114 ll query(int l, int r, int ql, int qr, int idx = 1){
115     pushDown(l, r, idx);
116     if(l > qr || r < ql) return -2e18;
117     if(l >= ql && r <= qr) return segTree[idx];
118     ll mid = (l+r) >> 1, ans = -2e18;
119     ans = max(ans, query(l, mid, ql, qr, Ls(idx)));
120     ans = max(ans, query(mid+1, r, ql, qr, Rs(idx)));
121     segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
122     return ans;
123 }
124
125 int main(){
126     IO;
127     int n, q;
128     cin >> n >> q;
129     for(int i = 1; i <= n; i++) cin >> tmp[i];
130     arr = tmp;
131     for(int i = 2; i <= n; i++) arr[i] += arr[i-1];
132     build(1, n, 1);
133
134     while(q--){
135         int j, k, u;
136         cin >> j >> k >> u;
137         if(j == 1) update(1, n, k, n, u - tmp[k]), tmp[k] = u;
138         else cout << max(0ll, query(1, n, k, u) - (k!=1?query(1, n, k-1, k-1):0)) << endl;
139     }
140 }
141

```

3.3 Kurskal

```

1 struct Edge {
2     int src;
3     int dest;
4     int weight;
5     Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
6 };
7 int findParent(const vector<int>& parent, int i) {
8     if (parent[i] == i) return i;
9     return findParent(parent, parent[i]);
10 }
11 vector<Edge> kruskalMST(const vector<vector<int>>& adjacencyMatrix) {
12     int size = adjacencyMatrix.size();
13     vector<Edge> mst, edges;
14     for (int i = 0; i < size; ++i) {
15         for (int j = i + 1; j < size; ++j) {

```

```

16         if (adjacencyMatrix[i][j] > 0) edges.push_back(
17             Edge(i, j, adjacencyMatrix[i][j]));
18     }
19     sort(edges.begin(), edges.end(), [](const Edge& e1, const Edge& e2) {
20         return e1.weight < e2.weight;
21     });
22     vector<int> parent(size);
23     for (int i = 0; i < size; ++i) parent[i] = i;
24     int count = 0;
25     for (const Edge& edge : edges) {
26         if (count == size - 1) break;
27         int srcParent = findParent(parent, edge.src),
28             destParent = findParent(parent, edge.dest);
29         if (srcParent != destParent) {
30             mst.push_back(edge);
31             ++count;
32             parent[srcParent] = destParent;
33         }
34     }
35     return mst;
36 }

```

3.4 Prim

```

1 vector<Edge> primMST(const vector<vector<int>>& adjacencyMatrix) {
2     int size = adjacencyMatrix.size();
3     vector<bool> visited(size, false);
4     vector<Edge> mst;
5     priority_queue<Edge, vector<Edge>, function<bool(Edge, Edge)>> pq(
6         [](const Edge& e1, const Edge& e2) { return e1.weight > e2.weight; }
7     );
8     visited[0] = true;
9     for (int i = 1; i < size; ++i) {
10         if (adjacencyMatrix[0][i] > 0) {
11             pq.push(Edge(0, i, adjacencyMatrix[0][i]));
12         }
13     }
14     while (!pq.empty()) {
15         Edge minEdge = pq.top();
16         pq.pop();
17         int u = minEdge.dest;
18         if (visited[u]) continue;
19         visited[u] = true;
20         mst.push_back(minEdge);
21         for (int v = 0; v < size; ++v) {
22             if (adjacencyMatrix[u][v] > 0 && !visited[v]) pq.push(Edge(u, v, adjacencyMatrix[u][v]));
23         }
24     }
25     return mst;
26 }

```

3.5 DAG

```

1 void DFS(map<int,queue<int>> &graph , map<int,bool> &visited
    , int place , stack<int> &ans){
2     if(visited[place] == true) return;
3     visited[place]=true;
4     while(!graph[place].empty())
5         DFS(graph, visited, graph[place].front() , ans) ,
            graph[place].pop() ;
6     ans.push(place);
7 }
8 for(auto&n:graph) DFS(graph , visited , n.first , ans) ;
9 while(!ans.empty()) cout << ans.top() << " " , ans.pop();

```

3.6 Unionfind

```

1 int Find(int*a,int n){
2     if(a[n]==n)return n;
3     return a[n]=Find(a,a[n]);
4 }
5 void Union(int*a,int n,int m){
6     a[Find(a,n)]=Find(a,m);
7 }

```

3.7 CDQ

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define fastio ios_base::sync_with_stdio(false); cin.tie(0);
5 #define endl '\n'
6 #define _ << ' ' <<
7
8 const int INF = 2e9;
9
10 struct info {
11     int t, p;
12     long long x;
13     // 當 x 界在 -1e9 ~ 1e9 , 代表這是修改
14     // 當 x >= INF , 代表這是詢問
15     // 並且詢問的編號為 (x - INF)
16     // 例：第 2 次詢問，那 x 就設為 INF + 2
17     // 當 x <= -INF , 一樣代表詢問，只不過是倒扣的詢問
18 };
19
20 int n, q;
21 vector<info> v;
22 vector<long long> ans;
23
24 void cdq(int l, int r) {
25     if (l == r) return;
26
27     int mid = (l+r) / 2; cdq(l, mid); cdq(mid+1, r);
28
29     vector<info> temp;
30     long long add_sum = 0;
31     int pl = l, pr = mid+1;
32
33     while (pl <= mid && pr <= r) {
34         if (v[pl].p <= v[pr].p) {
35             temp.emplace_back(v[pl]);

```

```

36         if (abs(v[pl].x) <= 1e9) {
37             add_sum += v[pl].x;
38         }
39         pl++;
40     }
41     else {
42         temp.emplace_back(v[pr]);
43         if (v[pr].x >= INF) {
44             ans[v[pr].x - INF] += add_sum;
45         } else if (v[pr].x <= -INF) {
46             ans[-(v[pr].x + INF)] -= add_sum;
47         }
48         pr++;
49     }
50 }
51 while (pl <= mid) {
52     temp.emplace_back(v[pl]);
53     // 小小優化，註解掉的這段一樣可以不寫
54     // if (abs(v[pl].x) <= 1e9) {
55     //     add_sum += v[pl].x;
56     // }
57     pl++;
58 }
59 while (pr <= r) {
60     temp.emplace_back(v[pr]);
61     if (v[pr].x >= INF) {
62         ans[v[pr].x - INF] += add_sum;
63     } else if (v[pr].x <= -INF) {
64         ans[-(v[pr].x + INF)] -= add_sum;
65     }
66     pr++;
67 }
68
69 for (int i = l, j = 0; i <= r; i++, j++) {
70     v[i] = temp[j];
71 }
72 }
73
74 int main() {
75     fastio
76
77     cin >> n >> q;
78     v.clear(); ans.clear();
79
80     int time = 0;
81     vector<long long> arr(n+1);
82
83     for (int p = 1; p <= n; p++) {
84         long long x; cin >> x;
85         v.emplace_back(info{++time, p, x});
86         arr[p] = x;
87     }
88
89     int ask_id = 0;
90     for (int i = 0; i < q; i++) {
91         int op; cin >> op;
92         if (op == 1) {
93             int p; long long x;
94             cin >> p >> x;
95             v.emplace_back(info{++time, p, x - arr[p]});
96             arr[p] = x;
97         }
98         else {
99             int l, r; cin >> l >> r;

```

```

101         v.emplace_back(info{++time, r, INF + ask_id});
102         if (l > r) v.emplace_back(info{time, l-1, -INF -
            ask_id});
103         ask_id++;
104         ans.emplace_back(0);
105     }
106 }
107
108 cdq(0, (int)v.size()-1);
109
110 for (int i = 0; i < (int)ans.size(); i++)
111     cout << ans[i] << endl;
112
113 return 0;
114 }
115
116 struct info {
117     int x, y, id; // id 是數對在原本陣列的位置
118 };
119 int N;
120 vector<info> v;
121 vector<int> ans;
122
123 void cdq(int L, int R) {
124     if (L == R) return;
125
126     int mid = (L + R) / 2;
127     cdq(L, mid);
128     cdq(mid + 1, R);
129
130     vector<info> temp;
131     int pl = L, pr = mid + 1;
132
133     while (pl <= mid && pr <= R) {
134         if (v[pl].y < v[pr].y) {
135             temp.emplace_back(v[pl]);
136             pl++;
137         } else {
138             temp.emplace_back(v[pr]);
139             ans[v[pr].id] += (pl - L);
140             pr++;
141         }
142     }
143
144     while (pl <= mid) {
145         temp.emplace_back(v[pl]);
146         pl++;
147     }
148     while (pr <= R) {
149         temp.emplace_back(v[pr]);
150         ans[v[pr].id] += (pl - L);
151         pr++;
152     }
153
154     for (int pt = 0, pv = L; pv <= R; pt++, pv++) {
155         v[pv] = temp[pt];
156     }
157 }
158
159 int main() {
160
161     // 把所有數對按照 x 排序;
162     cdq(0, N - 1);
163     // 輸出答案;
164

```

```

165     return 0;
166 }
167
168 struct info {
169     int x, y, z, id; // id 是數對在原本陣列的位置
170 };
171 int N;
172 vector<info> v;
173 vector<int> ans;
174
175 void cdq(int L, int R) {
176     if (L == R) return;
177     int mid = (L + R) / 2;
178     cdq(L, mid);
179     cdq(mid + 1, R);
180
181     vector<info> temp;
182     vector<pair<int, int>> BIT_op; // BIT_op: 紀錄你在 BIT
183         的操作, 包含修改位置、修改的值
184
185     int pl = L, pr = mid + 1;
186     while (pl <= mid && pr <= R) {
187         if (v[pl].y < v[pr].y) {
188             temp.emplace_back(v[pl]);
189             BIT.upd(v[pl].z, 1);
190             BIT_op.emplace_back(make_pair(v[pl].z, 1));
191             pl++;
192         } else {
193             temp.emplace_back(v[pr]);
194             ans[v[pr].id] += BIT.qry(v[pr].z - 1);
195             pr++;
196         }
197     }
198
199     while (pl <= mid) {
200         temp.emplace_back(v[pl]);
201         pl++;
202     }
203     while (pr <= R) {
204         temp.emplace_back(v[pr]);
205         ans[v[pr].id] += BIT.qry(v[pr].z - 1);
206         pr++;
207     }
208
209     for (int pt = 0, pv = L; pv <= R; pt++, pv++) {
210         v[pv] = temp[pt];
211     }
212
213     // 撤銷所有操作, 讓 BIT 變回初始狀態
214     for (auto& op : BIT_op) {
215         BIT.upd(op.first, -op.second);
216     }
217 }
218
219 int main() {
220     // 把所有數對按照 x 排序;
221     cdq(0, N - 1);
222     // 輸出答案;
223
224     return 0;
225 }

```

3.8 DSU

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class UnionFind {
5 private:
6     vector<int> parent;
7     vector<int> rank;
8
9 public:
10     UnionFind(int size) : parent(size), rank(size, 1) {
11         for (int i = 0; i < size; ++i) {
12             parent[i] = i;
13         }
14     }
15
16     int getSize() const {
17         return parent.size();
18     }
19
20     int find(int p) {
21         if (p < 0 || p >= parent.size())
22             throw out_of_range("Out of bound.");
23
24         while (p != parent[p]) {
25             parent[p] = parent[parent[p]];
26             p = parent[p];
27         }
28         return p;
29     }
30
31     int findR(int p) {
32         if (p < 0 || p >= parent.size())
33             throw out_of_range("Out of bound.");
34
35         if (p != parent[p])
36             parent[p] = findR(parent[p]);
37         return parent[p];
38     }
39
40     bool isConnected(int p, int q) {
41         return find(p) == find(q);
42     }
43
44     void unionElements(int p, int q) {
45         int pRoot = find(p);
46         int qRoot = find(q);
47
48         if (pRoot == qRoot)
49             return;
50
51         if (rank[pRoot] < rank[qRoot]) {
52             parent[pRoot] = qRoot;
53         } else if (rank[qRoot] < rank[pRoot]) {
54             parent[qRoot] = pRoot;
55         } else {
56             parent[pRoot] = qRoot;
57             rank[qRoot]++;
58         }
59     }
60
61     int countConnectedNodes(int p) {
62         int root = find(p);
63         int count = 0;
64         for (int i = 0; i < parent.size(); i++) {

```

```

64         if (find(i) == root) {
65             count++;
66         }
67     }
68     return count;
69 }
70 };
71
72 int main() {
73     UnionFind uf(10);
74
75     uf.unionElements(1, 2);
76     uf.unionElements(2, 3);
77     uf.unionElements(4, 5);
78     uf.unionElements(3, 4);
79
80     cout << "Is 1 connected to 3? " << (uf.isConnected(1, 3)
81         ? "Yes" : "No") << endl;
82     cout << "Is 1 connected to 5? " << (uf.isConnected(1, 5)
83         ? "Yes" : "No") << endl;
84     cout << "Number of nodes connected to 3: " << uf.
85         countConnectedNodes(4) << endl;
86
87     return 0;
88 }

```

4 DP

4.1 LIS

```

1 void LIS(vector<int> &arr){
2     vector<int> lis;
3     for (int i = 0; i < arr.size(); i++){
4         auto it = lower_bound(lis.begin(), lis.end(), arr[i]);
5         if (it == lis.end()) lis.push_back(arr[i]);
6         else *it = arr[i];
7     }
8     cout << lis.size() << endl;
9 }

```

4.2 LCS2LIS

```

1 int LCS2LIS(string &a, string &b){
2     vector<int> List;
3     for (int i = 0; i < b.size(); i++)
4         for (int j = 0; j < a.size(); j++)
5             if (b[i] == a[j]) List.push_back(j);
6     if (List.size() == 0) return 0;
7     vector<int> dp(List.size(), 1);
8     for (int i = 1; i < List.size(); i++){
9         if (List[i] > dp.back()) dp.push_back(List[i]);
10        else *lower_bound(dp.begin(), dp.end(), List[i]) =
11            List[i];
12    }
13    return dp.size();
14 }

```

4.3 LCS3D

```

1 int main(){
2     string x,y,z ; cin >> x >> y >> z ;
3     cout << lcsOf3(x,y,z , x.size() , y.size() , z.size()) <<
4         endl;
5 }
6 int lcsOf3( string X, string Y, string Z, int m, int n, int o
7 ){
8     int L[m+1][n+1][o+1];
9     for (int i=0; i<=m; i++){
10         for (int j=0; j<=n; j++){
11             for (int k=0; k<=o; k++){
12                 if (i == 0 || j == 0 || k==0) L[i][j][k] = 0;
13                 else if (X[i-1] == Y[j-1] && X[i-1]==Z[k-1])
14                     L[i][j][k] = L[i-1][j-1][k-1] + 1;
15                 else L[i][j][k] = max(max(L[i-1][j][k],L[i][j-1][k]),L[i][j][k-1]);
16             }
17         }
18     }
19     return L[m][n][o];
20 }

```

4.4 LCS

```

1 若一樣：左上+1
2 else max(左,上)

```

4.5 countingTours

```

1 #include <bits/stdc++.h>
2 #define IO cin.tie(0), ios::sync_with_stdio(0)
3 #define All(x) x.begin(), x.end()
4 #define sort_unique(x) sort(All(x)); x.erase(unique(All(x)),
5     x.end());
6 #define ne nth_element
7 #define INF 1e9
8 #define MOD 1000000007
9 #define Q 100000
10 typedef long long ll;
11 using namespace std;
12
13 vector<ll> a(Q+1), b(Q+1);
14
15 int main(){
16     IO;
17     ll t;
18     cin >> t;
19     a[0] = b[0] = 1;
20
21     for(ll i = 1; i <= Q; i++){
22         a[i] = a[i-1] + b[i-1] + a[i-1];
23         b[i] = a[i-1] + b[i-1] + b[i-1]*3;
24         a[i] %= MOD;
25     }

```

```

26     b[i] %= MOD;
27 }
28 while(t--){
29     ll q;
30     cin >> q;
31     cout << (a[q-1] + b[q-1])%MOD << endl;
32 }
33 }

```

4.6 retCutting

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 int dp[505][505];
6
7 ll f(int x, int y){
8     if(x==y) return 0 ;
9     if(dp[x][y] != 0) return dp[x][y];
10     ll ans = 2e18;
11     for(int i = 1; i < x; i++){
12         ans = min(ans, f(i,y) + f(x-i,y) + 1);
13     }
14     for(int i = 1; i < y; i++){
15         ans = min(ans, f(i,x) + f(y-i,x) + 1);
16     }
17     return dp[x][y] = ans;
18 }
19
20 int main(){
21
22     int x,y;
23     cin >> x >> y;
24     cout << f(x,y);
25 }

```

5 Graph

5.1 Dijkstra

```

1 /** 問某點到所有圖上的點的最短距離。0/1-based 都安全。 edge
2     要
3     * 是 {cost, dest} 格式。回傳的陣列若含有 -1 表示 src 到該位
4     置
5     * 不連通 **/
6 typedef pair<ll, int> pii;
7 vector<ll> dijkstra(int src, vector<vector<pii>&> edge) {
8     vector<ll> sum(edge.size(), -1);
9     priority_queue<pii, vector<pii>, greater<pii>> q;
10     q.emplace(0, src);
11     while (q.size()) {
12         int v = q.top().second; ll d = q.top().first;
13         q.pop();
14         if (sum[v] != -1) continue;
15         sum[v] = d;
16         for (auto& e : edge[v])

```

```

15         if (sum[e.second] == -1)
16             q.emplace(d + e.first, e.second);
17     } return sum;
18 }
19 int main(){
20     int n,m,s; cin>>n>>m>>s;
21     vector<vector<pii>> G(n+1);
22     while(m--){
23         int u,v,w; cin>>u>>v>>w;
24         G[u].emplace_back(w,v);
25     }
26     vector<ll> dis = dijkstra(s,G);
27     for(int i=1; i<=n; i++) cout<<dis[i]<<" \n"[i==n];
28 }

```

5.2 BellmanFord

```

1 typedef pair<int,int> pii;
2 const int maxn = 1e5+5;
3 const int INF = 0x3f3f3f3f;
4 vector<pii> G[maxn];
5 int dis[maxn];
6 bool BellmanFord(int n,int s) {
7     for(int i=1; i<=n; i++) dis[i] = INF;
8     dis[s] = 0;
9     bool relax;
10     for(int r=1; r<=n; r++) { //O(VE)
11         relax = false;
12         for(int i=1; i<=n; i++){
13             for(pii e:G[i])
14                 if( dis[i] + e.second < dis[e.first] )
15                     dis[e.first] = dis[i] + e.second, relax =
16                         true;
17         }
18         return relax; //有負環
19 }
20 int main(){
21     int n,m,s; cin>>n>>m>>s;
22     while(m--){
23         int u,v,w; cin>>u>>v>>w;
24         G[u].emplace_back(w,v);
25     }
26     if(BellmanFord(n,s)) cout<<"negative cycle\n";
27     else for(int i=1; i<=n; i++) cout<<dis[i]<<" \n"[i==n];
28 }

```

5.3 SPFA

```

1 typedef pair<int,int> pii;
2 const int maxn = 1e5+5;
3 const int INF = 0x3f3f3f3f;
4 vector<pii> G[maxn]; int dis[maxn];
5 void SPFA(int n,int s) {
6     for(int i=1; i<=n; i++) dis[i] = INF;
7     dis[s] = 0;
8     queue<int> q; q.push(s);
9     bool inque[maxn] = {};
10     while(!q.empty()) {
11         int u = q.front(); q.pop();

```

```

12     inque[u] = false;
13     for(pii e:G[u]) {
14         int v = e.first , w = e.second;
15         if( dis[u] + w < dis[v]) {
16             if(!inque[v]) q.push(v), inque[v] = true;
17             dis[v] = dis[u] + w;
18         }
19     }
20 }
21 }
22 int main(){
23     int n,m,s; cin>>n>>m>>s;
24     while(m--){
25         int u,v,w; cin>>u>>v>>w;
26         G[u].emplace_back(v,w);
27     }
28     SPFA(n,s);
29     for(int i=1; i<=n; i++) cout<<dis[i]<<"\n"[i==n];
30 }

```

5.4 MaxFlow

```

1 struct edge{
2     int from, to, cap, flow;
3     edge(int u, int v, int c, int f) : from(u), to(v), cap(c)
4     , flow(f) {}
5 };
6 struct Dinic{
7     vector<edge> edges;
8     vector<vector<int>>> adj;
9     vector<int> level, ptr;
10    int n, m = 0, s, t;
11    Dinic(int n, int s, int t) : n(n), s(s), t(t){
12        adj.resize(n);
13        level.resize(n);
14        ptr.resize(n);
15    }
16    void addEdge(int u, int v, int c){
17        edges.emplace_back(u, v, c, 0);
18        edges.emplace_back(v, u, 0, 0);
19        adj[u].push_back(m);
20        adj[v].push_back(m+1);
21        m += 2;
22    }
23    bool bfs(){
24        fill(level.begin(), level.end(), -1);
25        queue<int> q;
26        q.push(s);
27        level[s] = 0;
28        while(!q.empty()){
29            int u = q.front();
30            q.pop();
31            for(int id : adj[u]){
32                if(edges[id].cap - edges[id].flow < 1)
33                    continue;
34                int v = edges[id].to;
35                if(level[v] != -1) continue;
36                level[v] = level[u] + 1;
37                q.push(v);
38            }
39        }
40        return level[t] != -1;

```

```

40    }
41    int dfs(int u, int pushed){
42        if(pushed == 0) return 0;
43        if(u == t) return pushed;
44        for(int& cid = ptr[u]; cid < (int)adj[u].size(); cid
45            ++){
46            int id = adj[u][cid];
47            int v = edges[id].to;
48            if(level[u] + 1 != level[v] || edges[id].cap -
49                edges[id].flow < 1) continue;
50            int tr = dfs(v, min(pushed, edges[id].cap - edges
51                [id].flow));
52            if(tr == 0) continue;
53            edges[id].flow += tr;
54            edges[id^1].flow -= tr;
55            return tr;
56        }
57        return 0;
58    }
59    int flow(){
60        int f = 0;
61        while(true){
62            if(!bfs()) break;
63            fill(ptr.begin(), ptr.end(), 0);
64            while(int pushed = dfs(s, INT_MAX)){
65                f += pushed;
66            }
67        }
68        return f;
69    }
70    int main(){
71        int n, m, s, t;
72        cin >> n >> m >> s >> t;
73        Dinic dinic(n, s, t);
74        for(int i = 0; i < m; i++){
75            int u, v, c;
76            cin >> u >> v >> c;
77            dinic.addEdge(u, v, c);
78        }
79        cout << dinic.flow() << endl;
80    }

```

5.5 EdmondsKarp

```

1 int main(){
2     int n,m,b,e,s,t,w,q=1;
3     while(cin>>n,n){
4         int ans=0,go=1,Min,a[n+1][n+1]={},last[n+1]={},can[n
5             +1]={};cin>>b>>e>>m;
6         while(m--){cin>>s>>t>>w;a[s][t]=a[t][s]+=w;}
7         while(go){go=0;
8             queue<int> Q;Q.push(b);Min=99999999;
9             while(Q.size()){
10                int top=Q.front();Q.pop();can[top]=1;
11                for(int k=1;k<=n;k++){if(a[top][k]&&!can[k]){
12                    last[k]=top;if(k==e){go=1;break;}Q.push(
13                        k);}
14                if(go){
15                    int S=s=e;
16                    while(last[S])Min=min(Min,a[last[S]][S]),
17                        S=last[S];

```

```

14         while(last[S])a[last[S]][S]-=Min,a[S][
15             last[S]]+=Min,S=last[S];
16         ans+=Min;
17         fill_n(last,n+1,0);fill_n(can,n+1,0);
18         break;
19     }
20 }
21 printf("Network %d\nThe bandwidth is %d.\n\n",q++,ans
22 );
23 }

```

6 Math

6.1 Bignumber

```

1 // a + b
2 string add(string a, string b) {
3     string ans;
4     int carry = 0;
5     while (a.size() < b.size()) a = '0' + a;
6     while (a.size() > b.size()) b = '0' + b;
7     for (int i = a.size() - 1; i >= 0; i--) {
8         int tmp = a[i] - '0' + b[i] - '0' + carry;
9         carry = tmp / 10;
10        ans = char(tmp % 10 + '0') + ans;
11    }
12    if (carry) ans = '1' + ans;
13    return ans;
14 }
15
16 // a - b
17 string sub(string a, string b) {
18     string ans;
19     int carry = 0;
20     while (a.size() < b.size()) a = '0' + a;
21     while (a.size() > b.size()) b = '0' + b;
22     for (int i = a.size() - 1; i >= 0; i--) {
23         int tmp = a[i] - b[i] - carry;
24         if (tmp < 0) {
25             tmp += 10;
26             carry = 1;
27         } else carry = 0;
28         ans = char(tmp + '0') + ans;
29     }
30     while (ans.size() > 1 && ans[0] == '0') ans.erase(0, 1);
31     return ans;
32 }
33
34 // a * b
35 string mul(string a, string b) {
36     string ans;
37     int carry = 0;
38     for (int i = a.size() - 1; i >= 0; i--) {
39         string tmp;
40         for (int j = b.size() - 1; j >= 0; j--) {
41             int t = (a[i] - '0') * (b[j] - '0') + carry;
42             carry = t / 10;
43             tmp = char(t % 10 + '0') + tmp;
44         }

```

```

45     if (carry) tmp = char(carry + '0') + tmp;
46     for (int j = 0; j < a.size() - 1 - i; j++) tmp += '0';
47     ans = add(ans, tmp);
48     carry = 0;
49 }
50 return ans;
51 }
52
53 // a / b
54 string div(string a, string b) {
55     string ans;
56     string tmp;
57     for (int i = 0; i < a.size(); i++) {
58         tmp += a[i];
59         int cnt = 0;
60         while (tmp.size() >= b.size()) {
61             if (tmp.size() > b.size() || tmp >= b) {
62                 tmp = sub(tmp, b);
63                 cnt++;
64             } else break;
65         }
66         ans += char(cnt + '0');
67     }
68     while (ans.size() > 1 && ans[0] == '0') ans.erase(0, 1);
69     return ans;
70 }
71
72 // a % b
73 string mod(string a, string b) {
74     string ans;
75     string tmp;
76     for (int i = 0; i < a.size(); i++) {
77         tmp += a[i];
78         int cnt = 0;
79         while (tmp.size() >= b.size()) {
80             if (tmp.size() > b.size() || tmp >= b) {
81                 tmp = sub(tmp, b);
82                 cnt++;
83             } else break;
84         }
85         ans += char(cnt + '0');
86     }
87     return tmp;
88 }
89
90 // a ^ b
91 string pow(string a, string b) {
92     string ans = "1";
93     while (b != "0") {
94         if ((b[b.size() - 1] - '0') % 2) ans = mul(ans, a);
95         a = mul(a, a);
96         b = div(b, "2");
97     }
98     return ans;
99 }
100
101 // a + b (可为负数)
102 string add(string a, string b) {
103     if (a[0] == '-' && b[0] == '-') return '-' + add(a.erase(
104         0, 1), b.erase(0, 1));
105     if (a[0] == '-') return sub(b, a.erase(0, 1));
106     if (b[0] == '-') return sub(a, b.erase(0, 1));
107     return add(a, b);

```

```

108 // A - B
109 // B - A
110 // A + B
111
112 string add(string a, string b){
113     // center = '.'
114     // decimal part
115     string ans;
116     int carry = 0;
117     int center = 0;
118     while (a.size() < b.size()) a = '0' + a;
119     while (a.size() > b.size()) b = '0' + b;
120     for (int i = a.size() - 1; i >= 0; i--) {
121         if (a[i] == '.') {
122             center = 1;
123             ans = '.' + ans;
124             continue;
125         }
126         int tmp = a[i] - '0' + b[i] - '0' + carry;
127         carry = tmp / 10;
128         ans = char(tmp % 10 + '0') + ans;
129     }
130     if (carry) ans = '1' + ans;
131     if (center) ans = '.' + ans;
132     return ans;
133 }
134 }

```

6.2 BignumberDec

```

1 #include <iostream>
2 #include <string.h>
3 #include <set>
4 #include <sstream>
5 using namespace std;
6 typedef long long LL;
7 class Frac{public:
8     LL mi,mn,md;
9     Frac(LL i=0,LL n=0,LL d=1){
10         i+=n/d;n%=d;
11         while((i<0&&n>0)|| (i>0&&n<0)){
12             if(n<0)n+=d,i-=1;
13             else n-=d,i+=1;
14         }
15         mi=i,mn=n,md=d;
16         while(n%md)swap(n,d);if(d<0)d=-d;
17         mn/=d;md/=d;
18     }
19     Frac operator+(Frac b){return Frac(mi+b.mi,mn*b.md+md*b.
20         mn,md*b.md);}
21 };
22 istream&operator>>(istream&xin,Frac&a){string s;xin>>s;int sn
23     =s[0]=='-';if(sn)s=s.substr(1);
24     int pos=s.find('.');LL i,n,d;
25     if(pos==-1){istringstream(s)>>i;n=0;d=1;
26     }else{
27         istringstream(s.substr(0,pos))>>i;
28         istringstream(s.substr(pos+1))>>n;
29         istringstream(string(s.size()-1-pos,'9'))>>d;
30     }
31     if(sn)i=-i,n=-n;
32     a=Frac(i,n,d);return xin;
33 }
34 ostream&operator<<(ostream&xout,Frac&a){

```

```

32 LL i=a.mi,n=a.mn,d=a.md;
33 if(i<0||n<0){xout<<"-";i=-i,n=-n;}
34 xout<<i;if(n)xout<<".";
35 set<LL>ns;ns.insert(0);
36 while(ns.find(n)!=ns.end()){ns.insert(n);n*=10;xout<<n/d;
37     n%=d;}
38 return xout;
39 }
40 int main(){
41     int cs;cin>>cs;
42     Frac a,b;
43     while(cin>>a>>b){
44         auto ans=a+b;
45         cout<<ans<<endl;
46     }
47     return 0;
48 }
49 string operator+(string a,string b){
50     string ans = "";
51     if(a.size() > b.size()) b = string(a.size()-b.size(),'0')
52         .append(b);
53     else a = string(b.size()-a.size(),'0')
54         .append(a);
55     int c=0,tt=0;
56     for(int i=a.size()-1;i>=0;i--){
57         if(a[i] == '.') {ans = string(".").append(ans);
58             continue;}
59         int x = a[i] - '0', y = b[i] - '0';
60         x = x+y+c;
61         c=x/10;
62         if(c==1 && i-1>=0 && a[i-1] == '.') tt=1;
63         ans = to_string(x%10).append(ans);
64     }
65     string temppp = string("0.").append(ans.size()-2-ans.find(
66         '.', '0')).append("1");
67     if(tt) ans = ans + temppp;
68     if(c) ans = string("1").append(ans);
69     return ans;
70 }
71 string operator-(string a,string b){
72     string ans="";
73     if(a.size() > b.size()) b = string(a.size()-b.size(),'0')
74         .append(b);
75     else a = string(b.size()-a.size(),'0')
76         .append(a);
77     int c=0,tt=0,xx=0,ii=0;
78     while(ii<b.size() &&a[ii] == b[ii]) ii++;
79     if(ii==b.size());
80     else if(a[ii] < b[ii]) {tt=1;swap(a,b);}
81     for(int i=a.size()-1;i>=0;i--){
82         if(a[i] == '.') {ans = string(".").append(ans);
83             continue;}
84         int x=a[i] - '0',y=b[i] - '0';
85         x=x-c-y;
86         if(x<0){c=1;x+=10;}
87         else c=0;
88         if(c==1 && i-1>=0 && a[i-1]=='.') xx=1;
89         ans = to_string(x%10).append(ans);

```

```

90 }
91
92 string temppp = string("0.").append(ans.size()-2-ans.find
93 ('.', '0')).append("1");
94 if(xx) ans = ans - temppp;
95
96 while(ans[0]=='0' && ans.size()>2 && ans[1] != '.') {
97     ans = ans.substr(1,ans.size()-1);
98 }
99
100 if(tt || c==1) ans = string("-").append(ans);
101 return ans;
102 }

```

6.3 線篩

```

1 vector<bool>p(n,1);
2 vector<int>prime;
3 for(int i=2;i<n;i++){
4     if(p[i])prime.push_back(i);
5     for(auto&k:prime){
6         if(i*k>=n)break;
7         p[i*k]=0;
8         if(i%k==0)break;
9     }
10 }

```

6.4 Fib

```

1 1/sqrt(5)*(pow((1+sqrt(5))/2,n)-pow((1-sqrt(5))/2,n))

```

6.5 鞋帶

```

1 vector<pair<int,int>> v(n);
2 for(auto&n:v) cin >> n.first >> n.second;
3 int area = 0;
4 for(int i = 0; i < v.size(); i++){
5     area += v[i].first * v[(i+1)%v.size()].second;
6     area -= v[i].second * v[(i+1)%v.size()].first;
7 }
8 cout << abs(area)/2 << endl;

```

6.6 SG

```

1 Anti Nim (取走最後一個石子者敗) :
2 先手必勝 if and only if
3 1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
4 2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
5 -----
6 Anti-SG (決策集合為空的遊戲者贏) :
7 定義 SG 值為 0 時，遊戲結束，
8 則先手必勝 if and only if

```

```

9 1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
10 2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0。
11 -----
12 Sprague-Grundy :
13 1. 雙人、回合制
14 2. 資訊完全公開
15 3. 無隨機因素
16 4. 可在有限步內結束
17 5. 沒有和局
18 6. 雙方可採取的行動相同
19
20 SG(S) 的值為 0 : 後手(P)必勝
21 不為 0 : 先手(N)必勝
22 int mex(set S) {
23     // find the min number >= 0 that not in the S
24     // e.g. S = {0, 1, 3, 4} mex(S) = 2
25 }
26 state = []
27 int SG(A) {
28     if (A not in state) {
29         S = sub_states(A)
30         if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B)
31             for B in S])
32         else state[A] = mex(set(SG(B) for B in next_states(A)))
33     } return state[A]
34 }

```

6.7 擴展歐幾里德

```

1 // 給 a,b , 解 ax+by=gcd(a,b)
2 typedef pair<ll, ll> pii;
3 pii extgcd(ll a, ll b) {
4     if (b == 0) return {1, 0};
5     ll k = a / b;
6     pii p = extgcd(b, a - k * b);
7     return {p.second, p.first - k * p.second};
8 }

```

6.8 FPow

```

1 // 問 a ^ p
2 ll fastpow(ll a, int p) {
3     ll ret = 1;
4     while (p) {
5         if (p & 1) ret *= a;
6         a *= a, p >>= 1;
7     } return ret;
8 }
9 // 問 (a ^ p) mod m
10 ll fastpow(ll a, ll p, ll m) {
11     ll ret = 1;
12     while (p) {
13         if (p & 1) ret = ret * a % m;
14         a = a * a % m, p >>= 1;
15     } return ret;
16 }

```

6.9 質因數分解

```

1 LL func(const LL n,const LL mod,const int c) {
2     return (LLmul(n,n,mod)+c+mod)%mod;
3 }
4 LL pollorrho(const LL n, const int c) { //循環節長度
5     LL a=1, b=1;
6     a=func(a,n,c)%n;
7     b=func(b,n,c)%n; b=func(b,n,c)%n;
8     while(gcd(abs(a-b),n)!=1) {
9         a=func(a,n,c)%n;
10        b=func(b,n,c)%n; b=func(b,n,c)%n;
11    }
12    return gcd(abs(a-b),n);
13 }
14 void prefactor(LL &n, vector<LL> &v) {
15     for(int i=0;i<12;++i) {
16         while(n%prime[i]==0) {
17             v.push_back(prime[i]);
18             n/=prime[i];
19         }
20     }
21 }
22 void smallfactor(LL n, vector<LL> &v) {
23     if(n<MAXPRIME) {
24         while(isp[(int)n]) {
25             v.push_back(isp[(int)n]);
26             n/=isp[(int)n];
27         }
28         v.push_back(n);
29     } else {
30         for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
31             while(n%prime[i]==0) {
32                 v.push_back(prime[i]);
33                 n/=prime[i];
34             }
35         }
36         if(n!=1) v.push_back(n);
37     }
38 }
39 void comfactor(const LL &n, vector<LL> &v) {
40     if(n<1e9) {
41         smallfactor(n,v);
42         return;
43     }
44     if(Isprime(n)) {
45         v.push_back(n);
46         return;
47     }
48     LL d;
49     for(int c=3;++c) {
50         d = pollorrho(n,c);
51         if(d!=n) break;
52     }
53     comfactor(d,v);
54     comfactor(n/d,v);
55 }
56 void Factor(const LL &x, vector<LL> &v) {
57     LL n = x;
58     if(n==1) { puts("Factor 1"); return; }
59     prefactor(n,v);
60     if(n==1) return;
61     comfactor(n,v);
62     sort(v.begin(),v.end());
63 }

```

```

64 void AllFactor(const LL &n,vector<LL> &v) {
65     vector<LL> tmp;
66     Factor(n,tmp);
67     v.clear();
68     v.push_back(1);
69     int len;
70     LL now=1;
71     for(int i=0;i<tmp.size();++i) {
72         if(i==0 || tmp[i]!=tmp[i-1]) {
73             len = v.size();
74             now = 1;
75         }
76         now*=tmp[i];
77         for(int j=0;j<len;++j)
78             v.push_back(v[j]*now);
79     }
80 }

```

6.10 Expression

```

1  /**
2  * 支援處理四則運算的工具。給四則運算的字串，檢查格式並計算其
3  * 值。如果
4  * 格式不合法，會丟出錯誤。複雜度 O(字串長度)。支援的符號有
5  * 四則運算
6  * 和求餘數，先乘除後加減。可以使用括號、或前置正負號。數字開
7  * 頭可以為
8  * 零或禁止為零。可以兼容或禁止多重前置號 (例如 --1 視為 1 、
9  * ++-1
10 * 視為 -1)。空字串視為不合法。運算範圍限於 long long。如果
11 * 試圖除
12 * 以零或對零求餘也會丟出錯誤。
13 */
14 void req(bool b) { if (!b) throw ""; }
15 const int B = 2; // 可以調整成 B 進位
16 class Expr {
17 private:
18     deque<char> src;
19     Expr(const string& s) : src(s.begin(), s.end()) {}
20     inline char top() {
21         return src.empty() ? '\0' : src.front();
22     }
23     inline char pop() {
24         char c = src.front(); src.pop_front(); return c;
25     }
26     ll n() {
27         ll ret = pop() - '0';
28         // 若要禁止數字以 0 開頭，加上這行
29         // req(ret || !isdigit(top()));
30         while (isdigit(top())) ret = B * ret + pop() - '0';
31         return ret;
32     }
33     ll fac() {
34         if (isdigit(top())) return n();
35         if (top() == '-') { pop(); return -fac(); }
36         if (top() == '(') {
37             pop();
38             ll ret = expr(1);
39             req(pop() == ')');
40             return ret;
41         }
42     }
43 }

```

```

37 // 若要允許前置正號，加上這行
38 // if(top() == '+') { pop(); return fac(); }
39 throw "";
40 }
41 ll term() {
42     ll ret = fac(); char c = top();
43     while (c == '*' || c == '/' || c == '%') {
44         pop();
45         if (c == '*') ret *= fac();
46         else {
47             ll t = fac(); req(t);
48             if (c == '/') ret /= t; else ret %= t;
49         }
50         c = top();
51     } return ret;
52 }
53 ll expr(bool k) {
54     ll ret = term();
55     while (top() == '+' || top() == '-')
56         if (pop() == '+') ret += term();
57         else ret -= term();
58     req(top() == (k ? ')' : '\0'));
59     return ret;
60 }
61 public:
62 // 給定數學運算的字串，求其值。若格式不合法，丟出錯誤。
63 static ll eval(const string& s) {
64     // 若要禁止多重前置號，加上這四行
65     // req(s.find("--") == -1); // 禁止多重負號
66     // req(s.find("-+") == -1);
67     // req(s.find("+") == -1);
68     // req(s.find("++") == -1);
69     return Expr(s).expr(0);
70 }
71 };

```

6.11 JosephusProblem

```

1 // O(n) space and O(nlogn) time
2 #pragma GCC optimize("O3","unroll-loops")
3 #include <bits/extc++.h>
4 #include <bits/stdc++.h>
5 #define IO cin.tie(0), ios::sync_with_stdio(0)
6 #define ll long long
7 #define INF 0x3f3f3f3f
8 #define MAXN 100000
9 #define orderset tree<ll,null_type,less<ll>,rb_tree_tag,
10 tree_order_statistics_node_update>
11 using namespace __gnu_pbds;
12 using namespace std;
13 int32_t main(){
14     IO;
15     int n, k;
16     cin >> n >> k;
17     orderset s;
18     for(int i=1;i<=n;i++)s.insert(i);
19     int pos = 0;
20     while(s.size()){
21         pos = (pos + k) % s.size();
22         auto it = s.find_by_order(pos);
23         cout << *it << " ";

```

```

24         s.erase(it);
25     }
26 }
27 // O(1) space and O(n) time
28 #include <iostream>
29 using namespace std;
30 int Josephus(int N, int k){
31     // Initialize variables i and ans with 1 and 0
32     // respectively.
33     int i = 1, ans = 0;
34     // Run a while loop till i <= N
35     while (i <= N) {
36         // Update the Value of ans and Increment i by 1
37         ans = (ans + k) % i;
38         i++;
39     }
40     // Return required answer
41     return ans + 1;
42 }
43 // main function
44 int main(){
45     int N = 14, k = 2;
46     cout << Josephus(N, k) << endl;
47     return 0;
48 }
49 // O(n) space and O(n) time
50 #include <bits/stdc++.h>
51 using namespace std;
52 // Recursive function to implement the Josephus problem
53 int josephus(int n, int k)
54 {
55     if (n == 1)
56         return 1;
57     else
58         // The position returned by josephus(n - 1, k)
59         // is adjusted because the recursive call
60         // josephus(n - 1, k) considers the
61         // original position k % n + 1 as position 1
62         return (josephus(n - 1, k) + k - 1) % n + 1;
63 }
64 // Driver code
65 int main()
66 {
67     int n = 14;
68     int k = 2;
69     cout << "The chosen place is " << josephus(n, k);
70     return 0;
71 }
72 // asking last one O(klog(n))
73 //編號從1開始，結果要加1
74 int josephus(int n, int k) {
75     if (k == 1) return n - 1;
76     int ans = 0;
77     for (int i = 2; i <= n; ) {
78         if (ans + k >= i) {

```

```

89     ans = (ans + k) % i;
90     i++;
91     continue;
92 }
93 int step = (i - 1 - ans - 1) / (k - 1); // 向下取整
94 if (i + step > n) {
95     ans += (n - (i - 1)) * k;
96     break;
97 }
98 i += step;
99 ans += step * k;
100 }
101 return ans;
102 }
103
104 int main() {
105     int n, k;
106     while (scanf("%d%d", &n, &k) == 2)
107         printf("%d\n", josephus(n, k) % n + 1);
108     return 0;
109 }

```

7 String

7.1 Trie

```

1 class Trie {
2 private:
3     struct Node {
4         int cnt = 0, sum = 0;
5         Node *tr[128] = {};
6         ~Node() {
7             for (int i = 0; i < 128; i++)
8                 if (tr[i]) delete tr[i];
9         }
10    };
11    Node *root;
12 public:
13    void insert(char *s) {
14        Node *ptr = root;
15        for (; *s; s++) {
16            if (!ptr->tr[*s]) ptr->tr[*s] = new Node();
17            ptr = ptr->tr[*s];
18            ptr->sum++;
19        }
20        ptr->cnt++;
21    }
22    inline int count(char *s) {
23        Node *ptr = find(s);
24        return ptr ? ptr->cnt : 0;
25    }
26    Node *find(char *s) {
27        Node *ptr = root;
28        for (; *s; s++) {
29            if (!ptr->tr[*s]) return 0;
30            ptr = ptr->tr[*s];
31        } return ptr;
32    }
33    bool erase(char *s) {
34        Node *ptr = find(s);
35        if (!ptr) return false;

```

```

36        int num = ptr->cnt;
37        if (!num) return false;
38        ptr = root;
39        for (; *s; s++) {
40            Node *tmp = ptr;
41            ptr = ptr->tr[*s];
42            ptr->sum -= num;
43            if (!ptr->sum) {
44                delete ptr;
45                tmp->tr[*s] = 0;
46                return true;
47            }
48        }
49    }
50    Trie() { root = new Node(); }
51    ~Trie() { delete root; }
52 };

```

7.2 AC 自動機

```

1 struct _AC{
2     _AC *child[26];
3     _AC *Fail;
4     vector<pair<int,int>> out;
5     _AC(){
6         Fail = NULL;
7         memset(child,0,sizeof(child));
8     }
9 }*root;
10 void Insert_AC(string s){
11     int n;
12     _AC *p = root;
13     for(auto&k:s){
14         n = k - 'a';
15         if(!p->child[n]) p->child[n] = new _AC();
16         p = p->child[n];
17     }
18     p->out.push_back({s.size(),0});
19 }
20 void Construct_AC(){
21     queue<_AC*> Q;
22     for(int k=0;k<26;k++){
23         if(root->child[k]){
24             root->child[k]->Fail = root;
25             Q.push(root->child[k]);
26         }
27         else root->child[k] = root;
28     }
29     _AC *p;
30     while(!Q.empty()){
31         p = Q.front();
32         Q.pop();
33         for(int k=0;k<26;k++){
34             if(!p->child[k])p->child[k] = p->Fail->child[k];
35             else {
36                 p->child[k]->Fail = p->Fail->child[k];
37                 for(auto&i:p->Fail->child[k]->out)p->child[k]->out.
38                     push_back({i.first,1});
39                 Q.push(p->child[k]);
40             }
41         }
42     }

```

```

43 void Match_AC(string t){
44     int n;
45     _AC *p = root;
46     for(int k=0;k<t.size();k++){
47         n = t[k] - 'a';
48         p = p->child[n];
49         _AC *fail = p;
50         while(fail != root && fail->out.size()){
51             for(auto&i:fail->out)if(!i.second||fail->Fail->out.size
52                 ())cout<<t.substr(k-i.first+1,i.first)<<'\\n';
53             fail->out.clear();
54             fail->Fail->out.clear();
55             fail = fail->Fail;
56         }
57     }
58 }
59 int main(){
60     int n,m;string p,t;cin>>n;
61     while(cin>>n>>m){
62         root = new _AC();
63         while(n--){
64             cin>>p;
65             Insert_AC(p);
66         }
67         Construct_AC();
68         cin>>n>>m;
69         cin>>t;
70         Match_AC(t);
71     }
72 }

```

7.3 KMP

```

1 // KMP fail function.
2 int* kmp_fail(string& s) {
3     int* f = new int[s.size()]; int p = f[0] = -1;
4     for (int i = 1; s[i]; i++) {
5         while (p != -1 && s[p + 1] != s[i]) p = f[p];
6         if (s[p + 1] == s[i]) p++;
7         f[i] = p;
8     }
9     return f;
10 }
11 // 問 sub 在 str 中出現幾次。
12 int kmp_count(string& str, string& sub) {
13     int* fail = kmp_fail(sub); int p = -1, ret = 0;
14     for (int i = 0; i < str.size(); i++) {
15         while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
16         if (sub[p + 1] == str[i]) p++;
17         if (p == sub.size() - 1) p = fail[p], ret++;
18     }
19     delete[] fail; return ret;
20 }
21 // 問 sub 在 str 第一次出現的開頭 index 。 -1 表示找不到。
22 int kmp(string& str, string& sub) {
23     int* fail = kmp_fail(sub);
24     int i, j = 0;
25     while (i < str.size() && j < sub.size()) {
26         if (sub[j] == str[i]) i++, j++;
27         else if (j == 0) i++;
28         else j = fail[j - 1] + 1;

```

```

29 }
30 delete[] fail;
31 return j == sub.size() ? (i - j) : -1;
32 }

```

7.4 LPS

```

1 int lps(string s){
2     int N=2*s.size()+1;
3     vector<int> dp(N);
4     string s2="*";
5     for(auto&c:s){
6         s2.push_back(c);
7         s2.push_back('*');
8     }
9     int C=0,R=0;
10    for(int i = 1;i < N;i++){
11        if(i>R)C=R=i;
12        else{
13            int mirrorI=C-(i-C);
14            dp[i]=min(dp[mirrorI],R-i);
15        }
16        int j=dp[i]+1;
17        while((i-j)>=0)&&(i+j<N)&&(s2[i-j]==s2[i+j]))j++;
18        dp[i]=j-1;
19        if(i+dp[i]>R){
20            C=i;
21            R=i+dp[i];
22        }
23    }
24    return *max_element(dp.begin(),dp.end());
25 }
26
27 string lps(string s){
28     int N=2*s.size()+1;
29     vector<int> dp(N);
30     string s2="*";
31     for(auto&c:s){
32         s2.push_back(c);
33         s2.push_back('*');
34     }
35     int C=0,R=0;
36     for(int i = 1;i < N;i++){
37         if(i>R)C=R=i;
38         else{
39             int mirrorI=C-(i-C);
40             dp[i]=min(dp[mirrorI],R-i);
41         }
42         int j=dp[i]+1;
43         while((i-j)>=0)&&(i+j<N)&&(s2[i-j]==s2[i+j]))j++;
44         dp[i]=j-1;
45         if(i+dp[i]>R){
46             C=i;
47             R=i+dp[i];
48         }
49     }
50     auto it=max_element(dp.begin(),dp.end());
51     int maxLen=*it;
52     int index=it-dp.begin();
53     return s.substr((index-maxLen)/2,maxLen);
54 }

```

7.5 EditDistance

```

1 // 問從 src 到 dst 的最小 edit distance
2 // ins 插入一個字元的成本
3 // del 刪除一個字元的成本
4 // sst 替換一個字元的成本
5 ll edd(string& src, string& dst, ll ins, ll del, ll sst) {
6     ll dp[src.size() + 1][dst.size() + 1]; // 不用初始化
7     for (int i = 0; i <= src.size(); i++) {
8         for (int j = 0; j <= dst.size(); j++) {
9             if (i == 0) dp[i][j] = ins * j;
10            else if (j == 0) dp[i][j] = del * i;
11            else if (src[i - 1] == dst[j - 1])
12                dp[i][j] = dp[i - 1][j - 1];
13            else
14                dp[i][j] = min(dp[i][j - 1] + ins,
15                             min(dp[i - 1][j] + del,
16                                 dp[i - 1][j - 1] + sst));
17        }
18    }
19    return dp[src.size()][dst.size()];
20 }

```

8 Geometry

8.1 angle

```

1 // 有序的點集合，計算每個內角角度
2 // 逆時針方向，從第一個點開始
3 // 0 <= angle <= 2*PI
4 vector<double> angle(const vector<pair<int,int>>& v){
5     vector<double> ret;
6     for(int i = 0; i < v.size(); i++){
7         int x1 = v[(i-1+v.size())%v.size()].first - v[i].first;
8         int y1 = v[(i-1+v.size())%v.size()].second - v[i].second;
9         int x2 = v[(i+1)%v.size()].first - v[i].first;
10        int y2 = v[(i+1)%v.size()].second - v[i].second;
11        double a = atan2(x1, y1) - atan2(x2, y2);
12        if(a < 0) a += 2*M_PI;
13        ret.push_back(a);
14        // ret = ret*180/M_PI // convert to degree
15    }
16    return ret;
17 }
18
19 // 計算三個點的內角角度
20 // 0 <= angle <= 2*PI
21 double angle(const pair<int,int>& a, const pair<int,int>& b,
22              const pair<int,int>& c){
23     int x1 = a.first - b.first;
24     int y1 = a.second - b.second;
25     int x2 = c.first - b.first;
26     int y2 = c.second - b.second;
27     double ret = atan2(x1, y1) - atan2(x2, y2);
28     if(ret < 0) ret += 2*M_PI;
29     return ret;
30     // ret = ret*180/M_PI // convert to degree

```

```

30 }
31
32 // 求兩線交點(兩線必相交)
33 // 0 <= angle <= 2*PI
34 pair<double,double> intersection(const pair<double,double>&
35                                  p1, const pair<double,double>& p2, const pair<double,
36                                  double>& q1, const pair<double,double>& q2){
37     double a1 = p2.second - p1.second;
38     double b1 = p1.first - p2.first;
39     double c1 = a1 * p1.first + b1 * p1.second;
40     double a2 = q2.second - q1.second;
41     double b2 = q1.first - q2.first;
42     double c2 = a2 * q1.first + b2 * q1.second;
43     double det = a1 * b2 - a2 * b1;
44     return make_pair((b2 * c1 - b1 * c2) / det, (a1 * c2 - a2 * c1) / det);
45 }

```

8.2 ConvexHull

```

1 void Dhull(vector<pii> points, vector<pii> &hull){
2     hull.push_back(points[0]); hull.push_back(points[1]);
3     for(int i = 2; i < points.size(); i++){
4         while(hull.size() >= 2){
5             pair<int,int> p1 = hull[hull.size()-2], p2 =
6                 hull[hull.size()-1], p3 = points[i];
7             int x1 = p2.first - p1.first, y1 = p2.second -
8                 p1.second;
9             int x2 = p3.first - p2.first, y2 = p3.second -
10                p2.second;
11             if(x1*y2 - x2*y1 <= 0) break;
12             hull.pop_back();
13         }
14         hull.push_back(points[i]);
15     }
16 }
17
18 int main(){
19     IO;
20     vector<pair<int,int>> points,hull;
21     int n, x, y;
22     cin >> n;
23     for(int i = 0; i < n; i++) cin >> x >> y, points.pb({x,y});
24     sort(points.begin(), points.end());
25     Dhull(points,hull);
26     reverse(points.begin(),points.end());
27     Dhull(points,hull);
28     for(auto p : hull) cout << p.first << " " << p.second << endl;
29 }

```

8.3 旋轉卡尺

```

1 ### 問題：最遠點對
2
3 考慮一個問題，你有一個凸多邊形，你想找到多邊形上距離最遠的兩個點。這可以通過旋轉卡尺方法來解決。
4
5 ### 方法：
6

```

```

1. **初始化**：選擇凸多邊形的一個頂點作為起點，將一條與多邊形的邊平行的輔助線放在該頂點上。
2. **旋轉**：開始將這條輔助線旋轉。對於多邊形的每一條邊，都會有一個時刻該邊成為輔助線上的最遠邊。這意味著在輔助線上，與這條邊的兩個端點的距離是最遠的。
3. **更新**：在每次旋轉時，記錄與起始頂點距離最遠的點。繼續旋轉輔助線，直到它再次與初始邊平行。
4. **換邊**：選擇下一個頂點，並重複上述步驟，直到所有頂點都被考慮過。

經過這些步驟，你就可以找到多邊形上距離最遠的兩個點。

### 分析：
因為是凸多邊形，所以當你旋轉輔助線時，與輔助線距離最遠的點只會在某個方向上增加。這使得你可以在O(n)的時間複雜度內解決這個問題，其中n是多邊形的頂點數。對於每個頂點，只需要考慮一次旋轉，並且對於每次旋轉，只需要O(1)的時間來更新最遠的點。

這只是使用選轉卡尺方法的一個例子。該方法還可以用於解決其他與凸多邊形相關的問題，如計算最大面積的內接矩形等。

typedef pair<ll, ll> pii;
#define x first
#define y second
#define ii (i + 1) % n // 打字加速！
inline pii operator-(const pii& a, const pii& b) {
    return {a.x - b.x, a.y - b.y};
} // const 不可省略
inline ll operator*(const pii& a, const pii& b) {
    return a.x * b.y - a.y * b.x;
}
inline ll crzf(const pii& o, const pii& a, const pii& b) {
    return (a - o) * (b - o)
}
inline ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// 給平面上任意個點，求其凸包。返回順序為逆時針。此方法會移除重複點。
#define jud \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
vector<pii> makepoly(vector<pii>& pp) {
    int n = pp.size();
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= t && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back();
    return ret;
}

```

```

}
// (shoelace formula)
// 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
ll area(vector<pii>& poly) {
    int n = poly.size();
    ll ret = 0;
    for (int i = 0; i < n; i++)
        ret += (poly[i].x * poly[i+1].y);
    for (int i = 0; i < n; i++)
        ret -= (poly[i].y * poly[i+1].x);
    return ret;
}
// 給凸包，問其兩點最遠距離「的平方」。若要問平面上任意個點的兩點最遠距離，請先轉成凸包。若凸包少於兩個點，回傳零。
#define kk (k + 1) % n
ll maxdist(vector<pii>& poly) {
    int k = 1, n = poly.size();
    if (n < 2) return 0;
    if (n == 2) return dd(poly[0], poly[1]);
    ll ret = 0;
    for (int i = 0; i < n; i++) {
        while (abs(crzf(poly[kk], poly[i], poly[i+1])) >= abs(crzf(poly[k], poly[i], poly[i+1])))
            k = kk;
        ret = max(ret, max(dd(poly[i], poly[k]), dd(poly[i], poly[k+1])));
    }
    return ret;
}

建立凸包 (makepoly 函數)
計算多邊形面積 (area 函數)
計算凸包上兩點之間的最遠距離 (maxdist 函數)
以下是對各部分的簡單解析：
1. 建立凸包 (makepoly 函數)：
此函數的目的是為了找出一組點的凸包。算法首先對點進行排序，再進行上下凸殼的建立。在建立過程中，它利用叉積 (crzf 函數) 來確定三點之間的相對位置，並決定該點是否應該包含在凸殼內。
2. 計算多邊形面積 (area 函數)：
此函數使用 shoelace formula (鞋帶公式) 來計算凸包的面積。公式基本上是用選轉卡尺技巧來解決的。對於凸包上的每一點，找到與該點和它的下一個點形成的直線最遠的點。由於凸性，最遠的點會沿著凸包的邊移動，所以可以使用選轉卡尺技巧來有效地找到最遠的點。
3. 計算凸包上兩點之間的最遠距離 (maxdist 函數)：

```

8.4 closestPair

最近點對問題是計算幾何中的另一個經典問題，目的是在平面上的給定點集中找到距離最近的一對點。

常見的算法是基於分治策略，具有 $O(n \log n)$ 的時間複雜度。以下是算法的基本思路：

- 將點按照x坐標排序。
- 將點集分成兩半。
- 遞迴地找出每一半中的最近點對。令這兩個距離中的最小值為d。
- 考慮中間縱帶的寬度為2d的所有點。這些點在y坐標上必須排序。
- 對於帶中的每一點，檢查與其相鄰的點，直到y距離大於d。這部分的點對不會超過6個。
- 如果在中間帶中找到距離小於d的點對，則更新d。

以下是該算法的簡化版本：

```

struct Point {
    double x, y;

    double distanceTo(const Point& other) const {
        return sqrt((x - other.x) * (x - other.x) + (y - other.y) * (y - other.y));
    }
};

bool compareX(const Point &a, const Point &b) {
    return a.x < b.x;
}

bool compareY(const Point &a, const Point &b) {
    return a.y < b.y;
}

double closestPair(Point px[], Point py[], int n) {
    if (n <= 3) {
        double minDist = DBL_MAX;
        for (int i = 0; i < n; i++)
            for (int j = i + 1; j < n; j++)
                minDist = min(minDist, px[i].distanceTo(px[j]));
        return minDist;
    }

    int mid = n / 2;
    Point midPoint = px[mid];

    Point pyl[mid], pyr[n - mid];
    int li = 0, ri = 0;
    for (int i = 0; i < n; i++) {
        if (py[i].x < midPoint.x) pyl[li++] = py[i];
        else pyr[ri++] = py[i];
    }

    double dl = closestPair(px, pyl, mid);
    double dr = closestPair(px + mid, pyr, n - mid);
    double d = min(dl, dr);

    Point strip[n];
    int j = 0;
    for (int i = 0; i < n; i++)
        if (abs(py[i].x - midPoint.x) < d)
            strip[j++] = py[i];

    double minStrip = d;
    for (int i = 0; i < j; i++)
        for (int k = i + 1; k < j && (strip[k].y - strip[i].y) < minStrip; k++)

```

```

62         minStrip = min(minStrip, strip[i].distanceTo(
63             strip[k]));
64     return min(d, minStrip);
65 }
66
67 double closestPair(Point points[], int n) {
68     Point px[n], py[n];
69     for (int i = 0; i < n; i++) {
70         px[i] = points[i];
71         py[i] = points[i];
72     }
73
74     sort(px, px + n, compareX);
75     sort(py, py + n, compareY);
76
77     return closestPair(px, py, n);
78 }
79
80
81
82
83 typedef pair<ll, ll> pii;
84 #define x first
85 #define y second
86 ll dd(const pii& a, const pii& b) {
87     ll dx = a.x - b.x, dy = a.y - b.y;
88     return dx * dx + dy * dy;
89 }
90 const ll inf = 1e18;
91 ll dac(vector<pii>& p, int l, int r) {
92     if (l >= r) return inf;
93     int m = (l + r) / 2;
94     ll d = min(dac(p, l, m), dac(p, m + 1, r));
95     vector<pii> t;
96     for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
97         t.push_back(p[i]);
98     for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
99         t.push_back(p[i]);
100     sort(t.begin(), t.end(),
101         [](pii& a, pii& b) { return a.y < b.y; });
102     int n = t.size();
103     for (int i = 0; i < n - 1; i++)
104         for (int j = i + 1; j < n; j++)
105             // 這裡可以知道是哪兩點是最小點對
106             d = min(d, dd(t[i], t[j]));
107     return d;
108 }
109 // 給一堆點，求最近點對的距離「的平方」。
110 ll closest_pair(vector<pii>& pp) {
111     sort(pp.begin(), pp.end());
112     return dac(pp, 0, pp.size() - 1);
113 }
114
115 首先，定義了一個pair來表示點，其中x和y分別代表x和y坐標。
116 dd函數是用來計算兩個點之間的距離的平方。
117 dac是主要的分治函數。給定一個點集p和兩個索引l和r，這個函數會
118 返回這些點中距離最小的點對的距離平方。
119 如果l大於等於r，則返回inf，表示在這個區間沒有有效的點。
120 m是中點索引。
121 這個函數遞迴地計算左半部分和右半部分的最小距離。
122 t是一個臨時vector，它將存儲距中點在x軸上距離小於d的所有點。
123 根據x坐標從中點開始，左右兩邊選擇與中點在x軸上距離小於d的點。
124 接著，根據y坐標對t中的點進行排序。

```

```

124 然後對t中的每個點，只需要檢查接下來的三個點（如果它們存在），
    因為根據算法的性質，不可能有超過三個點在y軸上與當前點的
    距離小於d。
125 返回最小距離的平方。
126 closest_pair函數根據x坐標對所有點進行排序，然後調用dac函數。
127 這個程式碼的優點是它避免了計算所有點對之間的距離，從而將時間
    複雜度從 $O(n^2)$ 降低到 $O(n \log n)$ 。

```

8.5 MinCircle

```

1 using PT = point<T>;
2 using CPT = const PT;
3 PT circumcenter(CPT &a, CPT &b, CPT &c) {
4     PT u = b-a, v = c-a;
5     T c1 = u.abs2()/2, c2 = v.abs2()/2;
6     T d = u.cross(v);
7     return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
8 }
9 void solve(PT p[], int n, PT &c, T &r2){
10     random_shuffle(p,p+n);
11     c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
12     for(int i=1; i<n; i++)
13         if( (p[i]-c).abs2() > r2) {
14             c=p[i]; r2=0;
15             for(int j=0; j<i; j++)
16                 if( (p[j]-c).abs2() > r2) {
17                     c.x = (p[i].x+p[j].x)/2;
18                     c.y = (p[i].y+p[j].y)/2;
19                     r2 = (p[j]-c).abs2();
20                     for(int k=0; k<j; k++)
21                         if( (p[k]-c).abs2() > r2) {
22                             c = circumcenter(p[i], p[j], p[k]);
23                             r2 = (p[i]-c).abs2();
24                         }
25                 }
26         }
27 }

```

最小覆蓋圓是指一個最小的圓，其中包含給定的所有點。這是一個經典的計算幾何問題。

解決這個問題的一種方法是 Welzl 的算法，這是一個遞迴算法，對於隨機化的輸入，其平均時間複雜度是 $O(n)$ 。然而，最壞情況下，該算法的運行時間為 $\mathcal{O}(n^4)$ 。

以下是算法的基本想法和步驟：

1. ****基本情況****：如果輸入點的集合是空的或已有 3 個點在邊界集合中，則問題可以直接求解。
2. ****遞迴步驟****：從點集中移除一個點 `p`，遞迴地找到剩餘點的最小覆蓋圓。
3. 如果 `p` 已在圓內，那麼這個圓也是整個點集的最小覆蓋圓。
4. 如果 `p` 在圓外，那麼我們知道 `p` 必須在新的最小覆蓋圓的邊界上。這意味著最小覆蓋圓的邊界上至少有一個點是 `p`。我們可以遞迴地使用點 `p` 和邊界點集來解決這個問題。

要注意的是，Welzl 的算法通常在實踐中運行得非常快，但由於其遞迴的性質，它可能不是最適合大型實例的解決方案。

以下是該算法的精簡描述：

```

...cpp
47 Circle welzl(vector<Point> & P, list<Point> R = list<Point>())
48 {
49     if (P.empty() || R.size() == 3) {
50         return trivial(R);
51     }
52
53     int idx = rand() % P.size();
54     Point p = P[idx];
55     P.erase(P.begin() + idx);
56
57     Circle d = welzl(P, R);
58
59     if (inside(d, p)) {
60         return d;
61     }
62
63     R.push_back(p);
64     return welzl(P, R);
65 }
...

```

在上述函數中，`Circle` 和 `Point` 分別表示圓和點的資料結構，`trivial` 是一個輔助函數，用於基本情況（例如當邊界集合中有 0、1、2 或 3 個點時），`inside` 是一個檢查點是否在圓內的函數。

這只是算法的高級描述。在實際實施時，還需要考慮如何表示圓和點，以及如何執行諸如計算圓或檢查點是否在圓內等基本操作。

用於計算給定點集中所有點的最小覆蓋圓。它主要基於增量算法，其中每次增加一個新點，我們都檢查這個點是否在當前的圓中。如果不在，我們就更新圓。

以下是程式碼的詳細解析：

點的定義：

PT 是點的資料結構。根據給定的片段，我們不知道 point 的完整定義，但我們可以推斷它有 x 和 y 屬性，以及一些函數方法，如 abs2（可能是點到原點的距離平方）。
外接圓的中心（circumcenter 函數）：

這個函數計算三個點 a, b, 和 c 的外接圓的中心。給定三點，外接圓的中心可以用線性方程組解出。這裡用到了叉積來求解。
計算最小覆蓋圓（solve 函數）：

算法從一個隨機點開始，初始化圓心為這個點，半徑為 0。
然後，對於每一個點，檢查它是否在當前的圓中。
如果點不在圓內，我們知道該點必須在新圓的邊界上。我們接著檢查所有之前的點。
如果之前的點也不在新圓內，則我們知道新圓的邊界上至少有這兩個點。此時，新圓的中心是這兩個點的中點。
然後，再次檢查所有之前的點。如果還有點不在新圓內，那麼我們知道新圓的邊界上有這三個點。在這種情況下，新圓的中心是這三個點的外接圓的中心。

89 該算法基於以下事實：最小覆蓋圓的邊界上有 1、2 或 3 個點。因此，當我們找到一個不在當前圓內的點時，我們知道它必須在新圓的邊界上。使用這個事實，我們可以增量地找到新的最小覆蓋圓。

8.6 MaximumBipartiteMatching

```

1 // M is number of applicants
2 // and N is number of jobs
3 #define M 6
4 #define N 6
5
6 // A DFS based recursive function
7 // that returns true if a matching
8 // for vertex u is possible
9 bool bpm(bool bpGraph[M][N], int u, bool seen[], int matchR[])
10 {
11     // Try every job one by one
12     for (int v = 0; v < N; v++) {
13         // If applicant u is interested in
14         // job v and v is not visited
15         if (bpGraph[u][v] && !seen[v]) {
16             // Mark v as visited
17             seen[v] = true;
18             // If job 'v' is not assigned to an
19             // applicant OR previously assigned
20             // applicant for job v (which is matchR[v])
21             // has an alternate job available.
22             // Since v is marked as visited in
23             // the above line, matchR[v] in the following
24             // recursive call will not get job 'v' again
25             if (matchR[v] < 0 || bpm(bpGraph, matchR[v], seen,
26                                     matchR)) {
27                 matchR[v] = u;
28                 return true;
29             }
30         }
31     }
32     return false;
33 }
34
35 // Returns maximum number
36 // of matching from M to N
37 int maxBPM(bool bpGraph[M][N]) {
38     // An array to keep track of the
39     // applicants assigned to jobs.
40     // The value of matchR[i] is the
41     // applicant number assigned to job i,
42     // the value -1 indicates nobody is
43     // assigned.
44     int matchR[N];
45     // Initially all jobs are available
46     memset(matchR, -1, sizeof(matchR));
47     // Count of jobs assigned to applicants
48     int result = 0;
49     for (int u = 0; u < M; u++) {
50         // Mark all jobs as not seen
51         // for next applicant.
52         bool seen[N];
53         memset(seen, 0, sizeof(seen));
54         // Find if the applicant 'u' can get a job

```

```

54     if (bpm(bpGraph, u, seen, matchR)) result++;
55 }
56 return result;
57 }
58
59 int main() {
60     bool bpGraph[M][N] = {{0, 1, 1, 0, 0, 0},
61                             {1, 0, 0, 1, 0, 0},
62                             {0, 0, 1, 0, 0, 0},
63                             {0, 0, 1, 1, 0, 0},
64                             {0, 0, 0, 0, 0, 0},
65                             {0, 0, 0, 0, 0, 1}};
66
67     cout << "Maximum number of applicants that can get job is "
68           << maxBPM(bpGraph);

```

8.7 Hungarian

```

1 // Time: O(VE)
2 const int INF = 2e9;
3 const int N = ? ; // 男女總人數；女 id: 0 ~ p, 男 id: p
4 // +1 ~ N-1
5 int vis[N], rnd, m[N]; // 跑完匈牙利後配對結果儲存於此， -1
6 // 表示人醜
7 vector<int> g[N]; // 關係表
8 int dfs(int s) {
9     for (int x : g[s]) {
10         if (vis[x]) continue;
11         vis[x] = 1;
12         if (m[x] == -1 || dfs(m[x])) {
13             m[x] = s, m[s] = x;
14             return 1;
15         }
16     }
17     return 0;
18 }
19
20 int hungarian(int p) { // p : 女性人數
21     memset(m, -1, sizeof(m));
22     int c = 0;
23     for (int i = 0; i < p; i++) {
24         if (m[i] == -1) {
25             memset(vis, 0, sizeof(vis));
26             c += dfs(i);
27         }
28     }
29     return c; // 成功結婚對數
30 }
31
32 // 匈牙利算法的優化，二分圖最大匹配 O(EV)
33 int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
34 vector<int> edge[maxn]; // for Left
35 bool dfs(int u) {
36     vis[u] = true;
37     for (vector<int>::iterator it = edge[u].begin();
38         it != edge[u].end(); ++it) {
39         int v = pr2[*it];
40         if (v == -1 ||
41             (!vis[v] && level[u] < level[v] && dfs(v))) {
42             pr[u] = *it, pr2[*it] = u;
43             return true;
44         }
45     }
46 }

```

```

42     } return false;
43 }
44 int hopcroftKarp() {
45     memset(pr, -1, sizeof(pr));
46     memset(pr2, -1, sizeof(pr2));
47     for (int match = 0;;) {
48         queue<int> Q;
49         for (int i = 1; i <= n; ++i) {
50             if (pr[i] == -1) level[i] = 0, Q.push(i);
51             else level[i] = -1;
52         }
53         while (!Q.empty()) {
54             int u = Q.front(); Q.pop();
55             for (vector<int>::iterator it = edge[u].begin();
56                 it != edge[u].end(); ++it) {
57                 int v = pr2[*it];
58                 if (v != -1 && level[v] < 0)
59                     level[v] = level[u] + 1, Q.push(v);
60             }
61         }
62         for (int i = 1; i <= n; ++i) vis[i] = false;
63         int d = 0;
64         for (int i = 1; i <= n; ++i)
65             if (pr[i] == -1 && dfs(i)) ++d;
66         if (d == 0) return match;
67         match += d;
68     }
69 }

```

8.8 EulerCircuit

1 無向圖：如果恰有兩點的度數為奇數，則存在歐拉路徑，此二點分別為起終點；如果全部的點度數都是偶數，則存在歐拉迴路。

2. 有向圖：如果恰有一點の出度等於入度+1、另有一點的入度等於出度+1，其餘皆入度等於出度，則存在歐拉路徑，此二點分別為起終點；如果全部的點入度等於出度，則存在歐拉迴路。

9 sortingAndSearching

9.1 NestedRangeCheck

```

1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3", "unroll-loops")
3 #define IO cin.tie(0), ios::sync_with_stdio(0)
4 using namespace std;
5 #define int long long
6 typedef pair<int, int> pii;
7
8 int32_t main() {
9     IO;
10     int t, r;
11     cin >> t;
12     vector<int> a(t), b(t), aa(t), bb(t);
13     vector<pii> v(t);
14     map<pii, int> mp;
15 }

```

```

16 for(auto &[l, r]: v){
17     cin >> l >> r;
18     r *= -1;
19     mp[{l,-r}] = mp.size();
20 }
21 sort(v.begin(),v.end());
22
23 for(auto &n: v) n.second *= -1 ;
24
25 r = v[0].second;
26 for(int i = 1; i < t; i++){
27     if(v[i].second <= r) a[i] = 1;
28     else if(v[i].second > r) a[i] = 0 , r = v[i].second;
29     if(v[i-1] == v[i]) a[i-1] = a[i] = 1;
30 }
31
32 r = v.back().second;
33 for(int i = v.size() - 2 ; i > -1 ; i--){
34     if(v[i].second >= r) b[i] = 1;
35     else if(v[i].second < r) b[i] = 0 , r = v[i].second;
36     if(v[i] == v[i+1]) b[i+1] = b[i] = 1;
37 }
38
39 for(int i = 0; i < t; i++){
40     aa[mp[v[i]]] = a[i];
41     bb[mp[v[i]]] = b[i];
42 }
43 for(int i = 0; i < t; i++) cout << bb[i] << " " ;
44 cout << endl ;
45 for(int i = 0; i < t; i++) cout << aa[i] << " " ;
46 cout << endl ;
47 }

```

9.2 NestedRangesCount

```

1 #include <bits/stdc++.h>
2 #include <ext/pb_ds/assoc_container.hpp>
3 #include <ext/pb_ds/tree_policy.hpp>
4 using namespace __gnu_pbds;
5 #define multiset tree<int, null_type, less_equal<int>,
6     rb_tree_tag, tree_order_statistics_node_update>
7 /**
8 order_of_key(k) : nums strictly smaller than k
9 find_by_order(k): index from 0
10 */
11 #pragma GCC optimize("O3","unroll-loops")
12 #define IO cin.tie(0), ios::sync_with_stdio(0)
13 using namespace std;
14 #define int long long
15 typedef pair<int,int> pii;
16
17 int32_t main(){
18     IO;
19     int t, r;
20     cin >> t;
21     vector<int> a(t), b(t), aa(t), bb(t);
22     vector<pii> v(t);
23     map<pii,int> mp;
24
25     for(auto &[l, r]: v){
26         cin >> l >> r;
27         r *= -1;
28         mp[{l,-r}] = mp.size();

```

```

28 }
29 sort(v.begin(),v.end());
30
31 for(auto &n: v) n.second *= -1 ;
32
33 multiset cnt;
34 r = v[0].second;
35 for(int i = 0; i < t; i++){
36     cnt.insert(v[i].second);
37     a[i] = cnt.size() - cnt.order_of_key(v[i].second)-1;
38 }
39
40 cnt.clear();
41 r = v.back().second;
42 for(int i = v.size() - 1 ; i > -1 ; i--){
43     cnt.insert(v[i].second);
44     b[i] = cnt.order_of_key(v[i].second+1)-1;
45 }
46
47 for(int i = 0; i < t; i++){
48     aa[mp[v[i]]] = a[i];
49     bb[mp[v[i]]] = b[i];
50 }
51 for(int i = 0; i < t; i++) cout << bb[i] << " " ;
52 cout << endl ;
53 for(int i = 0; i < t; i++) cout << aa[i] << " " ;
54 cout << endl ;
55 }

```

9.3 SubarrayDistinctVal

```

1 Given an array of n integers, your task is to calculate the
2   number of subarrays that have at most k distinct values.
3
4 Input
5 The first input line has two integers n and k.
6
7 Output
8 Print one integer: the number of subarrays.
9
10 Input:
11 5 2
12 1 2 3 1 1
13 Output:
14 10
15
16 #include <bits/stdc++.h>
17 #pragma GCC optimize("O3","unroll-loops")
18 #define IO cin.tie(0), ios::sync_with_stdio(0)
19 using namespace std;
20 #define int long long
21
22 int32_t main(){
23     int n, t, l, c = 0, ans = 0; cin >> n >> t;
24     vector<int> v(n);
25     map<int,int> mp;
26     for(auto&n:v)cin>>n;
27     l = 0;
28     for(int i = 0 ; i < n ;i++){
29         if(!mp[v[i]]) c ++;
30         mp[v[i]] ++;
31         while(c > t){
32             mp[v[l]] -- ;

```

```

32         if(!mp[v[l]]) c--;
33         l++;
34     }
35     ans += i - l + 1;
36 }
37 cout << ans << endl ;
38 }

```

9.4 missingCoinSum

```

1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3","unroll-loops")
3 #define IO cin.tie(0), ios::sync_with_stdio(0)
4 using namespace std;
5 typedef long long ll;
6
7 int main(){
8     IO;
9     ll n ; cin >> n ;
10    vector<ll> v(n) ;
11    for(auto&x:v) cin >> x ;
12    sort(v.begin(),v.end());
13    ll i=v[0] , j = 1 ;
14    if(i>1){
15        cout << 1 << endl ;
16        return 0 ;
17    }
18    for(;j<n;j++){
19        if(v[j]>i+1) break;
20        i+=v[j];
21    }
22    cout << (ll)(i+1) << endl ;
23 }

```

9.5 sumOfFourValues

```

1 #include <bits/stdc++.h>
2 #pragma GCC optimize("O3","unroll-loops")
3 #define IO cin.tie(0), ios::sync_with_stdio(0)
4 using namespace std;
5 #define int long long
6 #define pii pair<int,int>
7
8 int32_t main(){
9     int n, t; cin >> n >> t;
10    vector<int> v(n);
11    for(auto&n:v) cin >> n;
12    map<int,pii> mp;
13    for(int i = 0; i < n; i++){
14        for(int j = i+1; j < n; j++){
15            if(mp.count(t-v[i]-v[j])){
16                cout << mp[t-v[i]-v[j]].first+1 << " " << mp[
17                    t-v[i]-v[j]].second+1 << " " << i+1 << "
18                    " << j+1 << endl;
19                return 0;
20            }
21        }
22    }
23    for(int j = 0; j < i; j++)mp[v[i]+v[j]] = {i,j};
24 }

```

```

23     cout << "IMPOSSIBLE" << endl;
24 }
25
26 // 3 val
27 #include <bits/stdc++.h>
28 #pragma GCC optimize("O3","unroll-loops")
29 #define IO cin.tie(0), ios::sync_with_stdio(0)
30 #define int long long
31 using namespace std;
32 typedef pair<int,int> pii;
33
34 int ans = 0;
35 vector<pii> v;
36
37 int binarySearch(int l, int r, int target){
38     l++, r--;
39     while(l<=r){
40         int mid = (l+r)/2;
41         if(v[mid].first == target){
42             ans = 1;
43             return v[mid].second;
44         }
45         else if(v[mid].first < target) l = mid+1;
46         else r = mid-1;
47     }
48     return -1;
49 }
50
51 int32_t main(){
52     int n, t, l, r, m; cin >> n >> t;
53     v.resize(n);
54     for(int i = 0; i < n; i++){
55         cin >> v[i].first;
56         v[i].second = i+1;
57     }
58     sort(v.begin(), v.end());
59
60     for(l = 0; l < n; l++)
61         for(r = l+1; r < n; r++){
62             m = binarySearch( r, n, t - v[l].first - v[r].first);
63             if(ans){
64                 cout << v[l].second << " " << m << " " << v[r].second << endl;
65                 return 0;
66             }
67         }
68     cout << "IMPOSSIBLE\n";
69 }

```

10 Other

10.1 精度

```

1 from decimal import*
2 getcontext().prec = 1000000
3 n = Decimal(input())

```

10.2 inversion

```

1 // 1
2 int inversion(){
3     ll Length, now, ans = 0;
4     Multiset place;
5     cin >> Length;
6     for(int i = 0; i < Length; i++){
7         cin >> now;
8         place.insert(now);
9         ans += i - place.order_of_key(now);
10    }
11    return ans;
12 }
13 // 2
14 void Merge(int a[], int b, int m, int e){
15     int ap = b, bp = m+1, cp = 0, sum = 0;
16     static int c[MAX];
17     while(ap <= m && bp <= e){
18         if(a[ap] <= a[bp]) c[cp++] = a[ap++], ans += sum;
19         else c[cp++] = a[bp++], sum++;
20     }
21     while(ap <= m) c[cp++] = a[ap++], ans += sum;
22     while(bp <= e) c[cp++] = a[bp++];
23     for(int k = 0; k < cp; k++) a[b+k] = c[k];
24 }
25 void Merge_Sort(int a[], int b, int e){
26     if(e - b < 1) return;
27     int m = (b + e) / 2;
28     Merge_Sort(a, b, m);
29     Merge_Sort(a, m+1, e);
30     Merge(a, b, m, e);
31 }

```

10.3 離散化

```

1 vector<int> v(1000), b(1000);
2 for(auto&n:v) cin >> n;
3 sort(v.begin(), v.end());
4 auto len = unique(v.begin(), v.end()) - v.begin();
5 v.resize(len);
6 for(int i = 0; i < v.size(); i++){
7     b[i] = lower_bound(v.begin(), v.end(), v[i]) - v.begin();
8 }

```

10.4 Convert

```

1 stoi
2 stoll
3 to_string

```

10.5 minSwap

```

1 int minSwaps(vector<int> &arr, int n) {
2     pii arrPos[n];

```

```

3     for (int i = 0; i < n; i++) {
4         arrPos[i].first = arr[i];
5         arrPos[i].second = i;
6     }
7     sort(arrPos, arrPos + n);
8     vector<bool> vis(n, false);
9     int ans = 0;
10    for (int i = 0; i < n; i++) {
11        if (vis[i] or arrPos[i].second == i) continue;
12        int cycle_size = 0, j = i;
13        while (!vis[j]){
14            vis[j] = 1;
15            j = arrPos[j].second;
16            cycle_size++;
17        }
18        if(cycle_size > 0) ans += (cycle_size - 1);
19    }
20    return ans;
21 }

```

10.6 莫隊

```

1 #include "bits/stdc++.h"
2 using namespace std;
3
4 #define ll long long
5 #define pii pair<int, int>
6 #define ql first.first
7 #define qr first.second
8 #define id second
9
10 int N, Q, K;
11 vector<int> A;
12 vector<pair<pii, int>> qrys;
13 ll ans[200007];
14
15 void init(){
16     cin >> N >> Q;
17     K = N / max((int)sqrt(Q), 1); // 塊的大小
18
19     A.clear(); A.resize(N);
20     qrys.clear(); qrys.resize(Q);
21
22     for(auto&i : A) cin >> i;
23     int cnt = 0;
24     for(auto&i : qrys){
25         cin >> i.ql >> i.qr;
26         --i.ql; --i.qr; // 轉換成 0-base
27         i.id = cnt++;
28     }
29 }
30
31 void solve(){
32     sort(qrys.begin(), qrys.end(),
33         [&](const pair<pii, int> &a, const pair<pii, int> &b){
34             if(a.ql/K == b.ql/K) // 左界塊相同，按照右界塊排序
35                 return a.qr < b.qr;
36             return a.ql/K < b.ql/K; // 否則，按照左界塊排序
37         });
38
39     int l = 0, r = -1; // 初始答案窗口
40     ll sum = 0;

```

```
41 |
42 | for(auto&i : qrys){
43 |     while(l > i.ql){ l--; sum += A[l]; } // 延伸左界
44 |     while(r < i.qr){ r++; sum += A[r]; } // 延伸右界
45 |     while(l < i.ql){ sum -= A[l]; l++; } // 内縮左界
46 |     while(r > i.qr){ sum -= A[r]; r--; } // 内縮右界
47 |     ans[i.id] = sum;
48 | }
49 |
50 | for(int i = 0; i < Q; i++) cout << ans[i] << '\n';
51 | }
52 |
53 | int main(){
54 |     init();
55 |     solve();
56 | }
```

MCU-NL CODEBOOK

Contents

1 Setup	1	5 Graph	5	8 Geometry	11
1.1 Setup	1	5.1 Dijkstra	5	8.1 angle	11
1.2 template	1	5.2 BellmanFord	5	8.2 ConvexHull	11
2 IO	1	5.3 SPFA	5	8.3 旋轉卡尺	11
2.1 IO	1	5.4 MaxFlow	6	8.4 closestPair	12
3 Data_Structure	1	5.5 EdmondsKarp	6	8.5 MinCircle	13
3.1 線段樹	1	6 Math	6	8.6 MaximumBipartiteMatching	14
3.2 區間修改線段樹	1	6.1 Bignumber	6	8.7 Hungarian	14
3.3 Kurskal	2	6.2 BignumberDec	7	8.8 EulerCircuit	14
3.4 Prim	2	6.3 線篩	8	9 sortingAndSearching	14
3.5 DAG	2	6.4 Fib	8	9.1 NestedRangeCheck	14
3.6 Unionfind	3	6.5 鞋帶	8	9.2 NestedRangesCount	15
3.7 CDQ	3	6.6 SG	8	9.3 SubarrayDistinctVal	15
3.8 DSU	4	6.7 擴展歐幾里德	8	9.4 missingCoinSum	15
4 DP	4	6.8 FPow	8	9.5 sumOfFourValues	15
4.1 LIS	4	6.9 質因數分解	8	10 Other	16
4.2 LCS2LIS	4	6.10 Expression	9	10.1 精度	16
4.3 LCS3D	5	6.11 JosephusProblem	9	10.2 inversion	16
		7 String	10	10.3 離散化	16
		7.1 Trie	10	10.4 Convert	16
		7.2 AC 自動機	10	10.5 minSwap	16
		7.3 KMP	10	10.6 莫隊	16
		4.4 LCS	5		
		4.5 countingTours	5		
		4.6 retCutting	5		
		7.4 LPS	11		
		7.5 EditDistance	11		