1 Surroundings

1.1 setup

```
1 測機 (test on C++ and Python)
2 AC：好好寫
3 WA：cout << "0\n" / 結尾多印一行；
4 RE：空間越界/除0
5 TLE：while(true);
6 CE：empty code
7 OLE：瘋狂Hello World
8 NO Output：default code
9 待測：stack深度、judge速度、陣列MAX
10 開賽
11 1. bash.rc打ac
12 2. 調gedit設定
13 3. 打default_code
14 4. 測試ac
```

1.2 bashrc

```
1 oj() {
2     ext=${1##*.}           #空格敏感
3     filename=${1##*/}      #空格敏感
4     filename=${filename%.*} #空格敏感
5     case $ext in
6     cpp ) g++ -o "/tmp/$filename" "$1" && "/tmp/$filename" ;;
                            #空格不敏感
7     py  ) python3 "$1" ;;
                                           #空格不敏感
8     esac
9 }
```

1.3 vimrc

```
1 set tabstop=4
2 set shiftwidth=4
3 set softtabstop=4
4 set expandtab
5 set autoindent
6 set number
```

2 Data_Structure

2.1 Sparse Table

```
1 // https://judge.yosupo.jp/problem/staticrmq 214 ms
2
3 template<typename T, int RANGE>
4 struct Sparse_Table {
5   struct Node {
6     T val;
7
8     Node(): val(INF) {}
9
10    Node operator +(const Node &rhs) {
11      Node ret;
12      ret.val = min(val, rhs.val);
13      return ret; // 視情況修改
14    }
15  };
16  vector<vector<Node>> arr;
17
18  Sparse_Table() {
19    arr.resize(__lg(RANGE) + 1, vector<Node>(RANGE, Node()));
20  }
21
22  void build(auto &v) {
23    for (int i = 1; i <= n; i++) {
24      arr[0][i].val = v[i];
25    }
26    for (int i = 1; i <= __lg(n); i++)
27      for (int j = 1; j + (1 << (i - 1)) <= n; j++)
28        arr[i][j] = arr[i - 1][j] + arr[i - 1][j + (1 << (i - 1))];
29  }
30
31  Node query(int ql, int qr) {
32    int lg = __lg(qr - ql + 1);
33    return arr[lg][ql] + arr[lg][qr - (1 << lg) + 1];
34  }
35 };
```

2.2 Fenwick Tree

```
1 /** 普通 BIT ，為了加速打字只支援 1-based **/
2 const int MAXN = ? ;  // 開全域加速打字
3 #define lowbit(x & (-x))
4
5
6 template<typename T>
7 struct Fenwick_Tree { // 1 based
8   // 二維：陣列開二維，修改跟查詢就是對 (x, y) 各自 +-
            lowbit
9   T arr[MAXN];
10  void init(int _n = MAXN) {
11    for (int i = 0; i < _n; i++)
12      arr[i] = 0;
13  }
14  void update(int i, T val) {
15    for (; i < MAXN; i += lowbit(i))
16      arr[i] += val;
17  }
18  T query(int i) {
19    T ret = 0;
20    for (; i; i -= lowbit(i))
21      ret += arr[i];
22    return ret;
23  }
24 };
```

2.3 單點修改、區間查詢線段樹

```
1 // https://judge.yosupo.jp/problem/point_add_range_sum 331 ms
2 // https://judge.yosupo.jp/problem/staticrmq 359 ms
3 template<typename T, int RANGE>
4 struct Segment_Tree {
5   struct Node {
6     T val;
7
8     Node (): val(0) {} // mx: -INF, mn: INF, sum: 0, gcd: 1,
          lcm: 1
9
10    Node operator +(const Node &rhs) {
11      Node ret;
12      ret.val = val + rhs.val; // 對應不同操作修改
13
14      return ret;
15    }
16
17    void update(int _val) {
18      val += _val;
19    }
20  };
21
22  vector<Node> arr;
23
24  Segment_Tree() {
25    arr.resize(RANGE << 2);
26  }
27
28  void build(vector<int> &v, int i = 1, int l = 1, int r = n)
        {
29    if (l == r) {
30      arr[i].val = v[l];
31      return;
32    }
33    int mid = (l + r) >> 1;
34    build(v, i << 1, l, mid);
35    build(v, i << 1 | 1, mid + 1, r);
36    arr[i] = arr[i << 1] + arr[i << 1 | 1];
37  }
38
39  void update(int pos, int val, int i = 1, int l = 1, int r =
        n) {
40    if (l == r) {
41      arr[i].update(val);
42      return;
43    }
44    int mid = (l + r) >> 1;
45    if (pos <= mid) update(pos, val, i << 1, l, mid);
46    else update(pos, val, i << 1 | 1, mid + 1, r);
47    arr[i] = arr[i << 1] + arr[i << 1 | 1];
48  }
49
50  Node query(int ql, int qr, int i = 1, int l = 1, int r = n)
        {
51    if (l > qr || r < ql)
52      return Node();
53    if (ql <= l && r <= qr)
54      return arr[i];
55    int mid = (l + r) >> 1;
56    return query(ql, qr, i << 1, l, mid) + query(ql, qr, i <<
          1 | 1, mid + 1, r);
57  }
58 };
```

2.4 最大區間和線段樹

```cpp
/** 計算最大子區間連續和的線段樹，限定 1-based 。
 * 複雜度 O(Q*log(N)) **/
#define ls i << 1
#define rs i << 1 | 1
class MaxSumSegmentTree {
    private:
    struct node {
        ll lss, rss, ss, ans;
        void set(ll v) { lss = rss = ss = ans = v; }
    };
    int n;
    vector<node> a;  // 萬萬不可用普通陣列，要用 vector
    vector<ll> z;
    void pull(int i) {
        a[i].ss = a[ls].ss + a[rs].ss;
        a[i].lss = max(a[ls].lss, a[ls].ss + a[rs].lss);
        a[i].rss = max(a[rs].rss, a[rs].ss + a[ls].rss);
        a[i].ans = max(max(a[ls].ans, a[rs].ans),
                        a[ls].rss + a[rs].lss);
    }
    void build(int i, int l, int r) {
        if (l == r) return a[i].set(z[l]), void();
        int m = (l + r) >> 1;
        build(ls, l, m), build(rs, m + 1, r), pull(i);
    }
    void set(int i, int l, int r, int q, ll v) {
        if (l == r) return a[i].set(v), void();
        int m = (l + r) >> 1;
        if (q <= m) set(ls, l, m, q, v);
        else set(rs, m + 1, r, q, v);
        pull(i);
    }
    node query(int i, int l, int r, int ql, int qr) {
        if (ql <= l && r <= qr) return a[i];
        int m = (l + r) >> 1;
        if (qr <= m) return query(ls, l, m, ql, qr);
        if (m < ql) return query(rs, m + 1, r, ql, qr);
        node lo = query(ls, l, m, ql, qr),
             ro = query(rs, m + 1, r, ql, qr), ans;
        ans.ss = lo.ss + ro.ss;
        ans.lss = max(lo.lss, lo.ss + ro.lss);
        ans.rss = max(ro.rss, ro.ss + lo.rss);
        ans.ans = max(max(lo.ans, ro.ans), lo.rss + ro.lss);
        return ans;
    }

    public:
    MaxSumSegmentTree(int n) : n(n) {
        a.resize(n << 2), z.resize(n << 2);
        build(1, 1, n);
    }
    // 單點設值。限定 1-based 。
    inline void set(int i, ll v) { set(1, 1, n, i, v); }
    // 問必區間 [l, r] 的最大子區間連續和。限定 1-based 。
    inline ll query(int l, int r) {
        return query(1, 1, n, l, r).ans;
    }
};
```

2.5 懶標線段樹

```cpp
struct Node {
    int sum, tag;
    Node(): sum(0), tag(0) {}

    void update(int val, int l, int r) {
        sum += (val) * (r - l + 1);
        tag += val;
    }
    Node operator +(const Node rhs) {
        Node ret;
        ret.sum = sum + rhs.sum;
        return ret;
    }
    void operator *=(const Node rhs) {
        sum = rhs.sum;
    }
};

template<typename T>
struct Segment_Tree {
    vector<T> arr;

    void init() {
        arr.resize(MAXN << 2, Node());
    }

    void push(int i, int l, int r) {
        if (l == r || arr[i].tag == 0)
            return;
        int mid = (l + r) / 2;
        arr[i * 2].update(arr[i].tag, l, mid);
        arr[i * 2 + 1].update(arr[i].tag, mid + 1, r);
        arr[i].tag = 0;
    }

    void update(int ql, int qr, int val, int i = 1, int l =
        1, int r = n) {
        if (ql <= l && r <= qr) {
            arr[i].update(val, l, r);
            return;
        }
        if (l > qr || r < ql) {
            return;
        }
        int mid = (l + r) / 2;
        push(i, l, r);
        update(ql, qr, val, i * 2, l, mid);
        update(ql, qr, val, i * 2 + 1, mid + 1, r);
        arr[i].sum = (arr[i * 2] + arr[i * 2 + 1]).sum;
    }

    T query(int ql, int qr, int i = 1, int l = 1, int r = n)
        {
        if (ql <= l && r <= qr)
            return arr[i];
        if (l > qr || r < ql)
            return T();
        push(i, l, r);
        int mid = (l + r) / 2;
        auto q1 = query(ql, qr, i * 2, l, mid);
        auto q2 = query(ql, qr, i * 2 + 1, mid + 1, r);
        return q1 + q2;
    }
};
```

2.6 持久化線段樹

```cpp
int a[maxn], b[maxn], root[maxn], cnt;
struct node {
    int sum, L_son, R_son;
} tree[maxn << 5];
int create(int _sum, int _L_son, int _R_son) {
    int idx = ++cnt;
    tree[idx].sum = _sum, tree[idx].L_son = _L_son, tree[idx
        ].R_son = _R_son;
    return idx;
}
void Insert(int &root, int pre_rt, int pos, int L, int R) {
    root = create(tree[pre_rt].sum+1, tree[pre_rt].L_son,
        tree[pre_rt].R_son);
    if(L==R) return;
    int M = (L+R)>>1;
    if(pos<=M) Insert(tree[root].L_son, tree[pre_rt].L_son,
        pos, L, M);
    else Insert(tree[root].R_son, tree[pre_rt].R_son, pos, M
        +1, R);
}
int query(int L_id, int R_id, int L, int R, int K) {
    if(L==R) return L;
    int M = (L+R)>>1;
    int s = tree[tree[R_id].L_son].sum - tree[tree[L_id].
        L_son].sum;
    if(K<=s) return query(tree[L_id].L_son, tree[R_id].L_son,
        L, M, K);
    return query(tree[L_id].R_son, tree[R_id].R_son, M+1, R,
        K-s);
}
int main() {
    int n,m; cin >> n >> m
    for(int i=1; i<=n; i++) {
        cin >> a[i]; b[i] = a[i];
    } sort(b+1,b+1+n); //離散化
    int b_sz = unique(b+1, b+1+n) - (b+1);
    cnt = root[0] = 0;
    for(int i=1; i<=n; i++) {
        int pos = lower_bound(b+1, b+1+b_sz, a[i]) - b;
        Insert(root[i], root[i-1], pos, 1, b_sz);
    }
    while(m--) {
        int l, r, k; cin >> l >> r >> k;
        int pos = query(root[l-1],root[r],1,b_sz,k);
        cout << b[pos] << endl;
    } return 0;
}
```

2.7 李超線段樹

```cpp
template<typename T>
struct LiChao_SegTree {
    T arr[MAXM << 2];

    void init() {
        for (int i = 0; i < (MAXM << 2); i++) {
            arr[i] = {m, 0};
        }
    }
```

```
11    void insert(int i, int l, int r, T x) {
12        if (l == r) {
13            if (x(l) < arr[i](l)) {
14                arr[i] = x;
15            }
16            return;
17        }
18
19        if (arr[i].a > x.a) {
20            swap(arr[i], x);
21        }
22
23        int mid = (l + r) / 2;
24
25        if (x(mid) > arr[i](mid)) {
26            insert(i * 2, l, mid, x);
27        }
28        else {
29            swap(arr[i], x);
30            insert(i * 2 + 1, mid + 1, r, x);
31        }
32    }
33
34    int query(int i, int l, int r, int pos) {
35        if (l == r)
36            return arr[i](pos);
37        int mid = (l + r) / 2;
38        int res;
39        if (pos <= mid) {
40            res = query(i * 2, l, mid, pos);
41        }
42        else {
43            res = query(i * 2 + 1, mid + 1, r, pos);
44        }
45        return min(res, arr[i](pos));
46    }
47 };
```

2.8   Treap

```
1  // 支援區間加值、區間反轉、區間 rotate 、區間刪除、插入元素、
       求區間
2  // 最小值的元素的 Treap。使用前建議 srand(time(0)); 除了 size
       ()
3  // 方法以外，所有操作都是 O(log N) 。所有 public 方法各自獨
       立，請
4  // 斟酌要使用到哪些方法，有需要的才抄。
5  class Treap {
6    private:
7      struct Node {
8          int pri = rand(), size = 1;
9          ll val, mn, inc = 0;
10         bool rev = 0;
11         Node *lc = 0, *rc = 0;
12         Node(ll v) { val = mn = v; }
13     };
14     Node* root = 0;
15     void rev(Node* t) {
16         if (!t) return;
17         swap(t->lc, t->rc), t->rev ^= 1;
18     }
19     void update(Node* t, ll v) {
```

```
20         if (!t) return;
21         t->val += v, t->inc += v, t->mn += v;
22     }
23     void push(Node* t) {
24         if (t->rev) rev(t->lc), rev(t->rc), t->rev = 0;
25         update(t->lc, t->inc), update(t->rc, t->inc);
26         t->inc = 0;
27     }
28     void pull(Node* t) {
29         t->size = 1 + size(t->lc) + size(t->rc);
30         t->mn = t->val;
31         if (t->lc) t->mn = min(t->mn, t->lc->mn);
32         if (t->rc) t->mn = min(t->mn, t->rc->mn);
33     }
34     // 看你要不要釋放記憶體
35     void discard(Node* t) {
36         if (!t) return;
37         discard(t->lc), discard(t->rc);
38         delete t;
39     }
40     void split(Node* t, Node*& a, Node*& b, int k) {
41         if (!t) return a = b = 0, void();
42         push(t);
43         if (size(t->lc) < k) {
44             a = t;
45             split(t->rc, a->rc, b, k - size(t->lc) - 1);
46             pull(a);
47         } else {
48             b = t;
49             split(t->lc, a, b->lc, k);
50             pull(b);
51         }
52     }
53     Node* merge(Node* a, Node* b) {
54         if (!a || !b) return a ? a : b;
55         if (a->pri > b->pri) {
56             push(a);
57             a->rc = merge(a->rc, b);
58             pull(a);
59             return a;
60         } else {
61             push(b);
62             b->lc = merge(a, b->lc);
63             pull(b);
64             return b;
65         }
66     }
67     inline int size(Node* t) { return t ? t->size : 0; }
68  public:
69     int size() { return size(root); }
70     void add(int l, int r, ll val) {
71         Node *a, *b, *c, *d;
72         split(root, a, b, r);
73         split(a, c, d, l - 1);
74         update(d, val);
75         root = merge(merge(c, d), b);
76     }
77     // 反轉區間 [l, r]
78     void reverse(int l, int r) {
79         Node *a, *b, *c, *d;
80         split(root, a, b, r);
81         split(a, c, d, l - 1);
82         swap(d->lc, d->rc);
83         d->rev ^= 1;
84         root = merge(merge(c, d), b);
```

```
85     }
86     // 區間 [l, r] 向右 rotate k 次，k < 0 表向左 rotate
87     void rotate(int l, int r, int k) {
88         int len = r - l + 1;
89         Node *a, *b, *c, *d, *e, *f;
90         split(root, a, b, r);
91         split(a, c, d, l - 1);
92         k = (k + len) % len;
93         split(d, e, f, len - k);
94         root = merge(merge(c, merge(f, e)), b);
95     }
96     // 插入一個元素 val 使其 index = i
97     // 注意 i <= size
98     void insert(int i, ll val) {
99         if (i == size() + 1) {
100            push_back(val);
101            return;
102        }
103        assert(i <= size());
104        Node *a, *b;
105        split(root, a, b, i - 1);
106        root = merge(merge(a, new Node(val)), b);
107    }
108    void push_back(ll val) {
109        root = merge(root, new Node(val));
110    }
111    void remove(int l, int r) {
112        int len = r - l + 1;
113        Node *a, *b, *c, *d;
114        split(root, a, b, l - 1);
115        split(b, c, d, len);
116        discard(c); // 看你要不要釋放記憶體
117        root = merge(a, d);
118    }
119    ll minn(int l, int r) {
120        Node *a, *b, *c, *d;
121        split(root, a, b, r);
122        split(a, c, d, l - 1);
123        int ans = d->mn;
124        root = merge(merge(c, d), b);
125        return ans;
126    }
127 };
```

2.9   Dynamic_KD_tree

```
1  template<typename T,size_t kd>//有kd個維度
2  struct kd_tree{
3    struct point{
4      T d[kd];
5      T dist(const point &x)const{
6        T ret=0;
7        for(size_t i=0;i<kd;++i)ret+=abs(d[i]-x.d[i]);
8        return ret;
9      }
10     bool operator==(const point &p){
11       for(size_t i=0;i<kd;++i)
12         if(d[i]!=p.d[i])return 0;
13       return 1;
14     }
15     bool operator<(const point &b)const{
16       return d[0]<b.d[0];
```

```
17          }
18      };
19  private:
20      struct node{
21          node *l,*r;
22          point pid;
23          int s;
24          node(const point &p):l(0),r(0),pid(p),s(1){}
25          ~node(){delete l,delete r;}
26          void up(){s=(l?l->s:0)+1+(r?r->s:0);}
27      }*root;
28      const double alpha,loga;
29      const T INF;//記得要給INF，表示極大值
30      int maxn;
31      struct __cmp{
32          int sort_id;
33          bool operator()(const node*x,const node*y)const{
34              return operator()(x->pid,y->pid);
35          }
36          bool operator()(const point &x,const point &y)const{
37              if(x.d[sort_id]!=y.d[sort_id])
38                  return x.d[sort_id]<y.d[sort_id];
39              for(size_t i=0;i<kd;++i)
40                  if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
41              return 0;
42          }
43      }cmp;
44      int size(node *o){return o?o->s:0;}
45      vector<node*> A;
46      node* build(int k,int l,int r){
47          if(l>r) return 0;
48          if(k==kd) k=0;
49          int mid=(l+r)/2;
50          cmp.sort_id = k;
51          nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
52          node *ret=A[mid];
53          ret->l = build(k+1,l,mid-1);
54          ret->r = build(k+1,mid+1,r);
55          ret->up();
56          return ret;
57      }
58      bool isbad(node*o){
59          return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
60      }
61      void flatten(node *u,typename vector<node*>::iterator &it){
62          if(!u)return;
63          flatten(u->l,it);
64          *it=u;
65          flatten(u->r,++it);
66      }
67      void rebuild(node*&u,int k){
68          if((int)A.size()<u->s)A.resize(u->s);
69          auto it=A.begin();
70          flatten(u,it);
71          u=build(k,0,u->s-1);
72      }
73      bool insert(node*&u,int k,const point &x,int dep){
74          if(!u) return u=new node(x), dep<=0;
75          ++u->s;
76          cmp.sort_id=k;
77          if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
78              if(!isbad(u))return 1;
79              rebuild(u,k);
80          }
81          return 0;
```

```
82      }
83      node *findmin(node*o,int k){
84          if(!o)return 0;
85          if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
86          node *l=findmin(o->l,(k+1)%kd);
87          node *r=findmin(o->r,(k+1)%kd);
88          if(l&&!r)return cmp(l,o)?l:o;
89          if(!l&&r)return cmp(r,o)?r:o;
90          if(!l&&!r)return o;
91          if(cmp(l,r))return cmp(l,o)?l:o;
92          return cmp(r,o)?r:o;
93      }
94      bool erase(node *&u,int k,const point &x){
95          if(!u)return 0;
96          if(u->pid==x){
97              if(u->r);
98              else if(u->l) u->r=u->l, u->l=0;
99              else return delete(u),u=0, 1;
100             --u->s;
101             cmp.sort_id=k;
102             u->pid=findmin(u->r,(k+1)%kd)->pid;
103             return erase(u->r,(k+1)%kd,u->pid);
104         }
105         cmp.sort_id=k;
106         if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x))
107             return --u->s, 1;
108         return 0;
109     }
110     T heuristic(const T h[])const{
111         T ret=0;
112         for(size_t i=0;i<kd;++i)ret+=h[i];
113         return ret;
114     }
115     int qM;
116     priority_queue<pair<T,point>> pQ;
117     void nearest(node *u,int k,const point &x,T *h,T &mndist){
118         if(u==0||heuristic(h)>=mndist)return;
119         T dist=u->pid.dist(x),old=h[k];
120         /*mndist=std::min(mndist,dist);*/
121         if(dist<mndist){
122             pQ.push(std::make_pair(dist,u->pid));
123             if((int)pQ.size()==qM+1)
124                 mndist=pQ.top().first,pQ.pop();
125         }
126         if(x.d[k]<u->pid.d[k]){
127             nearest(u->l,(k+1)%kd,x,h,mndist);
128             h[k] = abs(x.d[k]-u->pid.d[k]);
129             nearest(u->r,(k+1)%kd,x,h,mndist);
130         }else{
131             nearest(u->r,(k+1)%kd,x,h,mndist);
132             h[k] = abs(x.d[k]-u->pid.d[k]);
133             nearest(u->l,(k+1)%kd,x,h,mndist);
134         }
135         h[k]=old;
136     }
137     vector<point>in_range;
138     void range(node *u,int k,const point&mi,const point&ma){
139         if(!u)return;
140         bool is=1;
141         for(int i=0;i<kd;++i)
142             if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i])
143                 { is=0;break; }
144         if(is) in_range.push_back(u->pid);
145         if(mi.d[k]<=u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
146         if(ma.d[k]>=u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
147     }
```

```
148 public:
149     kd_tree(const T &INF,double a=0.75):
150     root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
151     ~kd_tree(){delete root;}
152     void clear(){delete root,root=0,maxn=1;}
153     void build(int n,const point *p){
154         delete root,A.resize(maxn=n);
155         for(int i=0;i<n;++i)A[i]=new node(p[i]);
156         root=build(0,0,n-1);
157     }
158     void insert(const point &x){
159         insert(root,0,x,__lg(size(root))/loga);
160         if(root->s>maxn)maxn=root->s;
161     }
162     bool erase(const point &p){
163         bool d=erase(root,0,p);
164         if(root&&root->s<alpha*maxn)rebuild();
165         return d;
166     }
167     void rebuild(){
168         if(root)rebuild(root,0);
169         maxn=root->s;
170     }
171     T nearest(const point &x,int k){
172         qM=k;
173         T mndist=INF,h[kd]={};
174         nearest(root,0,x,h,mndist);
175         mndist=pQ.top().first;
176         pQ = priority_queue<pair<T,point>>();
177         return mndist;//回傳離x第k近的點的距離
178     }
179     const vector<point> &range(const point&mi,const point&ma){
180         in_range.clear();
181         range(root,0,mi,ma);
182         return in_range;//回傳介於mi到ma之間的點vector
183     }
184     int size(){return root?root->s:0;}
185 };
```

2.10 Heavy Light

```
1   #include<vector>
2   #define MAXN 100005
3   int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
4   int link_top[MAXN],link[MAXN],cnt;
5   vector<int> G[MAXN];
6   void find_max_son(int u){
7       siz[u]=1;
8       max_son[u]=-1;
9       for(auto v:G[u]){
10          if(v==pa[u])continue;
11          pa[v]=u;
12          dep[v]=dep[u]+1;
13          find_max_son(v);
14          if(max_son[u]==-1||siz[v]>siz[max_son[u]])max_son[u]=v;
15          siz[u]+=siz[v];
16      }
17  }
18  void build_link(int u,int top){
19      link[u]=++cnt;
20      link_top[u]=top;
21      if(max_son[u]==-1)return;
```

```
22    build_link(max_son[u],top);
23    for(auto v:G[u]){
24      if(v==max_son[u]||v==pa[u])continue;
25      build_link(v,v);
26    }
27  }
28  int find_lca(int a,int b){
29    //求LCA，可以在過程中對區間進行處理
30    int ta=link_top[a],tb=link_top[b];
31    while(ta!=tb){
32      if(dep[ta]<dep[tb]){
33        swap(ta,tb);
34        swap(a,b);
35      }
36      //這裡可以對a所在的鏈做區間處理
37      //區間為(link[ta],link[a])
38      ta=link_top[a=pa[ta]];
39    }
40    //最後a,b會在同一條鏈，若a!=b還要在進行一次區間處理
41    return dep[a]<dep[b]?a:b;
42  }
```

2.11   HLD By Koying

```
1  //  https://cses.fi/problemset/task/1137/
2
3  struct HLD {
4      struct Info {
5          int sub, mxsub, dep, fa, root, id;
6      } arr[MAXN];
7
8      int index = 0;
9
10     void find_son(int i, int fa) {
11         pii mx(0, i);
12         arr[i].sub = 1;
13         for (auto it: G[i]) if (it != fa) {
14             arr[it].dep = arr[i].dep + 1;
15             arr[it].fa = i;
16             find_son(it, i);
17             cmax(mx, pii(arr[it].sub, it));
18             arr[i].sub += arr[it].sub;
19         }
20         arr[i].mxsub = mx.S;
21     }
22
23     void build(int i, int root) {
24         arr[i].root = root;
25         arr[i].id = ++index;
26         y[arr[i].id] = x[i];
27
28         if (arr[i].mxsub != i) {
29             build(arr[i].mxsub, root);
30             y[arr[i].id] += y[arr[arr[i].mxsub].id];
31         }
32
33         for (auto it: G[i]) if (it != arr[i].fa && it != arr[
             i].mxsub) {
34             build(it, it);
35             y[arr[i].id] += y[arr[it].id];
36         }
37     }
```

```
38
39     void jump(int a, int b) { // from a to b (dep(a) > dep(b)
            )
40         while (arr[a].root != arr[b].root) {
41             if (arr[arr[a].root].dep < arr[arr[b].root].dep)
42                 a = arr[arr[a].root].fa;
43         }
44
45         if (arr[a].dep < arr[b].dep)
46             swap(a, b);
47
48         return mx;
49     }
50 } HLD;
```

2.12   Link Cut Tree

```
1  struct splay_tree{
2    int ch[2],pa;//子節點跟父母
3    bool rev;//反轉的懶惰標記
4    splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5  };
6  vector<splay_tree> nd;
7  //有的時候用vector會TLE，要注意
8  //這邊以node[0]作為null節點
9  bool isroot(int x){//判斷是否為這棵splay tree的根
10   return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa].ch[1]!=x;
11 }
12 void down(int x){//懶惰標記下推
13   if(nd[x].rev){
14     if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
15     if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
16     swap(nd[x].ch[0],nd[x].ch[1]);
17     nd[x].rev=0;
18   }
19 }
20 void push_down(int x){//所有祖先懶惰標記下推
21   if(!isroot(x))push_down(nd[x].pa);
22   down(x);
23 }
24 void up(int x){}//將子節點的資訊向上更新
25 void rotate(int x){//旋轉，會自行判斷轉的方向
26   int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==x);
27   nd[x].pa=z;
28   if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
29   nd[y].ch[d]=nd[x].ch[d^1];
30   nd[nd[y].ch[d]].pa=y;
31   nd[y].pa=x,nd[x].ch[d^1]=y;
32   up(y),up(x);
33 }
34 void splay(int x){//將x伸展到splay tree的根
35   push_down(x);
36   while(!isroot(x)){
37     int y=nd[x].pa;
38     if(!isroot(y)){
39       int z=nd[y].pa;
40       if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))rotate(y);
41       else rotate(x);
42     }
43     rotate(x);
44   }
```

```
45 }
46 int access(int x){
47   int last=0;
48   while(x){
49     splay(x);
50     nd[x].ch[1]=last;
51     up(x);
52     last=x;
53     x=nd[x].pa;
54   }
55   return last;//access後splay tree的根
56 }
57 void access(int x,bool is=0){//is=0就是一般的access
58   int last=0;
59   while(x){
60     splay(x);
61     if(is&&!nd[x].pa){
62       //printf("%d\n",max(nd[last].ma,nd[nd[x].ch[1]].ma));
63     }
64     nd[x].ch[1]=last;
65     up(x);
66     last=x;
67     x=nd[x].pa;
68   }
69 }
70 void query_edge(int u,int v){
71   access(u);
72   access(v,1);
73 }
74 void make_root(int x){
75   access(x),splay(x);
76   nd[x].rev^=1;
77 }
78 void make_root(int x){
79   nd[access(x)].rev^=1;
80   splay(x);
81 }
82 void cut(int x,int y){
83   make_root(x);
84   access(y);
85   splay(y);
86   nd[y].ch[0]=0;
87   nd[x].pa=0;
88 }
89 void cut_parents(int x){
90   access(x);
91   splay(x);
92   nd[nd[x].ch[0]].pa=0;
93   nd[x].ch[0]=0;
94 }
95 void link(int x,int y){
96   make_root(x);
97   nd[x].pa=y;
98 }
99 int find_root(int x){
100   x=access(x);
101   while(nd[x].ch[0])x=nd[x].ch[0];
102   splay(x);
103   return x;
104 }
105 int query(int u,int v){
106   //傳回uv路徑splay tree的根結點
107   //這種寫法無法求LCA
108   make_root(u);
109   return access(v);
```

```
110 }
111 int query_lca(int u,int v){
112 //假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重
113   access(u);
114   int lca=access(v);
115   splay(u);
116   if(u==lca){
117     //return nd[lca].data+nd[nd[lca].ch[1]].sum
118   }else{
119     //return nd[lca].data+nd[nd[lca].ch[1]].sum+nd[u].sum
120   }
121 }
122 struct EDGE{
123   int a,b,w;
124 }e[10005];
125 int n;
126 vector<pair<int,int>> G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005],edge_node[10005];
129 //pa是父母節點，暫存用的，edge_node是每個編被存在哪個點裡面的
      陣列
130 void bfs(int root){
131 //在建構的時候把每個點都設成一個splay tree
132   queue<int > q;
133   for(int i=1;i<=n;++i)pa[i]=0;
134   q.push(root);
135   while(q.size()){
136     int u=q.front();
137     q.pop();
138     for(auto P:G[u]){
139       int v=P.first;
140       if(v!=pa[u]){
141         pa[v]=u;
142         nd[v].pa=u;
143         nd[v].data=e[P.second].w;
144         edge_node[P.second]=v;
145         up(v);
146         q.push(v);
147       }
148     }
149   }
150 }
151 void change(int x,int b){
152   splay(x);
153   //nd[x].data=b;
154   up(x);
155 }
```

3 DP

3.1 LCIS

```
1 vector<int> LCIS(vector<int> a, vector<int> b) {
2   int n = a.size(), m = b.size();
3   int dp[LEN][LEN] = {}, pre[LEN][LEN] = {};
4   for(int i=1; i<=n; i++) {
5     int p = 0;
6     for(int j=1; j<=m; j++)
7       if(a[i-1]!=b[j-1]) {
8         dp[i][j] = dp[i-1][j], pre[i][j] = j;
```

```
9        if( a[i-1]>b[j-1] && dp[i-1][j]>dp[i-1][p] )
10         p = j;
11       } else {
12         dp[i][j] = dp[i-1][p]+1, pre[i][j] = p;
13       }
14   }
15   int len = 0, p = 0;
16   for(int j=1; j<=m; j++)
17     if(dp[n][j]>len) len = dp[n][j], p = j;
18   vector<int> ans;
19   for(int i=n; i>=1; i--) {
20     if(a[i-1]==b[p-1]) ans.push_back(b[p-1]);
21     p = pre[i][p];
22   }
23   reverse(ans.begin(), ans.end());
24   return ans;
25 }
```

3.2 Bounded_Knapsack

```
1 namespace {
2   static const int MAXW = 1000005;
3   static const int MAXN = 1005;
4   struct BB {
5     int w, v, c;
6     BB(int w = 0, int v = 0, int c = 0): w(w), v(v), c(c)
         {}
7     bool operator<(const BB &x) const {
8       return w * c  < x.w * x.c;
9     }
10  };
11  static int run(BB A[], int dp[], int W, int N) {
12    static int MQ[MAXW][2];
13    for (int i = 0, sum = 0; i < N; i++) {
14      int w = A[i].w, v = A[i].v, c = A[i].c;
15      sum = min(sum + w*c, W);
16      for (int j = 0; j < w; j++) {
17        int l = 0, r = 0;
18        MQ[l][0] = 0, MQ[l][1] = dp[j];
19        for (int k = 1, tw = w+j, tv = v; tw <= sum
               && k <= c; k++, tw += w, tv += v) {
20          int dpv = dp[tw] - tv;
21          while (l <= r && MQ[r][1] <= dpv) r--;
22          r++;
23          MQ[r][0] = k, MQ[r][1] = dpv;
24          dp[tw] = max(dp[tw], MQ[l][1] + tv);
25        }
26        for (int k = c+1, tw = (c+1)*w+j, tv = (c+1)*
               v; tw <= sum; k++, tw += w, tv += v) {
27          if (k - MQ[l][0] > c) l++;
28          int dpv = dp[tw] - tv;
29          while (l <= r && MQ[r][1] <= dpv) r--;
30          r++;
31          MQ[r][0] = k, MQ[r][1] = dpv;
32          dp[tw] = max(dp[tw], MQ[l][1] + tv);
33        }
34      }
35    }
36  }
37  static int knapsack(int C[][3], int N, int W) { // O(WN)
38    vector<BB> A;
39    for (int i = 0; i < N; i++) {
40      int w = C[i][0], v = C[i][1], c = C[i][2];
```

```
41      A.push_back(BB(w, v, c));
42    }
43    assert(N < MAXN);
44    static int dp1[MAXW+1], dp2[MAXW+1];
45    BB Ar[2][MAXN];
46    int ArN[2] = {};
47    memset(dp1, 0, sizeof(dp1[0])*(W+1));
48    memset(dp2, 0, sizeof(dp2[0])*(W+1));
49    sort(A.begin(), A.end());
50    int sum[2] = {};
51    for (int i = 0; i < N; i++) {
52      int ch = sum[1] < sum[0];
53      Ar[ch][ArN[ch]] = A[i];
54      ArN[ch]++;
55      sum[ch] = min(sum[ch] + A[i].w*A[i].c, W);
56    }
57    run(Ar[0], dp1, W, ArN[0]);
58    run(Ar[1], dp2, W, ArN[1]);
59    int ret = 0;
60    for (int i = 0, j = W, mx = 0; i <= W; i++, j--) {
61      mx = max(mx, dp2[i]);
62      ret = max(ret, dp1[j] + mx);
63    }
64    return ret;
65  }
66 }
67 int main() {
68   int W, N;
69   assert(scanf("%d %d", &W, &N) == 2);
70   int C[MAXN][3];
71   for (int i = 0; i < N; i++)
72     assert(scanf("%d %d %d", &C[i][1], &C[i][0], &C[i
            ][2]) == 3);
73   printf("%d\n", knapsack(C, N, W));
74   return 0;
75 }
```

3.3 1D1D

```
1 int t, n, L, p;
2 char s[MAXN][35];
3 ll sum[MAXN] = {0};
4 long double dp[MAXN] = {0};
5 int prevd[MAXN] = {0};
6 long double pw(long double a, int n) {
7   if ( n == 1 ) return a;
8   long double b = pw(a, n/2);
9   if ( n & 1 ) return b*b*a;
10  else return b*b;
11 }
12 long double f(int i, int j) {
13   // cout << (sum[i] - sum[j]+i-j-1-L) << endl;
14   return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
15 }
16 struct INV {
17   int L, R, pos;
18 };
19 INV stk[MAXN*10];
20 int top = 1, bot = 1;
21 void update(int i) {
22   while ( top > bot && i < stk[top].L && f(stk[top].L, i) <
           f(stk[top].L, stk[top].pos) ) {
23     stk[top - 1].R = stk[top].R;
```

```
24          top--;
25      }
26      int lo = stk[top].L, hi = stk[top].R, mid, pos = stk[top
           ].pos;
27      // if ( i >= lo ) lo = i + 1;
28      while ( lo != hi ) {
29          mid = lo + (hi - lo) / 2;
30          if ( f(mid, i) < f(mid, pos) ) hi = mid;
31          else lo = mid + 1;
32      }
33      if ( hi < stk[top].R ) {
34          stk[top + 1] = (INV) { hi, stk[top].R, i };
35          stk[top++].R = hi;
36      }
37 }
38 int main() {
39      cin >> t;
40      while ( t-- ) {
41          cin >> n >> L >> p;
42          dp[0] = sum[0] = 0;
43          for ( int i = 1 ; i <= n ; i++ ) {
44              cin >> s[i];
45              sum[i] = sum[i-1] + strlen(s[i]);
46              dp[i] = numeric_limits<long double>::max();
47          }
48          stk[top] = (INV) {1, n + 1, 0};
49          for ( int i = 1 ; i <= n ; i++ ) {
50              if ( i >= stk[bot].R ) bot++;
51              dp[i] = f(i, stk[bot].pos);
52              update(i);
53              // cout << (ll) f(i, stk[bot].pos) << endl;
54          }
55          if ( dp[n] > 1e18 ) {
56              cout << "Too hard to arrange" << endl;
57          } else {
58              vector<PI> as;
59              cout << (ll)dp[n] << endl;
60          }
61      } return 0;
62 }
```

3.4 SOS

```
1 for(int i=0;i<N;i++){
2      dp[i] = arr[i];
3 }
4 for(int i=0;i<22;i++){
5      for(int j=0;j<=(1<<22);j++){
6          if(j&(1<<(i)))
7              dp[j] = max(dp[j],dp[j^(1<<i)]);
8      }
9 }
```

4 Graph

4.1 Dijkstra

```
1 /** 問某點到所有圖上的點的最短距離。0/1-based 都安全。 edge
      要
2  * 是 {cost，dest} 格式。回傳的陣列若含有 -1 表示 src 到該位
      置
3  * 不連通 **/
4 typedef pair<ll, int> pii;
5 vector<ll> dijkstra(int src, vector<vector<pii>>& edge) {
6      vector<ll> sum(edge.size(), -1);
7      priority_queue<pii, vector<pii>, greater<pii>> q;
8      q.emplace(0, src);
9      while (q.size()) {
10         int v = q.top().second; ll d = q.top().first;
11         q.pop();
12         if (sum[v] != -1) continue;
13         sum[v] = d;
14         for (auto& e : edge[v])
15             if (sum[e.second] == -1)
16                 q.emplace(d + e.first, e.second);
17     } return sum;
18 }
```

4.2 Bellman Ford

```
1 vector<pii> G[maxn];
2 int dis[maxn];
3 bool BellmanFord(int n,int s) {
4      for(int i=1; i<=n; i++) dis[i] = INF;
5      dis[s] = 0;
6      bool relax;
7      for(int r=1; r<=n; r++) { //O(VE)
8          relax = false;
9          for(int i=1; i<=n; i++)
10             for(pii e:G[i])
11                 if( dis[i] + e.second < dis[e.first] )
12                     dis[e.first] = dis[i] + e.second, relax =
                            true;
13     }
14     return relax; //有負環
15 }
```

4.3 SPFA

```
1 vector<pii> G[maxn]; int dis[maxn];
2 void SPFA(int n,int s) { //O(kE) k~2.
3      for(int i=1; i<=n; i++) dis[i] = INF;
4      dis[s] = 0;
5      queue<int> q; q.push(s);
6      bool inque[maxn] = {};
7      while(!q.empty()) {
8          int u = q.front(); q.pop();
9          inque[u] = false;
10         for(pii e:G[u]) {
11             int v = e.first , w = e.second;
12             if( dis[u] + w < dis[v]) {
13                 if(!inque[v]) q.push(v), inque[v] = true;
14                 dis[v] = dis[u] + w;
15             }
16         }
17     }
18 }
```

4.4 Prim

```
1 /** 0/1-based 安全， n 是節點數量（必須剛好）。 edge 格式為
2  * {cost, dest} ，回傳 -1 表示圖不連通。**/
3 typedef pair<ll, int> pii;
4 ll minpath(vector<vector<pii>>& edge, int n) {
5      vector<bool> vis(n + 1);
6      priority_queue<pii, vector<pii>, greater<pii>> q;
7      q.emplace(0, 1);
8      ll ret = 0; int nvis = 0;
9      while (nvis < n && q.size()) {
10         ll d = q.top().first;
11         int v = q.top().second; q.pop();
12         if (vis[v]) continue;
13         vis[v] = 1; ret += d;
14         if (++nvis == n) return ret;
15         for (auto& e : edge[v])
16             if (!vis[e.second]) q.push(e);
17     } return -1;
18 }
```

4.5 Mahattan MST

```
1 #define REP(i,n) for(int i=0;i<n;i++)
2 typedef long long LL;
3 const int N=200100;
4 int n,m;
5 struct PT {int x,y,z,w,id;} p[N];
6 inline int dis(const PT &a,const PT &b){return abs(a.xb.x)+
      abs(a.y-b.y);}
7 inline bool cpx(const PT &a,const PT &b)
8 {return a.x!=b.x? a.x>b.x:a.y>b.y;}
9 inline bool cpz(const PT &a,const PT &b){return a.z<b.z;}
10 struct E{int a,b,c;}e[8*N];
11 bool operator<(const E&a,const E&b){return a.c<b.c;}
12 struct Node{ int L,R,key; } node[4*N];
13 int s[N];
14 int F(int x) {return s[x]==x ? x : s[x]=F(s[x]); }
15 void U(int a,int b) {s[F(b)]=F(a);}
16 void init(int id,int L,int R) {
17     node[id] = (Node){L,R,-1};
18     if(L==R)return;
19     init(id*2,L,(L+R)/2);
20     init(id*2+1,(L+R)/2+1,R);
21 }
22 void ins(int id,int x) {
23     if(node[id].key==-1 || p[node[id].key].w>p[x].w)
24         node[id].key=x;
25     if(node[id].L==node[id].R) return;
26     if(p[x].z<=(node[id].L+node[id].R)/2) ins(id*2,x);
27     else ins(id*2+1,x);
28 }
29 int Q(int id,int L,int R){
30     if(R<node[id].L || L>node[id].R)return -1;
31     if(L<=node[id].L && node[id].R<=R)return node[id].key;
32     int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
33     if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
34     else return b;
35 }
36 void calc() {
37     REP(i,n) {
38         p[i].z = p[i].y-p[i].x;
```

```
39          p[i].w = p[i].x+p[i].y;
40      }
41      sort(p,p+n,cpz);
42      int cnt = 0, j, k;
43      for(int i=0; i<n; i=j){
44          for(j=i+1; p[j].z==p[i].z && j<n; j++);
45          for(k=i, cnt++; k<j; k++) p[k].z = cnt;
46      }
47      init(1,1,cnt);
48      sort(p,p+n,cpx);
49      REP(i,n) {
50          j=Q(1,p[i].z,cnt);
51          if(j!=-1) e[m++] = (E){p[i].id, p[j].id, dis(p[i],p[j
                ])};
52          ins(1,i);
53      }
54  }
55  LL MST() {
56      LL r=0;
57      sort(e, e+m);
58      REP(i, m) {
59          if(F(e[i].a)==F(e[i].b)) continue;
60          U(e[i].a, e[i].b);
61          r += e[i].c;
62      }
63      return r;
64  }
65  int main() {
66      int ts;
67      scanf("%d", &ts);
68      while (ts--) {
69          m = 0;
70          scanf("%d",&n);
71          REP(i,n) {scanf("%d%d",&p[i].x,&p[i].y);p[i].id=s[i]=
                i;}
72          calc();
73          REP(i,n)p[i].y= -p[i].y;
74          calc();
75          REP(i,n)swap(p[i].x,p[i].y);
76          calc();
77          REP(i,n)p[i].x=-p[i].x;
78          calc();
79          printf("%lld\n",MST()*2);
80      }
81      return 0;
82  }
```

4.6 LCA

```
1  /** 所有 LCA 都是 0/1-based 安全的。建構式 edge 表示 adj
2   * 邊資訊。 只支援無向樹。這三個類別各有優缺點。**/
3
4  /** 最快的 LCA O(N+Q) ，但非常吃記憶體 O(N^2)。支援非離線。*
       */
5  class SsadpTarjan {
6      private:
7      int n;
8      vector<int> par, dep; vector<vector<int>> ca;
9      int dfs(int u, vector<vector<int>>& edge, int d) {
10         dep[u] = d;
11         for (int a = 0; a < n; a++)
12             if (dep[a] != -1)
```

```
13                 ca[a][u] = ca[u][a] = parent(a);
14         for (int a : edge[u]) {
15             if (dep[a] != -1) continue;
16             dfs(a, edge, d + 1);
17             par[a] = u;
18         }
19     }
20     int parent(int x) {
21         if (par[x] == x) return x;
22         return par[x] = parent(par[x]);
23     }
24
25     public:
26     SsadpTarjan(vector<vector<int>>& edge, int root)
27         : n(edge.size()) {
28         dep.assign(n, -1); par.resize(n);
29         ca.assign(n, vector<int>(n));
30
31         for (int i = 0; i < n; i++) par[i] = i;
32         dfs(root, edge, 0);
33     }
34     int lca(int a, int b) { return ca[a][b]; }
35     int dist(int a, int b) {
36         return dep[a] + dep[b] - 2 * dep[ca[a][b]];
37     }
38 };
39
40 /** 最快的 LCA O(N+Q) 且最省記憶體 O(N+Q) 。但必須離線。**/
41 #define x first  // 加速
42 #define y second
43 class OfflineTarjan {
44     private:
45     vector<int> par, anc, dep, ans, rank;
46     vector<vector<pii>> qry;
47     // 出於安全考量你可以把 & 去掉
48     vector<vector<int>>& edge;
49     int root, n;
50
51     void merge(int a, int b) {
52         a = parent(a), b = parent(b);
53         if (rank[a] < rank[b]) swap(a, b);
54         par[b] = a;
55         if (rank[a] == rank[b]) rank[a]++;
56     }
57     void dfs(int u, int d) {
58         anc[parent(u)] = u, dep[u] = d;
59         for (int a : edge[u]) {
60             if (dep[a] != -1) continue;
61             dfs(a, d + 1);
62             merge(a, u);
63             anc[parent(u)] = u;
64         }
65         for (auto q : qry[u]) {
66             if (dep[q.first] != -1)
67                 ans[q.second] = anc[parent(q.first)];
68         }
69     }
70     int parent(int x) {
71         if (par[x] == x) return x;
72         return par[x] = parent(par[x]);
73     }
74     void solve(vector<pii>& query) {
75         dep.assign(n, -1), rank.assign(n, 0);
76         par.resize(n), anc.resize(n);
77         for (int i = 0; i < n; i++) anc[i] = par[i] = i;
```

```
78         ans.resize(query.size());
79         qry.resize(n);
80         for (int i = 0; i < query.size(); i++) {
81             auto& q = query[i];
82             qry[q.first].emplace_back(q.second, i);
83             qry[q.second].emplace_back(q.first, i);
84         }
85         dfs(root, 0);
86     }
87
88     public:
89     // edge 是傳 reference ，完成所有查詢前萬萬不可以改。
90     OfflineTarjan(vector<vector<int>>& edge, int root)
91         : edge(edge), root(root), n(edge.size()) {}
92     // 離線查詢， query 陣列包含所有詢問 {src, dst} 。呼叫一
                次無
93     // 論 query 量多少，複雜度都是 O(N) 。所以應盡量只呼叫一
                次。
94     vector<int> lca(vector<pii>& query) {
95         solve(query);
96         return ans;
97     }
98     vector<int> dist(vector<pii>& query) {
99         solve(query);
100         for (int i = 0; i < query.size(); i++) {
101             auto& q = query[i];
102             ans[i] = dep[q.first] + dep[q.second] -
103                 2 * dep[ans[i]];
104         }
105         return ans;
106     }
107 };
108
109 /** 威達的 LCA ，時間普通 O(Q*log(N)) ，記憶體需求也普通
110  *  O(N*log(N)) 。支援非離線。**/
111 class SparseTableTarjan {
112     private:
113     int maxlg;
114     vector<vector<int>> anc;
115     vector<int> dep;
116
117     void dfs(int u, vector<vector<int>>& edge, int d) {
118         dep[u] = d;
119         for (int i = 1; i < maxlg; i++)
120             if (anc[u][i - 1] == -1) break;
121             else anc[u][i] = anc[anc[u][i - 1]][i - 1];
122         for (int a : edge[u]) {
123             if (dep[a] != -1) continue;
124             anc[a][0] = u;
125             dfs(a, edge, d + 1);
126         }
127     }
128
129     public:
130     SparseTableTarjan(vector<vector<int>>& edge, int root) {
131         int n = edge.size();
132         maxlg = ceil(log2(n));
133         anc.assign(n, vector<int>(maxlg, -1));
134         dep.assign(n, -1);
135         dfs(root, edge, 0);
136     }
137     int lca(int a, int b) {
138         if (dep[a] > dep[b]) swap(a, b);
139         for (int k = 0; dep[b] - dep[a]; k++)
```

```
140          if (((dep[b] - dep[a]) >> k) & 1) b = anc[b][k];
141      if (a == b) return a;
142      for (int k = maxlg - 1; k >= 0; k--)
143          if (anc[a][k] != anc[b][k])
144              a = anc[a][k], b = anc[b][k];
145      return anc[a][0];
146  }
147  int dist(int a, int b) {
148      return dep[a] + dep[b] - 2 * dep[lca(a, b)];
149  }
150 };
```

4.7 Tarjan

```
1  割點
2  點 u 為割點 if and only if 滿足 1. or 2.
3  1. u 爲樹根，且 u 有多於一個子樹。
4  2. u 不爲樹根，且滿足存在 (u,v) 爲樹枝邊 (或稱父子邊，即 u 爲
        v 在搜索樹中的父親)，使得 DFN(u) <= Low(v)。
5  -----------------------------------------------------
6  橋
7  一條無向邊 (u,v) 是橋 if and only if (u,v) 爲樹枝邊，且滿足
        DFN(u) < Low(v)。
8  // 0 base
9  struct TarjanSCC{
10     static const int MAXN = 1000006;
11     int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
12     vector<int> G[MAXN];
13     stack<int> stk;
14     bool ins[MAXN];
15     void tarjan(int u) {
16         dfn[u] = low[u] = ++count;
17         stk.push(u);
18         ins[u] = true;
19         for(auto v:G[u]) {
20             if(!dfn[v]) {
21                 tarjan(v);
22                 low[u] = min(low[u], low[v]);
23             } else if(ins[v]) {
24                 low[u] = min(low[u], dfn[v]);
25             }
26         }
27         if(dfn[u] == low[u]) {
28             int v;
29             do {
30                 v = stk.top(); stk.pop();
31                 scc[v] = scn;
32                 ins[v] = false;
33             } while(v != u);
34             scn++;
35         }
36     }
37     void getSCC(){
38         memset(dfn,0,sizeof(dfn));
39         memset(low,0,sizeof(low));
40         memset(ins,0,sizeof(ins));
41         memset(scc,0,sizeof(scc));
42         count = scn = 0;
43         for(int i = 0 ; i < n ; i++ )
44             if(!dfn[i]) tarjan(i);
45     }
```

4.8 BCC_edge

```
1  邊雙連通
2  任意兩點間至少有兩條不重疊的路徑連接，找法：
3  1. 標記出所有的橋
4  2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通
5  // from BCW
6  struct BccEdge {
7      static const int MXN = 100005;
8      struct Edge { int v,eid; };
9      int n,m,step,par[MXN],dfn[MXN],low[MXN];
10     vector<Edge> E[MXN];
11     DisjointSet djs;
12     void init(int _n) {
13         n = _n; m = 0;
14         for (int i=0; i<n; i++) E[i].clear();
15         djs.init(n);
16     }
17     void add_edge(int u, int v) {
18         E[u].PB({v, m});
19         E[v].PB({u, m});
20         m++;
21     }
22     void DFS(int u, int f, int f_eid) {
23         par[u] = f;
24         dfn[u] = low[u] = step++;
25         for (auto it:E[u]) {
26             if (it.eid == f_eid) continue;
27             int v = it.v;
28             if (dfn[v] == -1) {
29                 DFS(v, u, it.eid);
30                 low[u] = min(low[u], low[v]);
31             } else {
32                 low[u] = min(low[u], dfn[v]);
33             }
34         }
35     }
36     void solve() {
37         step = 0;
38         memset(dfn, -1, sizeof(int)*n);
39         for (int i=0; i<n; i++) {
40             if (dfn[i] == -1) DFS(i, i, -1);
41         }
42         djs.init(n);
43         for (int i=0; i<n; i++) {
44             if (low[i] < dfn[i]) djs.uni(i, par[i]);
45         }
46     }
47 } graph;
```

4.9 最小平均環

```
1  #include<cfloat> //for DBL_MAX
2  int dp[MAXN][MAXN]; // 1-base,O(NM)
3  vector<tuple<int,int,int>> edge;
4  double mmc(int n){ //allow negative weight
5      const int INF = 0x3f3f3f3f;
```

```
6      for(int t=0; t<n; ++t){
7          memset(dp[t+1],0x3f,sizeof(dp[t+1]));
8          for(const auto &e:edge) {
9              int u, v, w; tie(u,v,w) = e;
10             dp[t+1][v] = min(dp[t+1][v],dp[t][u]+w);
11         }
12     }
13     double res = DBL_MAX;
14     for(int u=1; u<=n; ++u) {
15         if(dp[n][u]==INF) continue;
16         double val = -DBL_MAX;
17         for(int t=0;t<n;++t)
18             val = max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
19         res = min(res,val);
20     } return res;
21 }
```

4.10 2-SAT

```
1  const int MAXN = 2020;
2  struct TwoSAT{
3      static const int MAXv = 2*MAXN;
4      vector<int> GO[MAXv],BK[MAXv],stk;
5      bool vis[MAXv];
6      int SC[MAXv];
7      void imply(int u,int v){ // u imply v
8          GO[u].push_back(v);
9          BK[v].push_back(u);
10     }
11     int dfs(int u,vector<int>*G,int sc){
12         vis[u]=1, SC[u]=sc;
13         for (int v:G[u])if (!vis[v])
14             dfs(v,G,sc);
15         if (G==GO) stk.push_back(u);
16     }
17     int scc(int n=MAXv){
18         memset(vis,0,sizeof(vis));
19         for (int i=0; i<n; i++)
20             if (!vis[i]) dfs(i,GO,-1);
21         memset(vis,0,sizeof(vis));
22         int sc=0;
23         while (!stk.empty()){
24             if (!vis[stk.back()])
25                 dfs(stk.back(),BK,sc++);
26             stk.pop_back();
27         }
28     }
29 } SAT;
30 int main(){
31     SAT.scc(2*n);
32     bool ok  = 1;
33     for (int i=0; i<n; i++){
34         if (SAT.SC[2*i]==SAT.SC[2*i+1]) ok  = 0;
35     }
36     if (ok) {
37         for (int i=0; i<n; i++)
38             if (SAT.SC[2*i]>SAT.SC[2*i+1])
39                 cout << i << endl;
40     }
41     else puts("NO");
42 }
43 void warshall(){
44     bitset<2003> d[2003];
```

```
45      for (int k=0; k<n; k++)
46          for (int i=0; i<n; i++)
47              if (d[i][k]) d[i] |= d[k];
48 }
```

4.11 生成樹數量

```
1  // D : degree-matrix
2  // A : adjacent-matrix
3  // 無向圖
4  //   (u,v)
5  //   A[u][v]++, A[v][u]++
6  //   D[u][u]++, D[v][v]++
7  //   G = D-A
8  //   abs(det(G去掉i-col和i-row))
9  //   生成樹的數量
10 // 有向圖
11 //   A[u][v]++
12 //   D[v][v]++ (in-deg)
13 //   以i為root的樹形圖數量
14 //   所有節點都能到達root
```

5 Flow_Matching

5.1 Dinic

```
1  // 一般來說複雜度遠低於 O(EV^2) ， 二分圖約 O(E * sqrt(v)) 。
2  // 0/1-based 都安全。
3  class Dinic {
4      struct edge {
5          int d, r; ll c;
6          edge(int d, ll c, int r) : d(d), c(c), r(r){};
7      };
8  private:
9      vector<vector<edge>> adj; vector<int> lv, ve; int n;
10     bool mklv(int s, int d) {
11         lv.assign(n, -1); lv[s] = 0;
12         queue<int> q; q.push(s);
13         while (!q.empty()) {
14             int v = q.front(); q.pop();
15             for (auto& e : adj[v]) {
16                 if (e.c == 0 || lv[e.d] != -1) continue;
17                 lv[e.d] = lv[v] + 1, q.push(e.d);
18             }
19         }
20         return lv[d] > 0;
21     }
22     ll aug(int v, ll f, int d) {
23         if (v == d) return f;
24         for (; ve[v] < adj[v].size(); ve[v]++) {
25             auto& e = adj[v][ve[v]];
26             if (lv[e.d] != lv[v] + 1 || !e.c) continue;
27             ll sent = aug(e.d, min(f, e.c), d);
28             if (sent > 0) {
29                 e.c -= sent, adj[e.d][e.r].c += sent;
30                 return sent;
31             }
```

```
32         }
33         return 0;
34     }
35 public:
36     // 建立空圖， n 是節點 (包含 source, sink) 數量
37     Dinic(int n) : n(n + 1) { clear(); }
38     // 清空整個圖，這需要重複使用 dinic 時 (如二分搜) 很方便
39     void clear() { adj.assign(n, vector<edge>()); }
40     // 加有向邊 src->dst ，cap 是容量
41     void add_edge(int src, int dst, ll cap) {
42         edge ss(dst, cap, adj[dst].size());
43         edge dd(src, 0, adj[src].size());
44         adj[src].push_back(ss), adj[dst].push_back(dd);
45     }
46     ll max_flow(int s, int d) {
47         ll ret = 0;
48         while (mklv(s, d)) {
49             ve.assign(n, 0);
50             while (ll f = aug(s, 9e18, d)) ret += f;
51         }
52         return ret;
53     }
54 };
```

5.2 Min Cost Max Flow

```
1  /** Min cost max flow。0/1-based 都安全。**/
2  class MCMF {
3      private:
4          struct edge { int to, r; ll rest, c; };
5          int n; ll f = 0, c = 0;
6          vector<vector<edge>> g;
7          vector<int> pre, prel;
8          bool run(int s, int t) {
9              vector<ll> dis(n, inf); vector<bool> vis(n);
10             dis[s] = 0; queue<int> q; q.push(s);
11             while (q.size()) {
12                 int u = q.front(); q.pop(); vis[u] = 0;
13                 for (int i = 0; i < g[u].size(); i++) {
14                     int v = g[u][i].to; ll w = g[u][i].c;
15                     if (g[u][i].rest <= 0 ||
16                         dis[v] <= dis[u] + w)
17                         continue;
18                     pre[v] = u, prel[v] = i;
19                     dis[v] = dis[u] + w;
20                     if (!vis[v]) vis[v] = 1, q.push(v);
21                 }
22             }
23             if (dis[t] == inf) return 0;
24             ll tf = inf;
25             for (int v = t, u, l; v != s; v = u) {
26                 u = pre[v], l = prel[v];
27                 tf = min(tf, g[u][l].rest);
28             }
29             for (int v = t, u, l; v != s; v = u) {
30                 u = pre[v], l = prel[v], g[u][l].rest -= tf;
31                 g[v][g[u][l].r].rest += tf;
32             }
33             c += tf * dis[t], f += tf;
34             return 1;
35         }
36     public:
```

5.3 Ford Fulkerson

```
37         // 建立空圖，n 是節點數量 (包含 source 和 sink)
38         MCMF(int n)
39             : n(n + 1), g(n + 1), pre(n + 1), prel(n + 1) {}
40         // 加有向邊 u->v ，cap 容量 cost 成本
41         void add_edge(int u, int v, ll cap, ll cost) {
42             g[u].push_back({v, (int)g[v].size(), cap, cost});
43             g[v].push_back({u, (int)g[u].size() - 1, 0, -cost});
44         }
45         pair<ll, ll> query(int src, int sink) {
46             while (run(src, sink));
47             return {f, c};  //{min cost, max flow}
48         }
49 };
```

```
1  const int maxn = 1e5 + 10, INF = 1e9;
2  const long long INF64 = 1e18;
3  struct edge{ int to, cap, rev; };
4  vector<edge> G[maxn];
5  int n, m, s, t, a, b, c;
6  bool vis[maxn];
7  int dfs(int v, int t, int f) {
8      cout << v << ' ' << t << ' ' << f << '\n';
9      if (v == t) return f;
10     vis[v] = true;
11     for (edge &e: G[v]) {
12         if (!vis[e.to] && e.cap > 0) {
13             int d = dfs(e.to, t, min(f, e.cap));
14             if (d > 0) {
15                 e.cap -= d, G[e.to][e.rev].cap += d;
16                 return d;
17             }
18         }
19     }
20     return 0;
21 }
22 int ford_fulkerson(int s, int t) {
23     int flow = 0, f;
24     for (int i = 0; i < n; i++) {
25         cout << i << " : ";
26         for (edge e: G[i])
27             cout << '(' << e.to << ',' << e.cap << ')' << ' '
                   ;
28         cout << '\n';
29     }
30     do {
31         memset(vis, false, sizeof(vis));
32         f = dfs(s, t, INF);
33         for (int i = 0; i < n; i++) {
34             cout << i << " : ";
35             for (edge e: G[i])
36                 cout << '(' << e.to << ',' << e.cap << ')' <<
                       ' ';
37             cout << '\n';
38         }
39         cout << f << '\n';
40         flow += f;
41     } while (f > 0);
42     return flow;
43 }
44 void init(int n) {
45     for (int i = 0; i < n; i++) G[i].clear();
```

```
46 }
47 int main() {
48     cin >> n >> m >> s >> t;
49     init(n);
50     while (m--) {
51         cin >> a >> b >> c;
52         G[a].push_back((edge){b, c, (int)G[b].size()});
53         G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
54     }
55     cout << ford_fulkerson(s, t) << '\n';
56     return 0;
57 }
```

5.4 KM

```
1  /** 二分圖最大權值匹配 KM 演算法，複雜度 O(n^3)*/
2  #define inf 5e18
3  class KM {
4    private:
5      const vector<vector<ll>>& e;
6      int xx, yy;
7      vector<ll> cx, cy, wx, wy;
8      vector<bool> vx, vy;
9      ll z;
10
11     bool dfs(int u) {
12         vx[u] = 1;
13         for (int v = 0; v < yy; v++) {
14             if (vy[v] || e[u][v] == inf) continue;
15             ll t = wx[u] + wy[v] - e[u][v];
16             if (t == 0) {
17                 vy[v] = 1;
18                 if (cy[v] == -1 || dfs(cy[v])) {
19                     cx[u] = v, cy[v] = u;
20                     return 1;
21                 }
22             } else if (t > 0)
23                 z = min(z, t);
24         }
25         return 0;
26     }
27   public:
28     // 問最大匹配權重。
29     ll max_weight() {
30         for (int i = 0; i < xx; i++)
31             for (int j = 0; j < yy; j++) {
32                 if (e[i][j] == inf) continue;
33                 wx[i] = max(wx[i], e[i][j]);
34             }
35         for (int i = 0; i < xx; i++) {
36             while (1) {
37                 z = inf, vx.assign(xx, 0), vy.assign(yy, 0);
38                 if (dfs(i)) break;
39                 for (int j = 0; j < xx; j++)
40                     if (vx[j]) wx[j] -= z;
41                 for (int j = 0; j < yy; j++)
42                     if (vy[j]) wy[j] += z;
43             }
44         }
45         ll ans = 0;
46         for (int i = 0; i < xx; i++)
47             if (cx[i] != -1) ans += e[i][cx[i]];
```

```
48         return ans;
49     }
50     // 給他 n * m 的權重表 (n <= m)，求最大完全匹配權重，權重
           可以
51     // 是負數。注意 n > m 會導致無窮迴圈。
52     KM(vector<vector<ll>>& e) : e(e) {
53         xx = e.size(), yy = e[0].size();  // xx 要 <= yy !!
54         cx.assign(xx, -1), cy.assign(yy, -1);
55         wx.assign(xx, 0), wy.assign(yy, 0);
56     }
57 };
```

5.5 Hopcroft Karp

```
1  int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
2  vector<int> edge[maxn];  // for Left
3  bool dfs(int u) {
4      vis[u] = true;
5      for (vector<int>::iterator it = edge[u].begin();
6           it != edge[u].end(); ++it) {
7          int v = pr2[*it];
8          if (v == -1 ||
9              (!vis[v] && level[u] < level[v] && dfs(v))) {
10             pr[u] = *it, pr2[*it] = u;
11             return true;
12         }
13     }
14     return false;
15 }
16 int hopcroftKarp() {
17     memset(pr, -1, sizeof(pr));
18     memset(pr2, -1, sizeof(pr2));
19     for (int match = 0;;) {
20         queue<int> Q;
21         for (int i = 1; i <= n; ++i) {
22             if (pr[i] == -1) {
23                 level[i] = 0;
24                 Q.push(i);
25             } else
26                 level[i] = -1;
27         }
28         while (!Q.empty()) {
29             int u = Q.front();
30             Q.pop();
31             for (vector<int>::iterator it = edge[u].begin();
32                  it != edge[u].end(); ++it) {
33                 int v = pr2[*it];
34                 if (v != -1 && level[v] < 0) {
35                     level[v] = level[u] + 1;
36                     Q.push(v);
37                 }
38             }
39         }
40         for (int i = 1; i <= n; ++i) vis[i] = false;
41         int d = 0;
42         for (int i = 1; i <= n; ++i)
43             if (pr[i] == -1 && dfs(i)) ++d;
44         if (d == 0) return match;
45         match += d;
46     }
47 }
```

5.6 SW-MinCut

```
1  // all pair min cut
2  // global min cut
3  struct SW { // O(V^3)
4      static const int MXN = 514;
5      int n, vst[MXN], del[MXN];
6      int edge[MXN][MXN], wei[MXN];
7      void init(int _n){
8          n = _n; FZ(edge); FZ(del);
9      }
10     void addEdge(int u, int v, int w) {
11         edge[u][v] += w; edge[v][u] += w;
12     }
13     void search(int &s, int &t) {
14         FZ(vst); FZ(wei);
15         s = t = -1;
16         while (true){
17             int mx=-1, cur=0;
18             for (int i=0; i<n; i++)
19                 if (!del[i] && !vst[i] && mx<wei[i])
20                     cur = i, mx = wei[i];
21             if (mx == -1) break;
22             vst[cur] = 1;
23             s = t; t = cur;
24             for (int i=0; i<n; i++)
25                 if (!vst[i] && !del[i]) wei[i] += edge[cur][i
                       ];
26         }
27     }
28     int solve() {
29         int res = 2147483647;
30         for (int i=0, x, y; i<n-1; i++) {
31             search(x,y);
32             res = min(res,wei[y]);
33             del[y] = 1;
34             for (int j=0; j<n; j++)
35                 edge[x][j] = (edge[j][x] += edge[y][j]);
36         }
37         return res;
38     }
39 } graph;
```

5.7 Stable Marriage

```
1  // 演算法筆記
2  1. N位男士各自向自己最喜愛的女士求婚。
3  2. N位女士各自從自己的求婚者中，挑最喜愛的那位男士訂婚，但是
       往後可背約。
4     沒有求婚者的女士，就只好等等。
5  3. 失敗的男士們，只好各自向自己次喜愛的女士求婚。
6  4. N位女士各自從自己的求婚者中，挑最喜歡的那位男士訂婚，但是
       往後可背約。
7     已訂婚卻有更喜愛的男士求婚的女士，就毀約，改為與此男士訂
       婚。
8     沒有求婚者的女士，就只好再等等。
9  5. 重複3. 4.直到形成N對伴侶為止。
10 // Jinkela
11 queue<int> Q;
12 for ( i : 所有考生 ) {
```

```
13        設定在第0志願;
14        Q.push(考生i);
15 }
16 while(Q.size()){
17        當前考生=Q.front();Q.pop();
18        while ( 此考生未分發 ) {
19            指標移到下一志願;
20            if ( 已經沒有志願 or 超出志願總數 ) break;
21            計算該考生在該科系加權後的總分;
22            if ( 不符合科系需求 ) continue;
23            if ( 目前科系有餘額 ) {
24                依加權後分數高低順序將考生id加入科系錄取名單中;
25                break;
26            }
27            if ( 目前科系已額滿 ) {
28                if ( 此考生成績比最低分數還高 ) {
29                    依加權後分數高低順序將考生id加入科系錄取名單;
30                    Q.push(被踢出的考生);
31                }
32            }
33        }
34 }
```

6 Math

6.1 快速冪

```
1 const int P = 1e9 + 7;
2 #define ll long long
3 ll fpow(int a, int b) {
4     ll ret = 1;
5     while (b) {
6         if (b & 1)
7             ret = ret * a % P;
8         a = a * a % P;
9     }
10    return ret;
11 }
```

6.2 模逆元

```
1 // 解 (ax == 1) mod p 。p 必須是質數，a 是正整數。
2 ll modinv(ll a, ll p) {
3     if (p == 1) return 0;
4     ll pp = p, y = 0, x = 1;
5     while (a > 1) {
6         ll q = a / p, t = p;
7         p = a % p, a = t, t = y, y = x - q * y, x = t;
8     }
9     if (x < 0) x += pp;
10    return x;
11 }
12 // 解 (ax == b) mod p 。p 必須是質數，a 和 b 是正整數。
13 ll modinv(ll a, ll b, ll p) {
14     ll ret = modinv(a, p);
15     return ret * b % p;
```

```
16 }
```

6.3 離散根號

```
1 int order(ll b, ll p) {
2     if (__gcd(b, p) != 1) return -1;
3     int ret = 2;
4     while (++ret)
5         if (fastpow(b, ret, p) == 1) break;
6     return ret;
7 }
8 // 把 fastpow 也抄過來，會用到。
9 // 問 (x^2 = y) mod p 的解。回傳 -1 表示 x 無解。
10 ll dsqrt(ll y, ll p) {
11     if (__gcd(y, p) != 1) return -1;
12     if (fastpow(y, (p - 1 / 2), p) == p - 1) return -1;
13     int e = 0;
14     ll s = p - 1;
15     while (!(s & 1)) s >>= 1, e++;
16     int q = 2;
17     while (1)
18         if (fastpow(q, (p - 1) / 2, p) == p - 1)
19             break;
20         else q++;
21     ll x = fastpow(y, (s + 1) / 2, p);
22     ll b = fastpow(y, s, p);
23     ll g = fastpow(q, s, p);
24     while (1) {
25         int m;
26         for (m = 0; m < e; m++) {
27             int o = order(p, b);
28             if (o == -1) return -1;
29             if (o == fastpow(2, m, p)) break;
30         }
31         if (m == 0) return x;
32         x = x * fastpow(g, fastpow(2, e - m - 1), p) % p;
33         g = fastpow(g, fastpow(2, e - m, p), p);
34         b = b * g % p;
35         if (b == 1) return x;
36         e = m;
37     }
38 }
```

6.4 外星模運算

```
1 //a[0]^(a[1]^a[2]^...)
2 #define maxn 1000000
3 int euler[maxn+5];
4 bool is_prime[maxn+5];
5 void init_euler(){
6     is_prime[1] = 1; //一不是質數
7     for(int i=1; i<=maxn; i++) euler[i]=i;
8     for(int i=2; i<=maxn; i++) {
9         if(!is_prime[i]) { //是質數
10            euler[i]--;
11            for(int j=i<<1; j<=maxn; j+=i) {
12                is_prime[j]=1;
13                euler[j] = euler[j]/i*(i-1);
14            }
```

```
15        }
16    }
17 }
18 LL pow(LL a, LL b, LL mod) { //a^b%mod
19     LL ans=1;
20     for(; b; a=a*a%mod, b>>=1)
21         if(b&1) ans = ans*a%mod;
22     return ans;
23 }
24 bool isless(LL *a, int n, int k) {
25     if(*a==1)return k>1;
26     if(--n==0)return *a<k;
27     int next=0;
28     for(LL b=1;b<k;++next)
29         b *= *a;
30     return isless(a+1, n, next);
31 }
32 LL high_pow(LL *a, int n, LL mod){
33     if(*a==1||--n==0)return *a%mod;
34     int k = 0, r = euler[mod];
35     for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
36         tma = tma*(*a)%mod;
37     if(isless(a+1,n,k))return pow(*a,high_pow(a+1,n,k),mod);
38     int tmd = high_pow(a+1,n,r), t = (tmd-k+r)%r;
39     return pow(*a,k+t,mod);
40 }
41 LL a[1000005]; int t,mod;
42 int main(){
43     init_euler();
44     scanf("%d", &t);
45     #define n 4
46     while(t--){
47         for(int i=0;i<n;++i)scanf("%lld", &a[i]);
48         scanf("%d", &mod);
49         printf("%lld\n", high_pow(a,n,mod));
50     }
51     return 0;
52 }
```

6.5 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)
                  for B in S])
31     else state[A] = mex(set(SG(B) for B in next_states(A)))
32   } return state[A]
33 }
```

6.6 Matrix

```
1  struct Matrix {
2      int r, c;
3      vector<vector<ll>> m;
4      Matrix(int r, int c): r(r), c(c), m(r, vector<ll>(c)) {}
5      vector<ll> &operator[](int i) { return m[i]; }
6      Matrix operator +(const Matrix &a) {
7          Matrix rev(r, c);
8          for (int i = 0; i < r; ++i)
9              for (int j = 0; j < c; ++j)
10                 rev[i][j] = m[i][j] + a.m[i][j];
11         return rev;
12     }
13     Matrix operator -(const Matrix &a) {
14         Matrix rev(r, c);
15         for (int i = 0; i < r; ++i)
16             for (int j = 0; j < c; ++j)
17                 rev[i][j] = m[i][j] - a.m[i][j];
18         return rev;
19     }
20     Matrix operator *(const Matrix &a) {
21         Matrix rev(r, a.c);
22         Matrix tmp(a.c, a.r);
23         for (int i = 0; i < a.r; ++i)
24             for (int j = 0; j < a.c; ++j)
25                 tmp[j][i] = a.m[i][j];
26         for (int i = 0; i < r; ++i)
27             for (int j = 0; j < a.c; ++j)
28                 for (int k = 0; k < c; ++k)
29                     rev.m[i][j] += m[i][k] * tmp[j][k];
30         return rev;
31     }
32     // 回傳反矩陣。注意這是 const 方法所以原矩陣不受影響。
33     Matrix inverse() const {
34         Matrix t(r, r + c);
35         for (int y = 0; y < r; y++) {
36             t.m[y][c + y] = 1;
37             for (int x = 0; x < c; x++) t.m[y][x] = m[y][x];
38         }
39         if (!t.gauss()) return Matrix(0, 0);
40         Matrix ret(c, r);
41         for (int y = 0; y < r; y++)
42             for (int x = 0; x < c; x++)
43                 ret[y][x] = t.m[y][c + x] / t.m[y][y];
44         return ret;
```

```
45     }
46     // 做高斯消去（最高次係數應置於最左，常數應置於最右）並回
         傳 det
47     // 行列式值。複雜度 O(n^3)。如果不是方陣，回傳值無意義。
48     ll gauss() {
49         vector<ll> lazy(r, 1);
50         bool sign = false;
51         for (int i = 0; i < r; ++i) {
52             if (m[i][i] == 0) {
53                 int j = i + 1;
54                 while (j < r && !m[j][i]) j++;
55                 if (j == r) continue;
56                 m[i].swap(m[j]); sign = !sign;
57             }
58             for (int j = 0; j < r; ++j) {
59                 if (i == j) continue;
60                 lazy[j] = lazy[j] * m[i][i];
61                 ll mx = m[j][i];
62                 for (int k = 0; k < c; ++k)
63                     m[j][k] =
64                         m[j][k] * m[i][i] - m[i][k] * mx;
65             }
66         }
67         ll det = sign ? -1 : 1;
68         for (int i = 0; i < r; ++i) {
69             det = det * m[i][i] / lazy[i];
70             for (auto &j : m[i]) j /= lazy[i];
71         }
72         return det;
73     }
74 };
```

6.7 Karatsuba

```
1  // N is power of 2
2  template<typename Iter>
3  void DC(int N, Iter tmp, Iter A, Iter B, Iter res){
4      fill(res,res+2*N,0);
5      if (N<=32){
6          for (int i=0; i<N; i++)
7              for (int j=0; j<N; j++)
8                  res[i+j] += A[i]*B[j];
9          return;
10     }
11     int n = N/2;
12     auto a = A+n, b = A;
13     auto c = B+n, d = B;
14     DC(n,tmp+N,a,c,res+2*N);
15     for (int i=0; i<N; i++){
16         res[i+N] += res[2*N+i];
17         res[i+n] -= res[2*N+i];
18     }
19     DC(n,tmp+N,b,d,res+2*N);
20     for (int i=0; i<N; i++){
21         res[i] += res[2*N+i];
22         res[i+n] -= res[2*N+i];
23     }
24     auto x = tmp;
25     auto y = tmp+n;
26     for (int i=0; i<n; i++) x[i] = a[i]+b[i];
27     for (int i=0; i<n; i++) y[i] = c[i]+d[i];
28     DC(n,tmp+N,x,y,res+2*N);
```

```
29     for (int i=0; i<N; i++)
30         res[i+n] += res[2*N+i];
31 }
32 // DC(1<<16,tmp.begin(),A.begin(),B.begin(),res.begin());
```

6.8 Euler Function

```
1  // 查詢 phi(x) 亦即比 x 小且與 x 互質的數的數量。
2  int phi(int x) {
3      int r = x;
4      for (int p = 2; p * p <= x; p++) {
5          if (x % p == 0) {
6              while (x % p == 0) x /= p;
7              r -= r / p;
8          }
9      }
10     if (x > 1) r -= r / x;
11     return r;
12 }
13 // 查詢所有 phi(x)，且 x in [0, n)。注意右開區間，回傳陣
       列。
14 vector<int> phi_in(int n) {
15     vector<bool> p(n, 1); vector<int> r(n);
16     p[0] = p[1] = 0;
17     for (int i = 0; i < n; i++) r[i] = i;
18     for (int i = 2; i < n; i++) {
19         if (!p[i]) continue;
20         r[i]--;
21         for (int j = i * 2; j < n; j += i)
22             p[j] = 0, r[j] = r[j] / i * (i - 1);
23     }
24     r[1] = 0;
25     return r;
26 }
```

6.9 Miller Rabin

```
1  typedef long long LL;
2  inline LL mul(LL a,LL b,LL m){//a*b%m
3      return (a%m)*(b%m)%m;
4  }
5
6  template<typename T> bool isprime(T n, int num=3) { //num =
       3,7
7      int sprp[3] = {2,7,61}; //int範圍可解
8      //int llsprp[7] =
          {2,325,9375,28178,450775,9780504,1795265022}; //至少
          unsigned long long範圍
9      if(n==2) return true;
10     if(n<2 || n%2==0) return false;
11     //n-1 = u * 2^t
12     int t = 0;  T u = n-1;
13     while(u%2==0) u >>= 1, t++;
14     for(int i=0; i<num; i++) {
15         T a = sprp[i]%n;
16         if(a==0 || a==1 || a==n-1) continue;
17         T x = fpow(a,u,n);
18         if(x==1 || x==n-1) continue;
19         for(int j=1; j<t; j++) {
```

```
20          x = mul(x,x,n);
21          if(x==1) return false;
22          if(x==n-1) break;
23      }
24      if(x!=n-1) return false;
25  } return true;
26 }
```

6.10 質因數分解

```
 1 typedef __int128 ll;
 2 vector<ll> vv;
 3
 4 /* fastoi here */
 5
 6 ll abs(ll x){
 7     return (x>0?x:-x);
 8 }
 9 ll func(ll t,ll c,ll x) {
10   return (t*t+c)%x;
11 }
12 ll Pollard_Rho(ll x) {
13     ll t = 0;
14     ll c = rand() % (x - 1) + 1;
15     for (int i = 1; i < 1145; ++i) t = func(t, c, x);
16     ll s = t;
17     int step = 0, goal = 1;
18     ll val = 1;
19     for (goal = 1;; goal <<= 1, s = t, val = 1) {
20         for (step = 1; step <= goal; ++step) {
21             t = func(t, c, x);
22             val = val * abs(t - s) % x;
23             if (!val) return x;
24             if (step % 127 == 0) {
25                 ll d = __gcd(val, x);
26                 if (d > 1) return d;
27             }
28         }
29         ll d = __gcd(val, x);
30         if (d > 1) return d;
31     }
32 }
33 void prefactor(ll &n, vector<ll> &v) {
34     ll prime[12] = {2,3,5,7,11,13,17,19,23,29,31,37};
35   for(int i=0;i<12;++i) {
36     while(n%prime[i]==0) {
37       v.push_back(prime[i]);
38       n/=prime[i];
39     }
40   }
41 }
42 void comfactor(const ll &n, vector<ll> &v) {
43   if(isPrime(n,15)) { // MillerRabin
44     v.push_back(n);
45     return;
46   }
47   ll d = Pollard_Rho(n);
48   comfactor(d,v);
49   comfactor(n/d,v);
50 }
51 void Factor(const ll &x, vector<ll> &v) {
52   ll n = x;
53   if(n==1) { puts("Factor 1"); return; }
```

```
54   prefactor(n,v);
55   if(n==1) return;
56   comfactor(n,v);
57   sort(v.begin(),v.end());
58 }
59 void AllFactor(const ll &n,vector<ll> &v) {
60   vector<ll> tmp;
61   Factor(n,tmp);
62   v.clear();
63   v.push_back(1);
64   ll len;
65   ll now=1;
66   ll lentmp = tmp.size();
67   for(int i=0;i<lentmp;++i) {
68     if(i==0 || tmp[i]!=tmp[i-1]) {
69       len = v.size();
70       now = 1;
71     }
72     now*=tmp[i];
73     for(int j=0;j<len;++j)
74       v.push_back(v[j]*now);
75   }
76 }
77 void prime_factorization(){
78     srand(time(NULL));
79     ll n = read();
80     AllFactor(n,vv);
81     sort(vv.begin(),vv.end());
82     for(auto i:vv){
83         print(i); putchar(' ');
84     }
85 }
```

6.11 質數

```
 1 12721       13331       14341       75577
 2 123457      222557      556679      880301
 3 999983      1e6+99      1e9+9       2e9+99
 4 1e12+39     1e15+37     1e9+7       1e7+19
 5 1097774749  1076767633  100102021
 6 999997771   1001010013  1000512343
 7 987654361   999991231   999888733
 8 98789101    987777733   999991921
 9 1010101333  1010102101
10 2305843009213693951     4611686018427387847
11 9223372036854775783     18446744073709551557
```

6.12 實根

```
 1 // an*x^n + ... + a1x + a0 = 0;
 2 int sign(double x){
 3     return x < -eps ? -1 : x > eps;
 4 }
 5 double get(const vector<double>&coef, double x){
 6     double e = 1, s = 0;
 7     for(auto i : coef) s += i*e, e *= x;
 8     return s;
 9 }
10 double find(const vector<double>&coef, int n, double lo,
            double hi){
```

```
11     double sign_lo, sign_hi;
12     if( !(sign_lo = sign(get(coef,lo))) ) return lo;
13     if( !(sign_hi = sign(get(coef,hi))) ) return hi;
14     if(sign_lo * sign_hi > 0) return INF;
15     for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
16         double m = (lo+hi)/2.0;
17         int sign_mid = sign(get(coef,m));
18         if(!sign_mid) return m;
19         if(sign_lo*sign_mid < 0) hi = m;
20         else lo = m;
21     }
22     return (lo+hi)/2.0;
23 }
24 vector<double> cal(vector<double>coef, int n){
25     vector<double>res;
26     if(n == 1){
27         if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
28         return res;
29     }
30     vector<double>dcoef(n);
31     for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
32     vector<double>droot = cal(dcoef, n-1);
33     droot.insert(droot.begin(), -INF);
34     droot.pb(INF);
35     for(int i = 0; i+1 < droot.size(); ++i){
36         double tmp = find(coef, n, droot[i], droot[i+1]);
37         if(tmp < INF) res.pb(tmp);
38     }
39     return res;
40 }
41 int main () {
42     vector<double>ve;
43     vector<double>ans = cal(ve, n);
44     // 視情況把答案 +eps，避免 -0
45 }
```

6.13 FFT

```
 1 template<typename T,typename VT=vector<complex<T> > >
 2 struct FFT{
 3     const T pi;
 4     FFT(const T pi=acos((T)-1)):pi(pi){}
 5     unsigned bit_reverse(unsigned a,int len){
 6         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
 7         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
 8         a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
 9         a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
10         a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
11         return a>>(32-len);
12     }
13     void fft(bool is_inv,VT &in,VT &out,int N){
14         int bitlen=__lg(N),num=is_inv?-1:1;
15         for(int i=0;i<N;++i) out[bit_reverse(i,bitlen)]=in[i
            ];
16         for(int step=2; step<=N; step<<=1){
17             const int mh = step>>1;
18             for(int i=0; i<mh; ++i){
19                 complex<T> wi = exp(complex<T>(0,i*num*pi/mh)
                    );
20                 for(int j=i; j<N; j+=step){
21                     int k = j+mh;
22                     complex<T> u = out[j], t = wi*out[k];
23                     out[j] = u+t;
```

```
24            out[k] = u-t;
25          }
26        }
27      }
28      if(is_inv) for(int i=0;i<N;++i) out[i]/=N;
29    }
30 };
```

6.14 NTT

```
1  template<typename T,typename VT=std::vector<T> >
2  struct NTT{
3    const T P,G;
4    NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
5    inline unsigned int bit_reverse(unsigned int a,int len){
6      a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
7      a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
8      a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
9      a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
10     a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
11     return a>>(32-len);
12   }
13   inline T pow_mod(T n,T k,T m){
14     T ans=1;
15     for(n=(n>=m?n%m:n);k;k>>=1){
16       if(k&1)ans=ans*n%m;
17       n=n*n%m;
18     } return ans;
19   }
20   inline void ntt(bool is_inv,VT &in,VT &out,int N){
21     int bitlen=std::__lg(N);
22     for(int i=0;i<N;++i)out[bit_reverse(i,bitlen)]=in[i];
23     for(int step=2,id=1;step<=N;step<<=1,++id){
24       T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
25       const int mh=step>>1;
26       for(int i=0;i<mh;++i){
27         for(int j=i;j<N;j+=step){
28           u = out[j], t = wi*out[j+mh]%P;
29           out[j] = u+t;
30           out[j+mh] = u-t;
31           if(out[j]>=P)out[j]-=P;
32           if(out[j+mh]<0)out[j+mh]+=P;
33         }
34         wi = wi*wn%P;
35       }
36     }
37     if(is_inv){
38       for(int i=1;i<N/2;++i)std::swap(out[i],out[N-i]);
39       T invn=pow_mod(N,P-2,P);
40       for(int i=0;i<N;++i)out[i]=out[i]*invn%P;
41     }
42   }
43 };
44 #endif
```

6.15 Simplex

```
1  /*target:
2    max \sum_{j=1}^n A_{0,j}*x_j
3  condition:
4    \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0}  |i=1~m
5    x_j >= 0  |j=1~n
6  VDB = vector<double>*/
7  template<class VDB>
8  VDB simplex(int m,int n,vector<VDB> a){
9    vector<int> left(m+1), up(n+1);
10   iota(left.begin(), left.end(), n);
11   iota(up.begin(), up.end(), 0);
12   auto pivot = [&](int x, int y){
13     swap(left[x], up[y]);
14     auto k = a[x][y]; a[x][y] = 1;
15     vector<int> pos;
16     for(int j = 0; j <= n; ++j){
17       a[x][j] /= k;
18       if(a[x][j] != 0) pos.push_back(j);
19     }
20     for(int i = 0; i <= m; ++i){
21       if(a[i][y]==0 || i == x) continue;
22       k = a[i][y], a[i][y] = 0;
23       for(int j : pos) a[i][j] -= k*a[x][j];
24     }
25   };
26   for(int x,y;;){
27     for(int i=x=1; i <= m; ++i)
28       if(a[i][0]<a[x][0]) x = i;
29     if(a[x][0]>=0) break;
30     for(int j=y=1; j <= n; ++j)
31       if(a[x][j]<a[x][y]) y = j;
32     if(a[x][y]>=0) return VDB();//infeasible
33     pivot(x, y);
34   }
35   for(int x,y;;){
36     for(int j=y=1; j <= n; ++j)
37       if(a[0][j] > a[0][y]) y = j;
38     if(a[0][y]<=0) break;
39     x = -1;
40     for(int i=1; i<=m; ++i) if(a[i][y] > 0)
41       if(x == -1 || a[i][0]/a[i][y]
42         < a[x][0]/a[x][y]) x = i;
43     if(x == -1) return VDB();//unbounded
44     pivot(x, y);
45   }
46   VDB ans(n + 1);
47   for(int i = 1; i <= m; ++i)
48     if(left[i] <= n) ans[left[i]] = a[i][0];
49   ans[0] = -a[0][0];
50   return ans;
51 }
```

6.16 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
     值。如果
3   * 格式不合法，會丟出錯誤。複雜度 O(字串長度) 。支援的符號有
     四則運算
4   * 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
     頭可以為
5   * 零或禁止為零。可以兼容或禁止多重前置號（例如 --1 視為 1 、
     +-+-1
6   * 視為 -1）。空字串視為不合法。運算範圍限於 long long 。如果
     試圖除
7   * 以零或對零求餘也會丟出錯誤。
8   */
9  void req(bool b) { if (!b) throw ""; }
10 const int B = 2; // 可以調整成 B 進位
11 class Expr {
12   private:
13     deque<char> src;
14     Expr(const string& s) : src(s.begin(), s.end()) {}
15     inline char top() {
16       return src.empty() ? '\0' : src.front();
17     }
18     inline char pop() {
19       char c = src.front(); src.pop_front(); return c;
20     }
21     ll n() {
22       ll ret = pop() - '0';
23       // 若要禁止數字以 0 開頭，加上這行
24       // req(ret || !isdigit(top()));
25       while (isdigit(top())) ret = B * ret + pop() - '0';
26       return ret;
27     }
28     ll fac() {
29       if (isdigit(top())) return n();
30       if (top() == '-') { pop(); return -fac(); }
31       if (top() == '(') {
32         pop();
33         ll ret = expr(1);
34         req(pop() == ')');
35         return ret;
36       }
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 };
```

6.17 Pick's Theorem

```cpp
/* i:number of integer points interior to the polygon
b:the number of integer points on its boundary (including
    both vertices and points along the sides).
Then the area A of this polygon is: A = i + b/2 - 1 */

pair<ll, ll> operator-(const pair<ll, ll>& a, const pair<ll,
    ll>& b) {
  return {a.first - b.first, a.second - b.second};
}

int n;
pair<ll, ll> p[100010];

ll Pick() {
  cin >> n;
  for(int i = 0; i < n; ++i)
    cin >> p[i].first >> p[i].second;
  p[n] = p[0];
  ll area = 0;
  for(int i = 0; i < n; ++i)
    area += p[i].first * p[i + 1].second - p[i].second * p[i
        + 1].first;
  area = abs(area);
  ll b = 0;
  for(int i = 0; i < n; ++i) {
    pair<ll, ll> v = p[i + 1] - p[i];
    b += abs(__gcd(v.first, v.second));
  }
  ll a = (area + 2 - b) / 2;
  return a;
}
```

6.18 擴展歐幾里德

```cpp
// 給 a,b ，解 ax+by=gcd(a,b)
typedef pair<ll, ll> pii;
pii extgcd(ll a, ll b) {
    if (b == 0) return {1, 0};
    ll k = a / b;
    pii p = extgcd(b, a - k * b);
    return {p.second, p.first - k * p.second};
}
```

6.19 線性篩

```cpp
int prime[MAXN];
vector<int> p;
void sieve(int n){
    fill(prime+2,prime+n+1,1);
    for(int i=2;i<=n;++i){
        if(prime[i]==1) p.push_back(i);
        for(int j:p){
            if(i*j>n) break;
            prime[i*j]=j; //順便紀錄最小的質因數是誰
            if(i%j==0) break; //表示後面的質數都大於最小質因
                數了
        }
    }
}
```

6.20 linear_inv

```cpp
ll arr[max_n],pre[max_n],inv[max_n];

void linear_inv(){
    pre[1] = arr[1];
    pre[0] = 1;
    for(ll i=2;i<=n;i++){
        arr[i] = max(1ll,(m*arr[i-1]+k)%mod);
        pre[i] = (pre[i-1]*arr[i])%mod;
    }
    hehe[n] = fpow(pre[n],mod-2);
    inv[n] = (hehe[n] * pre[n-1])%mod;
    for(ll i=n-1;i>=1;i--){
        hehe[i] = (hehe[i+1]*arr[i+1])%mod;
        inv[i] = (hehe[i] * pre[i-1])%mod;
    }
}
/* (a*b*c)^-1 ➔ (a*b*c)
(a*b*c)^-1 * c ➔ (a*b*c*c) ➔ (a*b)^-1
c^-1 ➔ (a*b*c)^-1 * (a*b) */
```

7 String

7.1 Rolling Hash

```cpp
// 問 pat 在 str 第一次出現的開頭 index 。-1 表示找不到。
int rollhash(string& str, string& pat) {
    const ll x = 1e6 + 99;  // 隨意大質數，建議 1e6
    const ll m = 1e9 + 9;   // 隨意大質數，建議 1e9
    assert(pat.size());      // pat 不能是空字串
    ll xx = 1, sh = 0;
    for (char c : pat)
        sh = (sh * x + c) % m, xx = xx * x % m;
    deque<ll> hash = {0};
    int ret = 0;
    for (char c : str) {
        hash.push_back((hash.back() * x + c) % m);
        if (hash.size() <= pat.size()) continue;
        ll h = hash.back() - hash.front() * xx;
        h = (h % m + m) % m;
        if (h == sh) return ret;
        hash.pop_front();
        ret++;
    } return -1;
}
```

7.2 Trie

```cpp
class Trie {
private:
    struct Node {
        int cnt = 0, sum = 0;
        Node *tr[128] = {};
        ~Node() {
            for (int i = 0; i < 128; i++)
                if (tr[i]) delete tr[i];
        }
    };
    Node *root;
public:
    void insert(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) ptr->tr[*s] = new Node();
            ptr = ptr->tr[*s];
            ptr->sum++;
        }
        ptr->cnt++;
    }
    inline int count(char *s) {
        Node *ptr = find(s);
        return ptr ? ptr->cnt : 0;
    }
    Node *find(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) return 0;
            ptr = ptr->tr[*s];
        } return ptr;
    }
    bool erase(char *s) {
        Node *ptr = find(s);
        if (!ptr) return false;
        int num = ptr->cnt;
        if (!num) return false;
        ptr = root;
        for (; *s; s++) {
            Node *tmp = ptr;
            ptr = ptr->tr[*s];
            ptr->sum -= num;
            if (!ptr->sum) {
                delete ptr;
                tmp->tr[*s] = 0;
                return true;
            }
        }
    }
    Trie() { root = new Node(); }
    ~Trie() { delete root; }
};
```

7.3 AC 自動機

```cpp
template<char L='a',char R='z'>
class ac_automaton{
    struct joe{
        int next[R-L+1], fail, efl, ed, cnt_dp, vis;
        joe():ed(0),cnt_dp(0),vis(0){
            for(int i=0; i<=R-L; i++) next[i]=0;
        }
    };
```

```cpp
 9  public:
10    std::vector<joe> S;
11    std::vector<int> q;
12    int qs,qe,vt;
13    ac_automaton():S(1),qs(0),qe(0),vt(0){}
14    void clear(){
15      q.clear();
16      S.resize(1);
17      for(int i=0; i<=R-L; i++) S[0].next[i] = 0;
18      S[0].cnt_dp = S[0].vis = qs = qe = vt = 0;
19    }
20    void insert(const char *s){
21      int o = 0;
22      for(int i=0,id; s[i]; i++){
23        id = s[i]-L;
24        if(!S[o].next[id]){
25          S.push_back(joe());
26          S[o].next[id] = S.size()-1;
27        }
28        o = S[o].next[id];
29      }
30      ++S[o].ed;
31    }
32    void build_fail(){
33      S[0].fail = S[0].efl = -1;
34      q.clear();
35      q.push_back(0);
36      ++qe;
37      while(qs!=qe){
38        int pa = q[qs++], id, t;
39        for(int i=0;i<=R-L;i++){
40          t = S[pa].next[i];
41          if(!t)continue;
42          id = S[pa].fail;
43          while(~id && !S[id].next[i]) id = S[id].fail;
44          S[t].fail = ~id ? S[id].next[i] : 0;
45          S[t].efl = S[S[t].fail].ed ? S[t].fail : S[S[t].fail].efl;
46          q.push_back(t);
47          ++qe;
48        }
49      }
50    }
51    /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的
          次數O(N+M)*/
52    int match_0(const char *s){
53      int ans = 0, id, p = 0, i;
54      for(i=0; s[i]; i++){
55        id = s[i]-L;
56        while(!S[p].next[id] && p) p = S[p].fail;
57        if(!S[p].next[id])continue;
58        p = S[p].next[id];
59        ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算
              )*/
60      }
61      for(i=qe-1; i>=0; --i){
62        ans += S[q[i]].cnt_dp * S[q[i]].ed;
63        if(~S[q[i]].fail) S[S[q[i]].fail].cnt_dp += S[q[i]].
              cnt_dp;
64      }
65      return ans;
66    }
67    /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)
          */
68    int match_1(const char *s)const{
```

```cpp
69      int ans = 0, id, p = 0, t;
70      for(int i=0; s[i]; i++){
71        id = s[i]-L;
72        while(!S[p].next[id] && p) p = S[p].fail;
73        if(!S[p].next[id])continue;
74        p = S[p].next[id];
75        if(S[p].ed) ans += S[p].ed;
76        for(t=S[p].efl; ~t; t=S[t].efl){
77          ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
78        }
79      }
80      return ans;
81    }
82    /*枚舉(s的子字串⊇A)的所有相異字串各恰一次並傳回次數O(N*M
          ^(1/3))*/
83    int match_2(const char *s){
84      int ans=0, id, p=0, t;
85      ++vt;
86      /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成
              false
87      這種利用vt的方法可以O(1)歸零vis陣列*/
88      for(int i=0; s[i]; i++){
89        id = s[i]-L;
90        while(!S[p].next[id]&&p)p = S[p].fail;
91        if(!S[p].next[id])continue;
92        p = S[p].next[id];
93        if(S[p].ed && S[p].vis!=vt){
94          S[p].vis = vt;
95          ans += S[p].ed;
96        }
97        for(t=S[p].efl; ~t && S[t].vis!=vt; t=S[t].efl){
98          S[t].vis = vt;
99          ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
100       }
101     }
102     return ans;
103   }
104   /*把AC自動機變成真的自動機*/
105   void evolution(){
106     for(qs=1; qs!=qe;){
107       int p = q[qs++];
108       for(int i=0; i<=R-L; i++)
109         if(S[p].next[i]==0) S[p].next[i] = S[S[p].fail].next[
              i];
110     }
111   }
112 };
```

7.4 KMP

```cpp
 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) {
```

```cpp
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  }
```

7.5 Z

```cpp
 1  void z_build(string &s, int *z) {
 2      int bst = z[0] = 0;
 3      for (int i = 1; s[i]; i++) {
 4          if (z[bst] + bst < i) z[i] = 0;
 5          else z[i] = min(z[bst] + bst - i, z[i - bst]);
 6          while (s[z[i]] == s[i + z[i]]) z[i]++;
 7          if (z[i] + i > z[bst] + bst) bst = i;
 8      }
 9  }
10  // Queries how many times s appears in t
11  int z_match(string &s, string &t) {
12      int ans = 0;
13      int lens = s.length(), lent = t.length();
14      int z[lens + lent + 5];
15      string st = s + "$" + t;
16      z_build(st, z);
17      for (int i = lens + 1; i <= lens + lent; i++)
18          if (z[i] == lens) ans++;
19      return ans;
20  }
```

7.6 BWT

```cpp
 1  const int N = 8;           // 字串長度
 2  int s[N+N+1] = "suffixes";  // 字串，後面預留一倍空間。
 3  int sa[N];                 // 後綴陣列
 4  int pivot;
 5  int cmp(const void* i, const void* j) {
 6      return strncmp(s+*(int*)i, s+*(int*)j, N);
 7  }
 8  // 此處便宜行事，採用 O(N²logN) 的後綴陣列演算法。
 9  void BWT() {
10      strncpy(s + N, s, N);
11      for (int i=0; i<N; ++i) sa[i] = i;
12      qsort(sa, N, sizeof(int), cmp);
13      // 當輸入字串的所有字元都相同，必須當作特例處理。
```

```
14        // 或者改用stable sort。
15        for (int i=0; i<N; ++i)
16            cout << s[(sa[i] + N-1) % N];
17        for (int i=0; i<N; ++i)
18            if (sa[i] == 0) {
19                pivot = i;
20                break;
21            }
22    }
23    // Inverse BWT
24    const int N = 8;                    // 字串長度
25    char t[N+1] = "xuffessi";    // 字串
26    int pivot;
27    int next[N];
28    void IBWT() {
29        vector<int> index[256];
30        for (int i=0; i<N; ++i)
31            index[t[i]].push_back(i);
32        for (int i=0, n=0; i<256; ++i)
33            for (int j=0; j<index[i].size(); ++j)
34                next[n++] = index[i][j];
35        int p = pivot;
36        for (int i=0; i<N; ++i)
37            cout << t[p = next[p]];
38    }
```

7.7   Suffix_Array_LCP

```
1  #define radix_sort(x,y){
2    for(i=0;i<A;++i) c[i] = 0;
3    for(i=0;i<n;++i) c[x[y[i]]]++;
4    for(i=1;i<A;++i) c[i] += c[i-1];
5    for(i=n-1;~i;--i) sa[--c[x[y[i]]]] = y[i];
6  }
7  #define AC(r,a,b) r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
8  void suffix_array(const char *s,int n,int *sa,int *rank,int *
       tmp,int *c){
9    int A='z'+1,i,k,id=0;
10   for(i=0; i<n; ++i)rank[tmp[i]=i]=s[i];
11   radix_sort(rank,tmp);
12   for(k=1; id<n-1; k<<=1){
13     for(id=0,i=n-k; i<n; ++i) tmp[id++]=i;
14     for(i=0; i<n; ++i)
15       if(sa[i]>=k) tmp[id++]=sa[i]-k;
16     radix_sort(rank,tmp);
17     swap(rank,tmp);
18     for(rank[sa[0]]=id=0,i=1; i<n; ++i)
19       rank[sa[i]] = id+=AC(tmp,sa[i-1],sa[i]);
20     A = id+1;
21   }
22 }
23 //h:高度數組 sa:後綴數組 rank:排名
24 void suffix_array_lcp(const char *s,int len,int *h,int *sa,
       int *rank){
25   for(int i=0; i<len; ++i)rank[sa[i]]=i;
26   for(int i=0,k=0; i<len; ++i){
27     if(rank[i]==0)continue;
28     if(k)--k;
29     while(s[i+k]==s[sa[rank[i]-1]+k])++k;
30     h[rank[i]]=k;
31   }
32   h[0]=0;// h[k]=lcp(sa[k],sa[k-1]);
```

7.8   LPS

```
1  char t[1001];              // 原字串
2  char s[1001 * 2];          // 穿插特殊字元之後的t
3  int z[1001 * 2], L, R;   // 源自Gusfield's Algorithm
4  // 由a往左、由b往右,對稱地作字元比對。
5  int extend(int a, int b) {
6      int i = 0;
7      while (a-i>=0 && b+i<N && s[a-i] == s[b+i]) i++;
8      return i;
9  }
10 void longest_palindromic_substring() {
11     int N = strlen(t);
12     // t穿插特殊字元,存放到s。
13     // (實際上不會這麼做,都是細算索引值。)
14     memset(s, '.', N*2+1);
15     for (int i=0; i<N; ++i) s[i*2+1] = t[i];
16     N = N*2+1;
17     // s[N] = '\0';      // 可做可不做
18     // Manacher's Algorithm
19     z[0] = 1; L = R = 0;
20     for (int i=1; i<N; ++i)  {
21         int ii = L - (i - L);     // i的映射位置
22         int n = R + 1 - i;
23         if (i > R)  {
24             z[i] = extend(i, i);
25             L = i;
26             R = i + z[i] - 1;
27         } else if (z[ii] == n)  {
28             z[i] = n + extend(i-n, i+n);
29             L = i;
30             R = i + z[i] - 1;
31         } else z[i] = min(z[ii], n);
32     }
33     // 尋找最長迴文子字串的長度。
34     int n = 0, p = 0;
35     for (int i=0; i<N; ++i)
36         if (z[i] > n)  n = z[p = i];
37     // 記得去掉特殊字元。
38     cout << "最長迴文子字串的長度是" << (n-1) / 2;
39     // 印出最長迴文子字串,記得別印特殊字元。
40     for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
41         if (i & 1)  cout << s[i];
42 }
```

7.9   Edit Distance

```
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   Geometry

```
1  //Copy from Jinkela
2  const double PI=atan2(0.0,-1.0);
3  template<typename T>
4  struct point{
5    T x,y;
6    point(){}
7    point(const T&x,const T&y):x(x),y(y){}
8    point operator+(const point &b)const{
9      return point(x+b.x,y+b.y); }
10   point operator-(const point &b)const{
11     return point(x-b.x,y-b.y); }
12   point operator*(const T &b)const{
13     return point(x*b,y*b); }
14   point operator/(const T &b)const{
15     return point(x/b,y/b); }
16   bool operator==(const point &b)const{
17     return x==b.x&&y==b.y; }
18   T dot(const point &b)const{
19     return x*b.x+y*b.y; }
20   T cross(const point &b)const{
21     return x*b.y-y*b.x; }
22   point normal()const{//求法向量
23     return point(-y,x); }
24   T abs2()const{//向量長度的平方
25     return dot(*this); }
26   T rad(const point &b)const{//兩向量的弧度
27 return fabs(atan2(fabs(cross(b)),dot(b))); }
28   T getA()const{//對x軸的弧度
29     T A=atan2(y,x);//超過180度會變負的
30     if(A<=-PI/2)A+=PI*2;
31     return A;
32   }
33 };
34 template<typename T>
35 struct line{
36   line(){}
37   point<T> p1,p2;
38   T a,b,c;//ax+by+c=0
39   line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
40   void pton(){//轉成一般式
41     a=p1.y-p2.y;
42     b=p2.x-p1.x;
43     c=-a*p1.x-b*p1.y;
```

```cpp
    }
    T ori(const point<T> &p)const{//點和有向直線的關係，>0左
        邊、=0在線上<0右邊
        return (p2-p1).cross(p-p1);
    }
    T btw(const point<T> &p)const{//點投影落在線段上<=0
        return (p1-p).dot(p2-p);
    }
    bool point_on_segment(const point<T>&p)const{//點是否在線段
        上
        return ori(p)==0&&btw(p)<=0;
    }
    T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線
        /線段的距離平方
        point<T> v=p2-p1,v1=p-p1;
        if(is_segment){
            point<T> v2=p-p2;
            if(v.dot(v1)<=0)return v1.abs2();
            if(v.dot(v2)>=0)return v2.abs2();
        }
        T tmp=v.cross(v1);
        return tmp*tmp/v.abs2();
    }
    T seg_dis2(const line<T> &l)const{//兩線段距離平方
        return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2
            (p2,1)});
    }
    point<T> projection(const point<T> &p)const{//點對直線的投
        影
        point<T> n=(p2-p1).normal();
        return p-n*(p-p1).dot(n)/n.abs2();
    }
    point<T> mirror(const point<T> &p)const{
        //點對直線的鏡射，要先呼叫pton轉成一般式
        point<T> R;
        T d=a*a+b*b;
        R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
        R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
        return R;
    }
    bool equal(const line &l)const{//直線相等
        return ori(l.p1)==0&&ori(l.p2)==0;
    }
    bool parallel(const line &l)const{
        return (p1-p2).cross(l.p1-l.p2)==0;
    }
    bool cross_seg(const line &l)const{
        return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
            //直線是否交線段
    }
    int line_intersect(const line &l)const{//直線相交情況，-1無
        限多點、1交於一點、0不相交
        return parallel(l)?(ori(l.p1)==0?-1:0):1;
    }
    int seg_intersect(const line &l)const{
        T c1=ori(l.p1), c2=ori(l.p2);
        T c3=l.ori(p1), c4=l.ori(p2);
        if(c1==0&&c2==0){//共線
            bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
            T a3=l.btw(p1),a4=l.btw(p2);
            if(b1&&b2&&a3==0&&a4>=0) return 2;
            if(b1&&b2&&a3>=0&&a4==0) return 3;
            if(b1&&b2&&a3>=0&&a4>=0) return 0;
```

```cpp
            return -1;//無限交點
        }else if(c1*c2<=0&&c3*c4<=0)return 1;
        return 0;//不相交
    }
    point<T> line_intersection(const line &l)const{/*直線交點*/
        point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
        //if(a.cross(b)==0)return INF;
        return p1+a*(s.cross(b)/a.cross(b));
    }
    point<T> seg_intersection(const line &l)const{//線段交點
        int res=seg_intersect(l);
        if(res<=0) assert(0);
        if(res==2) return p1;
        if(res==3) return p2;
        return line_intersection(l);
    }
};
template<typename T>
struct polygon{
    polygon(){}
    vector<point<T> > p;//逆時針順序
    T area()const{//面積
        T ans=0;
        for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
            ans+=p[i].cross(p[j]);
        return ans/2;
    }
    point<T> center_of_mass()const{//重心
        T cx=0,cy=0,w=0;
        for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
            T a=p[i].cross(p[j]);
            cx+=(p[i].x+p[j].x)*a;
            cy+=(p[i].y+p[j].y)*a;
            w+=a;
        }
        return point<T>(cx/3/w,cy/3/w);
    }
    char ahas(const point<T>& t)const{//點是否在簡單多形內，
        是的話回傳1、在邊上回傳-1、否則回傳0
        bool c=0;
        for(int i=0,j=p.size()-1;i<p.size();j=i++)
            if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
            else if((p[i].y>t.y)!=(p[j].y>t.y)&&
            t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x
                )
                c=!c;
        return c;
    }
    char point_in_convex(const point<T>&x)const{
        int l=1,r=(int)p.size()-2;
        while(l<=r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳
            -1、否則回傳0
            int mid=(l+r)/2;
            T a1=(p[mid]-p[0]).cross(x-p[0]);
            T a2=(p[mid+1]-p[0]).cross(x-p[0]);
            if(a1>=0&&a2<=0){
                T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
                return res>0?1:(res>=0?-1:0);
            }else if(a1<0)r=mid-1;
            else l=mid+1;
        }
        return 0;
    }
    vector<T> getA()const{//凸包邊對x軸的夾角
```

```cpp
        vector<T>res;//一定是遞增的
        for(size_t i=0;i<p.size();++i)
            res.push_back((p[(i+1)%p.size()]-p[i]).getA());
        return res;
    }
    bool line_intersect(const vector<T>&A,const line<T> &l)
        const{//O(logN)
        int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-
            A.begin();
        int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-
            A.begin();
        return l.cross_seg(line<T>(p[f1],p[f2]));
    }
    polygon cut(const line<T> &l)const{//凸包對直線切割，得到直
        線l左側的凸包
        polygon ans;
        for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
            if(l.ori(p[i])>=0){
                ans.p.push_back(p[i]);
                if(l.ori(p[j])<0)
                    ans.p.push_back(l.line_intersection(line<T>(p[i],p[
                        j])));
            }else if(l.ori(p[j])>0)
                ans.p.push_back(l.line_intersection(line<T>(p[i],p[j
                    ])));
        }
        return ans;
    }
    static bool graham_cmp(const point<T>& a,const point<T>& b)
        {//凸包排序函數
        return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
    }
    void graham(vector<point<T> > &s){//凸包
        sort(s.begin(),s.end(),graham_cmp);
        p.resize(s.size()+1);
        int m=0;
        for(size_t i=0;i<s.size();++i){
            while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
            p[m++]=s[i];
        }
        for(int i=s.size()-2,t=m+1;i>=0;--i){
            while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
            p[m++]=s[i];
        }
        if(s.size()>1)--m;
        p.resize(m);
    }
    T diam(){//直徑
        int n=p.size(),t=1;
        T ans=0;p.push_back(p[0]);
        for(int i=0;i<n;i++){
            point<T> now=p[i+1]-p[i];
            while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
                +1)%n;
            ans=max(ans,(p[i]-p[t]).abs2());
        }
        return p.pop_back(),ans;
    }
    T min_cover_rectangle(){//最小覆蓋矩形
        int n=p.size(),t=1,r=1,l;
        if(n<3)return 0;//也可以做最小周長矩形
        T ans=1e99;p.push_back(p[0]);
        for(int i=0;i<n;i++){
            point<T> now=p[i+1]-p[i];
```

```
217        while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
               +1)%n;
218        while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n
               ;
219        if(!i)l=r;
220        while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%
               n;
221        T d=now.abs2();
222        T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(
               p[l]-p[i]))/d;
223        ans=min(ans,tmp);
224      }
225      return p.pop_back(),ans;
226    }
227    T dis2(polygon &pl){//凸包最近距離平方
228      vector<point<T> > &P=p,&Q=pl.p;
229      int n=P.size(),m=Q.size(),l=0,r=0;
230      for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
231      for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
232      P.push_back(P[0]),Q.push_back(Q[0]);
233      T ans=1e99;
234      for(int i=0;i<n;++i){
235        while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
236        ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],
               Q[r+1])));
237        l=(l+1)%n;
238      }
239      return P.pop_back(),Q.pop_back(),ans;
240    }
241    static char sign(const point<T>&t){
242      return (t.y==0?t.x:t.y)<0;
243    }
244    static bool angle_cmp(const line<T>& A,const line<T>& B){
245      point<T> a=A.p2-A.p1,b=B.p2-B.p1;
246      return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
247    }
248    int halfplane_intersection(vector<line<T> > &s){//半平面交
249      sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平
               面
250      int L,R,n=s.size();
251      vector<point<T> > px(n);
252      vector<line<T> > q(n);
253      q[L=R=0]=s[0];
254      for(int i=1;i<n;++i){
255        while(L<R&&s[i].ori(px[R-1])<=0)--R;
256        while(L<R&&s[i].ori(px[L])<=0)++L;
257        q[++R]=s[i];
258        if(q[R].parallel(q[R-1])){
259          --R;
260          if(q[R].ori(s[i].p1)>0)q[R]=s[i];
261        }
262        if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
263      }
264      while(L<R&&q[L].ori(px[R-1])<=0)--R;
265      p.clear();
266      if(R-L<=1)return 0;
267      px[R]=q[R].line_intersection(q[L]);
268      for(int i=L;i<=R;++i)p.push_back(px[i]);
269      return R-L+1;
270    }
271  };
272  template<typename T>
273  struct triangle{
274    point<T> a,b,c;
275    triangle(){}
276    triangle(const point<T> &a,const point<T> &b,const point<T>
             &c):a(a),b(b),c(c){}
277    T area()const{
278      T t=(b-a).cross(c-a)/2;
279      return t>0?t:-t;
280    }
281    point<T> barycenter()const{//重心
282      return (a+b+c)/3;
283    }
284    point<T> circumcenter()const{//外心
285      static line<T> u,v;
286      u.p1=(a+b)/2;
287      u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
288      v.p1=(a+c)/2;
289      v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
290      return u.line_intersection(v);
291    }
292    point<T> incenter()const{//內心
293      T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).
             abs2());
294      return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B
             +C);
295    }
296    point<T> perpencenter()const{//垂心
297      return barycenter()*3-circumcenter()*2;
298    }
299  };
300  template<typename T>
301  struct point3D{
302    T x,y,z;
303    point3D(){}
304    point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
305    point3D operator+(const point3D &b)const{
306      return point3D(x+b.x,y+b.y,z+b.z);}
307    point3D operator-(const point3D &b)const{
308      return point3D(x-b.x,y-b.y,z-b.z);}
309    point3D operator*(const T &b)const{
310      return point3D(x*b,y*b,z*b);}
311    point3D operator/(const T &b)const{
312      return point3D(x/b,y/b,z/b);}
313    bool operator==(const point3D &b)const{
314      return x==b.x&&y==b.y&&z==b.z;}
315    T dot(const point3D &b)const{
316      return x*b.x+y*b.y+z*b.z;}
317    point3D cross(const point3D &b)const{
318      return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
319    T abs2()const{//向量長度的平方
320      return dot(*this);}
321    T area2(const point3D &b)const{//和b、原點圍成面積的平方
322      return cross(b).abs2()/4;}
323  };
324  template<typename T>
325  struct line3D{
326    point3D<T> p1,p2;
327    line3D(){}
328    line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2
             (p2){}
329    T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直
             線/線段的距離平方
330      point3D<T> v=p2-p1,v1=p-p1;
331      if(is_segment){
332        point3D<T> v2=p-p2;
333        if(v.dot(v1)<=0)return v1.abs2();
334        if(v.dot(v2)>=0)return v2.abs2();
335      }
336      point3D<T> tmp=v.cross(v1);
337      return tmp.abs2()/v.abs2();
338    }
339    pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &
             l)const{
340      point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
341      point3D<T> N=v1.cross(v2),ab(p1-l.p1);
342      //if(N.abs2()==0)return NULL;平行或重合
343      T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
344      point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1
             ;
345      T t1=(G.cross(d2)).dot(D)/D.abs2();
346      T t2=(G.cross(d1)).dot(D)/D.abs2();
347      return make_pair(p1+d1*t1,l.p1+d2*t2);
348    }
349    bool same_side(const point3D<T> &a,const point3D<T> &b)
             const{
350      return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
351    }
352  };
353  template<typename T>
354  struct plane{
355    point3D<T> p0,n;//平面上的點和法向量
356    plane(){}
357    plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n)
             {}
358    T dis2(const point3D<T> &p)const{//點到平面距離的平方
359      T tmp=(p-p0).dot(n);
360      return tmp*tmp/n.abs2();
361    }
362    point3D<T> projection(const point3D<T> &p)const{
363      return p-n*(p-p0).dot(n)/n.abs2();
364    }
365    point3D<T> line_intersection(const line3D<T> &l)const{
366      T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
367      return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
368    }
369    line3D<T> plane_intersection(const plane &pl)const{
370      point3D<T> e=n.cross(pl.n),v=n.cross(e);
371      T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
372      point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
373      return line3D<T>(q,q+e);
374    }
375  };
376  template<typename T>
377  struct triangle3D{
378    point3D<T> a,b,c;
379    triangle3D(){}
380    triangle3D(const point3D<T> &a,const point3D<T> &b,const
             point3D<T> &c):a(a),b(b),c(c){}
381    bool point_in(const point3D<T> &p)const{//點在該平面上的投
             影在三角形中
382      return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c).
             same_side(p,b)&&line3D<T>(a,b).same_side(p,c);
383    }
384  };
385  template<typename T>
386  struct tetrahedron{//四面體
387    point3D<T> a,b,c,d;
388    tetrahedron(){}
389    tetrahedron(const point3D<T> &a,const point3D<T> &b,const
             point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d)
             {}
```

```
390    T volume6()const{//體積的六倍
391        return (d-a).dot((b-a).cross(c-a));
392    }
393    point3D<T> centroid()const{
394        return (a+b+c+d)/4;
395    }
396    bool point_in(const point3D<T> &p)const{
397        return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,
                d,a).point_in(p);
398    }
399 };
400 template<typename T>
401 struct convexhull3D{
402    static const int MAXN=1005;
403    struct face{
404        int a,b,c;
405        face(int a,int b,int c):a(a),b(b),c(c){}
406    };
407    vector<point3D<T>> pt;
408    vector<face> ans;
409    int fid[MAXN][MAXN];
410    void build(){
411        int n=pt.size();
412        ans.clear();
413        memset(fid,0,sizeof(fid));
414        ans.emplace_back(0,1,2);//注意不能共線
415        ans.emplace_back(2,1,0);
416        int ftop = 0;
417        for(int i=3, ftop=1; i<n; ++i,++ftop){
418            vector<face> next;
419            for(auto &f:ans){
420                T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.
                    c]-pt[f.a]));
421                if(d<=0) next.push_back(f);
422                int ff=0;
423                if(d>0) ff=ftop;
424                else if(d<0) ff=-ftop;
425                fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
426            }
427            for(auto &f:ans){
428                if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
429                    next.emplace_back(f.a,f.b,i);
430                if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
431                    next.emplace_back(f.b,f.c,i);
432                if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
433                    next.emplace_back(f.c,f.a,i);
434            }
435            ans=next;
436        }
437    }
438    point3D<T> centroid()const{
439        point3D<T> res(0,0,0);
440        T vol=0;
441        for(auto &f:ans){
442            T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
443            res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
444            vol+=tmp;
445        }
446        return res/(vol*4);
447    }
448 };
```

8.2 旋轉卡尺

```
1  typedef pair<ll, ll> pii;
2  #define x first
3  #define y second
4  #define ii (i + 1) % n  // 打字加速！
5  inline pii operator-(const pii& a, const pii& b) {
6      return {a.x - b.x, a.y - b.y};
7  }  // const 不可省略
8  inline ll operator*(const pii& a, const pii& b) {
9      return a.x * b.y - a.y * b.x;
10 }
11 inline ll crzf(const pii& o, const pii& a, const pii& b) {
12     return (a - o) * (b - o)
13 }
14 inline ll dd(const pii& a, const pii& b) {
15     ll dx = a.x - b.x, dy = a.y - b.y;
16     return dx * dx + dy * dy;
17 }
18 // 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。
19 #define jud \
20     crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
21 vector<pii> makepoly(vector<pii>& pp) {
22     int n = pp.size();
23     sort(pp.begin(), pp.end());
24     pp.erase(unique(pp.begin(), pp.end()), pp.end());
25     vector<pii> ret;
26     for (int i = 0; i < n; i++) {
27         while (ret.size() >= 2 && jud) ret.pop_back();
28         ret.push_back(pp[i]);
29     }
30     for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
31         while (ret.size() >= t && jud) ret.pop_back();
32         ret.push_back(pp[i]);
33     }
34     if (n >= 2) ret.pop_back();
35     return ret;
36 }
37 // (shoelace formula)
38 // 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
39 ll area(vector<pii>& poly) {
40     int n = poly.size();
41     ll ret = 0;
42     for (int i = 0; i < n; i++)
43         ret += (poly[i].x * poly[ii].y);
44     for (int i = 0; i < n; i++)
45         ret -= (poly[i].y * poly[ii].x);
46     return ret;
47 }
48 // 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
49 // 距離，請先轉成凸包。若凸包少於兩個點，回傳零。
50 #define kk (k + 1) % n
51 ll maxdist(vector<pii>& poly) {
52     int k = 1, n = poly.size();
53     if (n < 2) return 0;
54     if (n == 2) return dd(poly[0], poly[1]);
55     ll ret = 0;
56     for (int i = 0; i < n; i++) {
57         while (abs(crzf(poly[kk], poly[i], poly[ii])) >=
58                abs(crzf(poly[k], poly[i], poly[ii])))
59             k = kk;
60         ret = max(ret, max(dd(poly[i], poly[k]),
```

```
61                dd(poly[ii], poly[k])));
62     }
63     return ret;
64 }
```

8.3 最近點對

```
1  typedef pair<ll, ll> pii;
2  #define x first
3  #define y second
4  ll dd(const pii& a, const pii& b) {
5      ll dx = a.x - b.x, dy = a.y - b.y;
6      return dx * dx + dy * dy;
7  }
8  const ll inf = 1e18;
9  ll dac(vector<pii>& p, int l, int r) {
10     if (l >= r) return inf;
11     int m = (l + r) / 2;
12     ll d = min(dac(p, l, m), dac(p, m + 1, r));
13     vector<pii> t;
14     for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
15         t.push_back(p[i]);
16     for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
17         t.push_back(p[i]);
18     sort(t.begin(), t.end(),
19         [](pii& a, pii& b) { return a.y < b.y; });
20     int n = t.size();
21     for (int i = 0; i < n - 1; i++)
22         for (int j = 1; j < 4 && i + j < n; j++)
23             // 這裡可以知道是哪兩點是最小點對
24             d = min(d, dd(t[i], t[i + j]));
25     return d;
26 }
27 // 給一堆點，求最近點對的距離「的平方」。
28 ll closest_pair(vector<pii>& pp) {
29     sort(pp.begin(), pp.end());
30     return dac(pp, 0, pp.size() - 1);
31 }
```

8.4 最小覆蓋圓

```
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 }
```

8.5 Rectangle Union Area

```
1  const int maxn = 1e5 + 10;
2  struct rec{
3      int t, b, l, r;
4  } r[maxn];
5  int n, cnt[maxn << 2];
6  long long st[maxn << 2], ans = 0;
7  vector<int> x, y;
8  vector<pair<pair<int, int>, pair<int, int>>> v;
9  void modify(int t, int l, int r, int ql, int qr, int v) {
10     if (ql <= l && r <= qr) cnt[t] += v;
11     else {
12         int m = (l + r) >> 1;
13         if (qr <= m) modify(t << 1, l, m, ql, qr, v);
14         else if (ql >= m) modify(t << 1 | 1, m, r, ql, qr, v)
                ;
15         else modify(t << 1, l, m, ql, m, v), modify(t << 1 |
                1, m, r, m, qr, v);
16     }
17     if (cnt[t]) st[t] = y[r] - y[l];
18     else if (r - l == 1) st[t] = 0;
19     else st[t] = st[t << 1] + st[t << 1 | 1];
20 }
21 int main() {
22     cin >> n;
23     for (int i = 0; i < n; i++) {
24         cin >> r[i].l >> r[i].r >> r[i].b >> r[i].t;
25         if (r[i].l > r[i].r) swap(r[i].l, r[i].r);
26         if (r[i].b > r[i].t) swap(r[i].b, r[i].t);
27         x.push_back(r[i].l);
28         x.push_back(r[i].r);
29         y.push_back(r[i].b);
30         y.push_back(r[i].t);
31     }
32     sort(x.begin(), x.end());
33     sort(y.begin(), y.end());
34     x.erase(unique(x.begin(), x.end()), x.end());
35     y.erase(unique(y.begin(), y.end()), y.end());
36     for (int i = 0; i < n; i++) {
37         r[i].l = lower_bound(x.begin(), x.end(), r[i].l) - x.
                begin();
38         r[i].r = lower_bound(x.begin(), x.end(), r[i].r) - x.
                begin();
39         r[i].b = lower_bound(y.begin(), y.end(), r[i].b) - y.
                begin();
40         r[i].t = lower_bound(y.begin(), y.end(), r[i].t) - y.
                begin();
41         v.emplace_back(make_pair(r[i].l, 1), make_pair(r[i].b
                , r[i].t));
42         v.emplace_back(make_pair(r[i].r, -1), make_pair(r[i].
                b, r[i].t));
43     }
```

```
44     sort(v.begin(), v.end(), [](pair<pair<int, int>, pair<int
            , int>> a, pair<pair<int, int>, pair<int, int>> b){
45         if (a.first.first != b.first.first) return a.first.
                first < b.first.first;
46         return a.first.second > b.first.second;
47     });
48     for (int i = 0; i < v.size(); i++) {
49         if (i) ans += (x[v[i].first.first] - x[v[i - 1].first
                .first]) * st[1];
50         modify(1, 0, y.size(), v[i].second.first, v[i].second
                .second, v[i].first.second);
51     }
52     cout << ans << '\n';
53     return 0;
54 }
```

9 Other

9.1 Fastio

```
1  inline ll read(){
2      ll x=0,f=0;
3      char ch = getchar();
4      if(ch==EOF)
5          return 0;
6      while(ch<'0'||ch>'9')f|=ch=='-',ch=getchar();
7      while(ch>='0'&&ch<='9')x=(x<<3)+(x<<1)+(ch^48),ch=getchar
            ();
8      return f?-x:x;
9  }
10
11 inline void print(ll x,bool bk = false) {
12     if(x<0){
13         putchar('-');
14         x = -x;
15     }
16     if(x==0){
17         if(!bk)putchar('0');
18         return;
19     }
20     print(x/10,true);
21     putchar((x-10*(x/10))^'0');
22 }
```

9.2 pbds

```
1  #include<bits/extc++.h>
2  using namespace __gnu_pbds;
3
4  // hash_table：用法和map差不多 //均攤O(1)
5  gp_hash_table <string,int> mp;
6  mp.find(); mp[]=;
7  mp.insert(make_pair())
8
9  // heaps
10 priority_queue<int, greater<int>, TAG> Q;
11 /*
```

Tag	push	pop	join	modify
pairing_heap_tag	O(1)	O(lgN)	O(1)	O(lgN)
thin_heap_tag	O(1)	O(lgN)	慢	慢
binomial_heap_tag	O(1)	O(lgN)	O(lgN)	O(lgN)
rc_binomial_heap_tag	O(1)	O(lgN)	O(lgN)	O(lgN)
binary_heap_tag	O(1)	O(lgN)	慢	O(lgN)

```
18 */    //可以用迭代器遍歷
19 Q.push(x); Q.pop(); Q.top();
20 Q.join(b); //merge two heap
21 Q.empty(); Q.size();
22 Q.modify(it, 6); Q.erase(it);
23
24 // k-th
25 typedef tree<int,null_type,less<int>,rb_tree_tag,
26         tree_order_statistics_node_update> set_t;
27 set_t s; s.insert(12); s.insert(505);
28 assert(*s.find_by_order(0) == 12);
29 assert(*s.find_by_order(3) == 505);
30 assert(s.order_of_key(12) == 0);
31 assert(s.order_of_key(505) == 1);
32 s.erase(12);
33 assert(*s.find_by_order(0) == 505);
34 assert(s.order_of_key(505) == 0);
```

9.3 BuiltIn

```
1  //gcc專用
2  //unsigned int ffs
3  //unsigned long ffsl
4  //unsigned long long ffsll
5  unsigned int x; scanf("%u",&x);
6  printf("右起第一個1:的位置");
7  printf("%d\n",__builtin_ffs(x));
8  printf("左起第一個1之前0的個數:");
9  printf("%d\n",__builtin_clz(x));
10 printf("右起第一個1之後0的個數:");
11 printf("%d\n",__builtin_ctz(x));
12 printf("1的個數:");
13 printf("%d\n",__builtin_popcount(x));
14 printf("1的個數的奇偶性:");
15 printf("%d\n",__builtin_parity(x));
```

9.4 莫隊算法-區間眾數

```
1  using namespace std;
2  const int maxn = 1e6 + 10;
3  struct query { int id, bk, l, r; };
4  int arr[maxn], cnt[maxn], d[maxn], n, m, bk, mx;
5  pair<int,int> ans[maxn];
6  vector<query> q;
7  bool cmp(query x,query y) {
8      return (x.bk < y.bk || (x.bk == y.bk) && x.r < y.r);
9  }
10 void add(int pos) {
11     d[cnt[arr[pos]]]--;
12     cnt[arr[pos]]++;
13     d[cnt[arr[pos]]]++;
14     if(d[mx + 1] > 0) mx++;
```

```
15 }
16 void del(int pos) {
17     d[cnt[arr[pos]]]--;
18     cnt[arr[pos]]--;
19     d[cnt[arr[pos]]]++;
20     if(d[mx] == 0) mx--;
21 }
22 void mo(int n, int m) {
23     sort(q.begin(), q.end(), cmp);
24     for(int i = 0, cl = 1, cr = 0; i < m; i++) {
25         while(cr < q[i].r) add(++cr);
26         while(cl > q[i].l) add(--cl);
27         while(cr > q[i].r) del(cr--);
28         while(cl < q[i].l) del(cl++);
29         ans[q[i].id] = make_pair(mx, d[mx]);
30     }
31 }
32 int main(){
33     cin >> n >> m;
34     bk = (int)sqrt(n + 0.5);
35     for(int i = 1; i <= n; i++) cin >> arr[i];
36     q.resize(m);
37     for(int i = 0; i < m; i++) {
38         cin >> q[i].l >> q[i].r;
39         q[i].id = i,q[i].bk = (q[i].l - 1) / bk;
40     }
41     mo(n, m);
42     for(int i = 0; i < m; i++)
43         cout << ans[i].first << ' ' << ans[i].second << '\n';
44     return 0;
45 }
```

9.5 整體二分搜概念

```
1 void do_things(int l, int r) {
2     for (int i = l; i <= r; i++)
3         // do something
4 }
5
6 void split(vector<int> &qrys, vector<int> &ok, vector<int> &
     fail) {// 決定 qrys 中的元素該去哪邊
7     // 檢查
8     vector<int>().swap(qrys); // 釋放記憶體
9 }
10
11 void undo(int l, int r) { // 取消 l, r 的操作
12     // undo something
13 }
14
15 void total_BS(int l, int r, vector<Query> &qrys) {
16     if (l == r) // 整個 qrys 的答案 = l
17     int mid = (l + r) / 2;
18     do_things(l, mid); // 做所有 <= mid 時會做的事
19
20     /* 原本二分搜時,我們會做:
21     if (check) r = mid;
22     else l = mid + 1;
23     */
24
25     vector<int> lft = qrys 裡需要壓右界的 (r = mid,答案在左
        邊)
```

```
26     vector<int> rgt = qrys 裡需要壓左屆的 (l = mid + 1,答案
        在右邊)
27
28     total_BS(mid + 1, r, rgt);
29     undo_things(l, mid)
30     total_BS(l, mid, lft)
31 }
```

9.6 CNF

```
1 #define MAXN 55
2 struct CNF{
3     int s,x,y;//s->xy | s->x, if y==-1
4     int cost;
5     CNF(){}
6     CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
7 };
8 int state;//規則數量
9 map<char,int> rule;//每個字元對應到的規則,小寫字母為終端字符
10 vector<CNF> cnf;
11 void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 void add_to_cnf(char s,const string &p,int cost){
17     //加入一個s -> <p>的文法,代價為cost
18     if(rule.find(s)==rule.end())rule[s]=state++;
19     for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
22     }else{
23         int left=rule[s];
24         int sz=p.size();
25         for(int i=0;i<sz-2;++i){
26             cnf.push_back(CNF(left,rule[p[i]],state,0));
27             left=state++;
28         }
29         cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost))
            ;
30     }
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限
        小的情形
34 void relax(int l,int r,const CNF &c,long long cost,bool neg_c
     =0){
35     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][
        c.s])){
36         if(neg_c||neg_INF[l][r][c.x]){
37             dp[l][r][c.s]=0;
38             neg_INF[l][r][c.s]=true;
39         }else dp[l][r][c.s]=cost;
40     }
41 }
42 void bellman(int l,int r,int n){
43     for(int k=1;k<=state;++k)
44         for(auto c:cnf)
45             if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
46 }
47 void cyk(const vector<int> &tok){
48     for(int i=0;i<(int)tok.size();++i){
```

```
49         for(int j=0;j<(int)tok.size();++j){
50             dp[i][j]=vector<long long>(state+1,INT_MAX);
51             neg_INF[i][j]=vector<bool>(state+1,false);
52         }
53         dp[i][i][tok[i]]=0;
54         bellman(i,i,tok.size());
55     }
56     for(int r=1;r<(int)tok.size();++r){
57         for(int l=r-1;l>=0;--l){
58             for(int k=l;k<r;++k)
59                 for(auto c:cnf)
60                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c
                        .cost);
61             bellman(l,r,tok.size());
62         }
63     }
64 }
```

9.7 提醒事項

```
1 Debug List:
2 1. Long Long !!
3 2. python3 整數除法 "//"
4 3. connected / unconnected
5 4. 範圍看清楚
6 5. eps 夠小嗎!!
7 6. 可多生 case 測
8 7. 找不用胖資結的其他作法 e.g. multiset -> 單調對列
9 8. 離散化
10 9. 鴿籠原理
11 10. TLE 後找人多想
12
13 ------------------------------------------------
14 Lucas's Theorem
15     For non-negative integer n,m and prime P,
16     C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
17     = mult_i ( C(m_i,n_i) )
18     where m_i is the i-th digit of m in base P.
19 ------------------------------------------------
20 Kirchhoff's theorem
21     A_{ii} = deg(i), A_{ij} = (i,j) \in E ? -1 : 0
22     Deleting any one row, one column, and cal the det(A)
23 ------------------------------------------------
24 Nth Catalan recursive function:
25 C_0 = 1, C_{n+1} = C_n * 2(2n + 1)/(n+2)
26 ------------------------------------------------
27 Mobius Formula
28 u(n) = 1          , if n = 1
29        (-1)^m     , 若 n 無平方數因數,且 n = p1*p2*p3*...*pk
30        0          , 若 n 有大於 1 的平方數因數
31 - Property
32 1. (積性函數) u(a)u(b) = u(ab)
33 2. ∑_{d|n} u(d) = [n == 1]
34 ------------------------------------------------
35 Mobius Inversion Formula
36 if     f(n) = ∑_{d|n} g(d)
37 then   g(n) = ∑_{d|n} u(n/d)f(d)
38              = ∑_{d|n} u(d)f(n/d)
39 - Application
40 the number/power of gcd(i, j) = k
41 - Trick
```

```
42 分塊，O(sqrt(n))
43 ----------------------------------------------------------
44 Chinese Remainder Theorem (m_i 兩兩互質)
45   x = a_1 (mod m_1)
46   x = a_2 (mod m_2)
47   ....
48   x = a_i (mod m_i)
49 construct a solution:
50   Let M = m_1 * m_2 * m_3 * ... * m_n
51   Let M_i = M / m_i
52   t_i = 1 / M_i
53   t_i * M_i = 1 (mod m_i)
54   solution x = a_1 * t_1 * M_1 + a_2 * t_2 * M_2 + ... + a_n
           * t_n * M_n + k * M
55   = k*M + ∑ a_i * t_i * M_i, k is positive integer.
56   under mod M, there is one solution x = ∑ a_i * t_i * M_i
57 ----------------------------------------------------------
58 Burnside's lemma
59 |G| * |X/G|  = sum( |X^g| ) where g in G
60 總方法數：每一種旋轉下不動點的個數總和  除以  旋轉的方法數
61 ----------------------------------------------------------
62 Linear Algebra
63 trace： tr(A) = 對角線和
64 eigen vector： Ax = cx  =>  (A-cI)x = 0
65 ----------------------------------------------------------
66 Josephus Problem
67 f(n,k) = (f(n-1,k)+k)(mod n)
68 f(1,k) = 0
```

9.8　霍夫曼樹

NTHU-ElephantGang Codebook

Contents