Standard Code Library

Moyi

East Northern University

September 9102

Contents

一切的开始

宏定义

- qwq 7

```cpp
#include<bits/stdc++.h>
using namespace std;

#define EPS (1e-10)
#define int long long
//#define lson (rt<<1)
//#define rson (rt<<1|1)
//#define mid ((l+r)>>1)
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcodeforce++)
inline int read(){
    int x = 0, f = 1, c = getchar();
    while(!isdigit(c)) {if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)) {x=(x<<1)+(x<<3)+(c^48);c=getchar();}
    return f==1?x:-x;
}
const int maxn = 2e5 +7;
const int maxm = 2e5 +7;
const int inf = 0x3f3f3f3f;
const int mod = 1e9 +7;

int n;

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

对拍

```cpp
#include<bits/stdc++.h>
using namespace std;
int n;
signed main() {
    while(1) {
        system("make.exe");
        system("std.exe");
        system("mine.exe");
        if(system("fc std.out mine.out")) {
            printf("Wrong Answer!\n");
            int qwq; cin>>qwq;
            return 0;
        }
    }
    return (0);
}
```

- make

```cpp
signed main() {
    #ifdef moyi_qwq
        freopen("in.txt","w",stdout);
    #endif

    srand((int)time(0));
    int maxlen = 5;
    int maxnum = 5 - 1;
    int round = 2;
```

```cpp
        cout<<1<<" "<<20000<<endl;
        for(int i = 1; i <= 20000; i++) {
            cout<<1<<" "<<1<<endl;
        }

        //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
        return (0);
}
```

- std 和 mine.cpp 要加这个

```cpp
freopen("in.txt","r",stdin);
freopen("mine.out","w",stdout);
```

整行输入

- getline

```cpp
char s[];
int len;
cin.getline(s, len);
```

- gets

```cpp
char s[];
gets(s);
```

int128

```cpp
inline __int128 read(){
    __int128 x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9'){
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9'){
        x = x*10 + ch - '0';
        ch = getchar();
    }
    return x * f;
}
inline void print(__int128 x){
    if (x < 0){
        putchar('-');
        x = -x;
    }
    if (x > 9) print(x / 10);
    putchar(x % 10 + '0');
}
```

- 只有基本的加减乘除

```cpp
__int128 a = read();
__int128 b = read();
print(a+b);
```

数据结构

线段树

```cpp
#define lson (rt<<1)
```

4

```
#define rson (rt<<1|1)
#define mid ((l+r)>>1)
```

区间加 & 区间求和

```
1   ll a[maxn];
2   struct SegTree{
3       ll val, add;
4   }tree[maxn << 2];
5
6   void pushdown(int l, int r, int o) {
7
8       tree[lson].val += (mid - l + 1) * tree[o].add;
9       tree[rson].val += (r - mid) * tree[o].add;
10      tree[lson].add += tree[o].add;
11      tree[rson].add += tree[o].add;
12      tree[o].add = 0;
13
14      return ;
15  }
16
17  void buildtree(int l, int r, int o) {
18      tree[o].add = 0;//多组输入
19      if(l == r)
20      {
21          tree[o].val = a[l];
22          return;
23      }
24      buildtree(l , mid, lson);
25      buildtree(mid + 1, r, rson);
26      tree[o].val = tree[lson].val + tree[rson].val;
27      return;
28  }
29
30  ll query(int ql, int qr, int l, int r, int o) {
31      if(l > qr || r < ql) return 0;
32      if(ql <= l && qr >= r) return tree[o].val;
33      pushdown(l, r, o);
34      return query(ql, qr, l, mid, lson) + query(ql, qr, mid + 1, r, rson);
35  }
36
37  void update(int ql, int qr, int l, int r, int o, int addval) { //区间加
38      if(l > qr || r < ql) return;
39      if(ql <= l && qr >= r) {
40          tree[o].val += addval * (r - l + 1);
41          tree[o].add += addval;
42          return;
43      }
44      pushdown(l, r, o);
45
46      update(ql, qr, l, mid, lson, addval);
47      update(ql, qr, mid + 1, r, rson, addval);
48      tree[o].val = tree[lson].val + tree[rson].val;
49      return;
50  }
```

```
buildtree(1, n, 1);
update(x, y, 1, n, 1, k); //区间加
cout<<query(x, y, 1, n, 1)<<endl; //区间和
```

区间加区间乘 & 区间求和 (带取模)

```
1   int n, m, md;
2   int a[maxn];
3   struct node{
4       int val;
5       int add, mult;
6   }tree[maxn<<2];
7
8   void buildtree(int l, int r, int rt) {
```

```
9        if(l == r) {
10           tree[rt].val = a[l];
11           tree[rt].add = 0;
12           tree[rt].mult = 1;
13           tree[rt].val %= md;
14           return ;
15       }
16
17       buildtree(l, mid, lson);
18       buildtree(mid+1, r, rson);
19       tree[rt].add = 0;
20       tree[rt].mult = 1;
21       tree[rt].val = tree[lson].val + tree[rson].val;
22       tree[rt].val %= md;
23       return ;
24   }
25   //先乘再加
26   void pushdown(int l, int r, int rt) {
27       tree[lson].val = (tree[lson].val * tree[rt].mult + tree[rt].add * (mid-l+1)) % md;
28       tree[lson].add = (tree[lson].add * tree[rt].mult + tree[rt].add) % md;
29       tree[lson].mult = (tree[lson].mult * tree[rt].mult) % md;
30
31       tree[rson].val = (tree[rson].val * tree[rt].mult + tree[rt].add * (r-mid)) % md;
32       tree[rson].add = (tree[rson].add * tree[rt].mult + tree[rt].add) % md;
33       tree[rson].mult = (tree[rson].mult * tree[rt].mult) % md;
34
35       tree[rt].add = 0;
36       tree[rt].mult = 1;
37       return ;
38   }
39
40   int query(int l, int r, int rt, int L, int R) {
41       if(L > r || R < l) return 0;
42       if(l >= L && r <= R) {return tree[rt].val;}
43
44       pushdown(l, r, rt);
45       return query(l, mid, lson, L, R) + query(mid+1, r, rson, L, R);
46   }
47
48   void add(int l, int r, int rt, int L, int R, int val) {
49       if(L > r || R < l) return ;
50       if(l >= L && r <= R) {
51           tree[rt].add += val;
52           tree[rt].val += val * (r-l+1);
53           tree[rt].val %= md;
54           return ;
55       }
56       pushdown(l,r,rt);
57       add(l, mid, lson, L, R, val);
58       add(mid+1, r, rson, L, R, val);
59       tree[rt].val = tree[lson].val + tree[rson].val;
60           tree[rt].val %= md;
61       return ;
62   }
63
64   void mult(int l, int r, int rt, int L, int R, int val) {
65       if(L > r || R < l) return ;
66       if(l >= L && r <= R) {
67           pushdown(l, r, rt);
68           tree[rt].mult *= val;
69           tree[rt].val *= val;
70           tree[rt].val %= md;
71           return ;
72       }
73
74       pushdown(l, r, rt);
75       mult(l, mid, lson, L, R, val);
76       mult(mid+1, r, rson, L, R, val);
77       tree[rt].val = tree[lson].val + tree[rson].val;
78           tree[rt].val %= md;
79   }
```

```
buildtree(1, n, 1);
mult(1, n, 1, x, y, k);
add(1, n, 1, x, y, k);
cout<<query(1, n, 1, x, y) % md<<endl;
```

广义线段树

```
1   int a[maxn];
2   struct node {
3       int val;
4       /* 所需的属性 */
5   }tree[maxn<<2];
6   node Merge(node a, node b) {
7       node res;
8       /* 合并两个 node*/
9       return res;
10  }
11  void update(int rt) { //更新 rt
12      tree[rt] = Merge(tree[lson], tree[rson]);
13  }
14  void build(int l, int r,int rt) {
15      if(l == r){
16          /* 初始化 */
17          return ;
18      }
19      build(l, mid, lson);
20      build(mid+1, r, rson);
21      update(rt);
22  }
23
24  void modify(int l, int r, int rt, int x, int y) { //单点修改
25      if(l == r) {
26          /* 所需的修改 */
27          tree[rt].s = tree[rt].t = y;
28          return ;
29      }
30      if(x <= mid) modify(l, mid, lson, x, y);
31      else modify(mid+1, r, rson, x, y);
32      update(rt);
33  }
34  node query(int l, int r, int rt, int L, int R) { //区间查询
35      if(L <= l && r <= R) return tree[rt]; /* 包含进去了 */
36      node tmp;
37      tmp.len = -1;
38      if(L <= mid) tmp = query(l ,mid, lson, L, R); //左边
39      if(mid < R) { //右边 合并
40          if(tmp.len == -1) tmp = query(mid+1, r, rson, L, R);
41          else tmp = Merge(tmp, query(mid+1, r, rson, L, R));
42      }
43      return tmp;
44  }
45
46  signed main() {
47      int n, q;
48      cin>>n>>q;
49      for(int i = 1; i <= n; i++) cin>>a[i];
50      build(1,n,1);
51      int f, x, y;
52      for(int i = 1; i <= q; i++) {
53          cin>>f>>x>>y;
54          if(f == 1) modify(1,n,1,x,y);
55          else if(f == 2) cout<<query(1,n,1,x,y).val<<endl;
56      }
57      return (0);
58  }
```

+加法乘法

```
1   int n, m, md;
2   int a[maxn];
3   struct node{
```

```
4        int val;
5        int add, mult;
6    }tree[maxn<<2];
7
8    void buildtree(int l, int r, int rt) {
9        if(l == r) {
10           tree[rt].val = a[l];
11           tree[rt].add = 0;
12           tree[rt].mult = 1;
13           tree[rt].val %= md;
14           return ;
15       }
16
17       buildtree(l, mid, lson);
18       buildtree(mid+1, r, rson);
19       tree[rt].add = 0;
20       tree[rt].mult = 1;
21       tree[rt].val = tree[lson].val + tree[rson].val;
22       tree[rt].val %= md;
23       return ;
24   }
25   //先乘再加
26   void pushdown(int l, int r, int rt) {
27       tree[lson].val = (tree[lson].val * tree[rt].mult + tree[rt].add * (mid-l+1)) % md;
28       tree[lson].add = (tree[lson].add * tree[rt].mult + tree[rt].add) % md;
29       tree[lson].mult = (tree[lson].mult * tree[rt].mult) % md;
30
31       tree[rson].val = (tree[rson].val * tree[rt].mult + tree[rt].add * (r-mid)) % md;
32       tree[rson].add = (tree[rson].add * tree[rt].mult + tree[rt].add) % md;
33       tree[rson].mult = (tree[rson].mult * tree[rt].mult) % md;
34
35       tree[rt].add = 0;
36       tree[rt].mult = 1;
37       return ;
38   }
39
40   int query(int l, int r, int rt, int L, int R) {
41       if(L > r || R < l) return 0;
42       if(l >= L && r <= R) {return tree[rt].val;}
43
44       pushdown(l, r, rt);
45       return query(l, mid, lson, L, R) + query(mid+1, r, rson, L, R);
46   }
47
48   void add(int l, int r, int rt, int L, int R, int val) {
49       if(L > r || R < l) return ;
50       if(l >= L && r <= R) {
51           tree[rt].add += val;
52           tree[rt].val += val * (r-l+1);
53           tree[rt].val %= md;
54           return ;
55       }
56       pushdown(l,r,rt);
57       add(l, mid, lson, L, R, val);
58       add(mid+1, r, rson, L, R, val);
59       tree[rt].val = tree[lson].val + tree[rson].val;
60           tree[rt].val %= md;
61       return ;
62   }
63
64   void mult(int l, int r, int rt, int L, int R, int val) {
65       if(L > r || R < l) return ;
66       if(l >= L && r <= R) {
67           pushdown(l, r, rt);
68           tree[rt].mult *= val;
69           tree[rt].val *= val;
70           tree[rt].val %= md;
71           return ;
72       }
73
74       pushdown(l, r, rt);
```

```
75        mult(l, mid, lson, L, R, val);
76        mult(mid+1, r, rson, L, R, val);
77        tree[rt].val = tree[lson].val + tree[rson].val;
78            tree[rt].val %= md;
79    }
```

树状数组

- 注意: 0 是无效下标要 1 开始
- lowbit 是最低的 1 的数比如 10001100 就是 100

单点修改 & 区间查询

```
1   int tree[maxn];
2   int n;
3   int lowbit(int x) {return x & -x;}
4   void add(int pos, int val) { //在 pos 位置加 val
5       while(pos <= n)
6       {
7           tree[pos] += val;
8           pos += lowbit(pos);
9       }
10  }
11  int sum(int pos)  { // [1-pos] 的和
12      int ans = 0;
13      while(pos)
14      {
15          ans += tree[pos];
16          pos -= lowbit(pos);
17      }
18      return ans;
19  }
20  int query(int l, int r) { // [l, r] 的和
21      return sum(r) - sum(l-1);
22  }
```

```
add(i,x);
cout<<query(l, r)<<endl;
```

区间修改 & 单点查询

```
int tree[maxn];
int n, m;
int lowbit(int x) { return x & -x; }
int getsum(int x) { // 输出pos的值
    int ans = 0;
    while (x) {
        ans += tree[x];
        x -= lowbit(x);
    }
    return ans;
}
void add(int pos, int val) {
    while (pos <= n) {
        tree[pos] += val;
        pos += lowbit(pos);
    }
    return;
}
void modify(int l, int r, int val) { //[l,r]加上val
    add(l, val);
    add(r + 1, -val);
}
```

```
modify(x, y, k);
cout<<getsum(x)<<endl;
```

区间修改 & 区间查询

其中 n 是数组的最大长度，q 是无用变量

```cpp
1   int tree1[maxn], tree2[maxn];
2   int n, q;
3   int lowbit(int x) {return x & -x;}
4   void add(int pos, int val) {
5       int addval = val * pos;
6       while(pos <= n) {
7           tree1[pos] += val;
8           tree2[pos] += addval;
9           pos += lowbit(pos);
10      }
11  }
12  int sum1(int pos) {
13      // cout<<"sum1:"<<pos<<"=";
14      int ans = 0;
15      while(pos) {
16          ans += tree1[pos];
17          pos -= lowbit(pos);
18      }
19      // cout<<ans<<"\n";
20      return ans;
21  }
22  int sum2(int pos) {
23      // cout<<"sum2:"<<pos<<"=";
24      int ans = 0;
25      while(pos) {
26          ans += tree2[pos];
27          pos -= lowbit(pos);
28      }
29      // cout<<ans<<endl;
30      return ans;
31  }
32  int sum(int pos) {return sum1(pos) * (pos + 1) - sum2(pos);}
33  void modify(int l, int r, int val) { // [l,r] 加上 val
34      add(l ,val);
35      add(r+1, -val);
36  }
37  int query(int l, int r) { // [l,r] 的区间和
38      return sum(r) - sum(l-1);
39  }
```

```
modify(l,r,v);
cout<<query(l,r)<<endl;
```

二维单点修改 & 区间查询

```
int tree[maxn][maxn];
int xn, yy;
int lowbit(int x) {
    return x & -x;
}
void add(int x, int y, int val) { // (x,y)单点+val
    int my = y;
    while (x <= xn) {
        y = my;
        while (y <= yy) {
            tree[x][y] += val;
            y += lowbit(y);
        }
        x += lowbit(x);
    }
```

```
}
int getsum(int x, int y) {
    int ans = 0;
    int my = y;
    while (x) {
        y = my;
        while (y) {
            ans += tree[x][y];
            y -= lowbit(y);
        }
        x -= lowbit(x);
    }
    return ans;
}
int q_get(int x1, int y1, int x2, int y2) { // [x1,x2][y1,y2]的区间和
    int ans = 0;
    ans += getsum(x2, y2);
    ans -= getsum(x1 - 1, y2);
    ans -= getsum(x2, y1 - 1);
    ans += getsum(x1 - 1, y1 - 1);
    return ans;
}

add(x, y, val);
cout<<q_get(a, b, c, d)<<endl;
```

二维区间修改 & 单点查询

```
int n, m;
int a[maxn][maxn];
int tree[maxn][maxn]; //b[i][j] = a[i][j] + a[i-1][j-1] - a[i][j-1] - a[i-1][j]
int lowbit(int x) {return x&-x;}
int geta(int x, int y) { //(x,y)位置的值
    int ans = 0;
    int memy = y;
    while(x) {
        y = memy;
        while(y) {
            ans += tree[x][y];
            y -= lowbit(y);
        }
        x -= lowbit(x);
    }
    return ans;
}
void modify(int x, int y, int val) {
    int memoy = y;
    while(x <= n) {
        y = memoy;
        while(y <= m) {
            tree[x][y] += val;
            y += lowbit(y);
        }
        x += lowbit(x);
    }
}
void add(int xx1, int yy1, int xx2, int yy2, int val) { //区间[xx1,xx2]加val
```

```
        modify(xx1, yy1, val);
        modify(xx1, yy2 + 1, -val);
        modify(xx2 + 1, yy1, -val);
        modify(xx2 + 1, yy2 + 1, val);
    }

add(a,b,c,d,k);
cout<<geta(x,y)<<endl;
```

二维区间修改 & 区间查询

```
int n, m;
int a[maxn][maxn];
int t1[maxn][maxn], t2[maxn][maxn], t3[maxn][maxn], t4[maxn][maxn];
//bij bij*j bij*i bij *i*j
int lowbit(int x) {return x & -x;}
void add(int x, int y, int val) {
    // cout<<x<<" "<<y<<" "<<val<<endl;
    int memoy = y, memox = x;
    while(x <= n) {
        y = memoy;
        while(y <= m) {
            t1[x][y] += val;
            t2[x][y] += val * memoy;
            t3[x][y] += val * memox;
            t4[x][y] += val * memox * memoy;
            y += lowbit(y);
        }
        x += lowbit(x);
    }
}
int ask(int x, int y) {
    int ans = 0;
    int memoy = y, memox = x;
    while(x) {
        y = memoy;
        while(y) {
            ans += (memoy+1)*(memox+1)*t1[x][y];
            ans -= t2[x][y] * (memox + 1);
            ans -= t3[x][y] * (memoy + 1);
            ans += t4[x][y];
            y -= lowbit(y);
        }
        x -= lowbit(x);
    }
    return ans;
}
void range_add(int xx1, int yy1, int xx2, int yy2, int val) { //区间加
    add(xx1,yy1, val);
    add(xx1, yy2 + 1, -val);
    add(xx2 + 1, yy1, -val);
    add(xx2+1,yy2+1,val);
}
int range_ask(int xx1, int yy1, int xx2, int yy2) { //区间和
    int ans = 0;
    ans += ask(xx1-1,yy1-1);
    ans -= ask(xx1-1,yy2);
```

```
        ans -= ask(xx2, yy1-1);
        ans += ask(xx2, yy2);
        return ans;
}

range_add(a,b,c,d,x);
cout<<range_ask(a,b,c,d)<<endl;
```

并查集

并查集

```
1   int n;
2   int fa[maxn];
3   void init() {for(int i = 0; i <= n; i++) fa[i]=i;}
4   int find(int x) {return fa[x]==x?x:fa[x]=find(fa[x]);} //寻找 x 的祖先
5   void merge(int x, int y) { //合并 x y
6       int a = find(x), b = find(y);
7       fa[a] = b;
8   }
```

带权并查集

```
1   int n, m;
2   int fa[maxn],d[maxn];
3   void init() {for(int i = 0; i <= n; i++) fa[i]=i;}
4   int find(int x) { //寻找 x 的祖先
5       if(fa[x] == x) return x;
6       else {
7           int oldFa = fa[x];
8           fa[x] = find(oldFa);
9           d[x] = d[x] + d[oldFa];
10          return fa[x];
11      }
12  }
13  void merge(int x,int y,int w) { //合并 x y
14      int fax = find(x),fay = find(y);
15      if(fax == fay) return;
16      fa[fax] = fay;
17      d[fax] = -d[x] + d[y] + w;
18  }
19  int dist(int x,int y) { //x y 的距离 (必须在一个集里)
20      int fax = find(x),fay = find(y);
21      if(fax != fay) return -1;
22      else return d[x] - d[y];
23  }
```

```
merge(x,y,z);
if(find(x)!=find(y)) cout<<"?"<<endl;
else cout<<dist(x,y)<<endl;
```

堆

```
class HEAP{
    public:
    int minheap[maxn];//1 index
    int heap_cnt = 0;
    void swap(int i, int j) {
        int t = minheap[i];
        minheap[i] = minheap[j];
        minheap[j] = t;
    }
    void push(int x) {
        heap_cnt++;
        minheap[heap_cnt] = x;
```

```cpp
        for(int i = heap_cnt, j = i >> 1; ; ) {
            if(j == 0) break; //to the most top
            if(minheap[i] < minheap[j]) {
                swap(i,j);
            }
            i = j;
            j = i >> 1;
        }
        return ;
    }
    void pop() {
        minheap[1] = minheap[heap_cnt];
        heap_cnt--;
        for(int i = 1, j = i << 1; ; ){
            if(j > heap_cnt) break; // to the bottom
            if(j < heap_cnt && minheap[j+1] < minheap[j]) j++;// find the smaller one of sons
            if(minheap[j] > minheap[i]) break;// right position
            swap(i,j);
            i = j;
            j = i << 1;
        }
        return ;
    }
    int top() {
        return minheap[1];
    }
};

h.push(x);
cout<<h.top()<<endl;
h.pop();
```

平衡树

Treap

```cpp
struct node{
    int key, priority;
    node *left, *right;
};
typedef node* Node;

Node _delete(Node &t, int key);

Node rt = NIL;

Node rightRotate(Node &t) {
    Node s = t->left;
    t->left = s->right;
    s->right = t;
    return s; // the new root of subtree
}
Node leftRotate(Node &t) {
    Node s = t->right;
    t->right = s->left;
    s->left = t;
    return s; // the new root of subtree
}
```

```
Node insert(Node &t, int key, int priority){          // search the corresponding place recursively
    if (t == NIL) {
        t = new node;
        Node &newnd = t;
        newnd->key = key;
        newnd->priority = priority;                    // create a new node when you reach a leaf
        newnd->left = NIL;
        newnd->right = NIL;
        return newnd;
    }
    if (key == t->key) {
        return t;                                      // ignore duplicated keys
    }


    if(key < t->key){                                  // move to the left child
        t->left = insert(t->left, key, priority);     // update the pointer to the left child
       if(t->priority < t->left->priority)            // rotate right if the left child has higher priority
            t = rightRotate(t);
    }
    else {                                             // move to the right child
        t->right = insert(t->right, key, priority);   // update the pointer to the right child
       if(t->priority < t->right->priority) {         // rotate left if the right child has higher priority
            t = leftRotate(t);
        }
    }
    return t;
}
Node delete1(Node &t, int key) {                       // seach the target recursively
    if(t == NIL)
        return NIL;
    if(key < t->key) {                                 // search the target recursively
        t->left = delete1(t->left, key);
    }
    else if (key > t->key)
        t->right = delete1(t->right, key);
    else
        return _delete(t, key);
    return t;
}
Node _delete(Node &t, int key) {                       // if t is the target node
    if(t->left == NIL && t->right == NIL)              // if t is a leaf
        return NIL;
   else if (t->left == NIL)               // if t has only the right child, then perform left rotate
        t = leftRotate(t);
   else if (t->right == NIL)              // if t has only the left child, then perform right rotate
        t = rightRotate(t);
    else {                                             // if t has both the left and right child
        if (t->left->priority > t->right->priority)  // pull up the child with higher priority
            t = rightRotate(t);
        else
            t = leftRotate(t);
    }
    return delete1(t, key);
}
void Print(Node rt, int f) {
```

```cpp
    if(rt == NIL) return ;
    if(f == 1) cout<<" "<<rt->key;
    Print(rt->left,f);
    if(f == 2) cout<<" "<<rt->key;
    Print(rt->right,f);
}
bool Find(Node rt, int x) {
    if(rt == NIL) return false;
    if(x == rt->key) return true;
    else if(x < rt->key) return Find(rt->left,x);
    else return Find(rt->right, x);
    return false;
}

int n;

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif

    cin>>n;
    string s;
    for(int i = 1; i <= n; i++) {
        cin>>s;
        if(s == "insert") {
            int k, p;
            cin>>k>>p;
            insert(rt,k,p);
        }
        else if(s == "print") {
            Print(rt,2); cout<<endl;
            Print(rt,1); cout<<endl;
        }
        else if(s == "find") {
            int x; cin>>x;
            if(Find(rt, x)) cout<<"yes"<<endl;
            else cout<<"no"<<endl;
        }
        else if(s == "delete") {
            int x; cin>>x;
            delete1(rt,x);
        }
    }


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

Splay

```cpp
inline ll read()
{
    rll x=0;
    bool fg=false;
```

```cpp
        char ch=getchar();
        while(ch<'0'||ch>'9')
        {
            if(ch=='-') fg=true;
            ch=getchar();
        }
        while(ch>='0'&&ch<='9')
        {
            x=(x<<3)+(x<<1)+(ch^48);
            ch=getchar();
        }
        return fg?~x+1:x;
}
const int N=1e5+5;
int n,tot,root;
int ch[N][2],fa[N];//左孩子,右孩子,父亲
ll val[N],siz[N],cnt[N];//点值
void pushup(int id) { //更新siz
    siz[id]=siz[ch[id][0]]+siz[ch[id][1]]+cnt[id];
}
void spin(int x) {
    rint y=fa[x],z=fa[y],d=(ch[y][1]==x);//d 判断x是y的左孩子还是右孩子
    ch[z][ch[z][1]==y]=x,fa[x]=z;//处理x与z的关系
    ch[y][d]=ch[x][d^1],fa[ch[x][d^1]]=y;//处理y的孩子与x的孩子的关系
    ch[x][d^1]=y;fa[y]=x;//处理y与x的关系
    pushup(y);//先更新y
    pushup(x);//在更新x
}
void splay(int x,int goal) {
    while(fa[x]!=goal)//判断是否已经到目标点的下边
    {
        rint y=fa[x],z=fa[y];
        if(z!=goal)//判断是情况一还是情况二、三
            (ch[y][0]==x)^(ch[z][0]==y)?spin(x):spin(y);
        //判断是情况二还是情况三
        spin(x);
    }
    if(goal==0) root=x;//如果移动到了根节点,则更新根节点
}
void insert(ll x) {
    int u=root,fat=0;
    while(u&&val[u]!=x) { //先向下找
        fat=u;
        u=ch[u][x>val[u]];
    }
    if(u)   cnt[u]++;
    else {
        u=++tot;
        if(fat) ch[fat][x>val[fat]]=u;//如果不是根节点,更新孩子节点
        fa[u]=fat;//插入操作
        val[u]=x;
        siz[u]=1;
        cnt[u]=1;
    }
    splay(u,0);//每次都要伸展,避免成链
}
```

```cpp
void find(ll x) {
    int u=root;
    if(!u)  return;//不存在该节点,直接返回
    while(ch[u][x>val[u]]&&x!=val[u])//找到该节点的位置
        u=ch[u][x>val[u]];
    splay(u,0);//伸展
}
int get(ll x,int d) { //d:0找前驱 1找后继
    find(x);//先伸展
    int u=root;
    if((val[u]>x&&d)||(val[u]<x&&!d))   return u;
    //如果该节点已经符合要求,直接返回位置
    u=ch[u][d];//找到左右子树
    while(ch[u][d^1])   u=ch[u][d^1];
    //找左子树中最大的或右子树中最小的(关键看你找前驱还是后继)
    return u;//返回前驱或后继的位置
}
void del(ll x) {
    int pre=get(x,0),nxt=get(x,1);//找前驱后继
    splay(pre,0),splay(nxt,pre);//伸展
    int id=ch[nxt][0];//要删除的点
    if(cnt[id]>1)//如果这个数值有重复,直接--cnt即可
    {
        --cnt[id];
        splay(id,0);//伸展
    }
    else
    {
        ch[nxt][0]=0,fa[id]=0;//先切断联系
        val[id]=0,cnt[id]=0,siz[id]=0;//再进行删除
        pushup(nxt),pushup(pre);//最后更新siz
    }
}
ll k_th(int k) {
    int x=root;
    if(siz[x]<k)    return 0;//整棵树的大小都比k小,则没有这个数
    while(1)
    {
        int y=ch[x][0];
        if(k>siz[y]+cnt[x])//比左子树的大小和该节点的重复次数大
        {
            k-=(siz[y]+cnt[x]);
            x=ch[x][1];//去右子树中搜
        }
        else
        {
            if(siz[y]>=k)   x=y;//小于等于左子树,去左子树中搜
            else    return val[x];//否则,返回该值
        }
    }
}
signed main() {
    insert(-inf),insert(inf); //////////////////////// 注意 ////////////////////////
    n=read();
    for(rint i=1;i<=n;++i) {
        int op=read();
```

```
            ll x=read();
            switch(op) {
                case 1:
                    insert(x); //插入数x
                    break;
                case 2:
                    del(x); //删除数x（若有多个相同的数，只删除一个）；
                    break;
                case 3:
                    find(x); //查询x的排名（若有多个相同的数，输出最小的排名）；
                    printf("%lld\n",siz[ch[root][0]]);//因为前面还加了一个-INF，所以不用再加1了
                    break;
                case 4:
                    printf("%lld\n",k_th(x+1)); //查询排名为x的数
                    break;
                case 5:
                    printf("%lld\n",val[get(x,0)]); //求x的前驱（前驱定义为小于x，且最大的数）
                    break;
                case 6:
                    printf("%lld\n",val[get(x,1)]); //求x的后继（后继定义为大于x，且最小的数）
                    break;
            }
        }
        return 0;
    }
```

数学

gcd

```
1  ll Gcd(ll a,ll b){return b==0?a:Gcd(b,a%b);}
```

组合数

```
int qpow(int x, int n) {
    if(!n) return 1;
    int t = 1;
    while(n) {
        if(n&1) t *= x;
        n >>= 1;
        x *= x;
        x = (x + mod) % mod;
        t = (t + mod) % mod;
    }
    return t;
}

int jc[maxn], inv_jc[maxn];
int get_inv(int x) {return qpow(x, mod - 2);}
int A(int x, int n) { //get x from n
    if(x>n)
        return 0;
    else
        return (jc[n]*inv_jc[n-x])%mod;
}
int C(int x, int n) { //get x from n
    if(x>n)
```

```
        return 0;
    else if(x==n||x==0)
        return 1;
    else
        return (A(x,n)*inv_jc[x])%mod;
}
```

素数筛

- 线性筛

visit[] 第 i 个数是不是质数,1 表示质数，prime[] 质数集合 2 3 5 7，k 质数数量, 筛到 maxn

```
1   bool visit[maxn]; //visit 存的第 i 个数是不是质数 0 表示质数
2   int k, prime[maxn]; //k 存的质数数量，筛到 maxn,prime 存的质数集合 (0 开始) 2 3 5 7 ....
3
4   void initPrime()//init
5   {
6       visit[0] = visit[1] = 1;
7       for(int i = 2; i < maxn; i++)
8       {
9           if(!visit[i]) prime[k++] = i;
10          for(int j = 0; j < k && i*prime[j]<maxn; j++)//遍历素数数组
11          {
12              visit[i * prime[j]] = 1;
13              if(i % prime[j] == 0) break;
14          }
15      }
16  }
```

- 线性筛 + 欧拉函数

欧拉函数（Euler's totient function），即 $\varphi(n)$，表示的是小于等于 n 和 n 互质的数的个数。存在 phi[i] 里。

```
1   //phi(i):= 小于等于 i 的与 i 互质的数的个数
2   const int p_max = 1e5 + 100;
3   int phi[p_max];
4   void get_phi() {
5       static bool vis[p_max];
6       static int prime[p_max], p_sz, d;
7       vis[0] = vis[1] = 1; phi[1] = 1;
8       for(int i = 2; i < p_max; i++) {
9           if (!vis[i]) {
10              prime[p_sz++] = i;
11              phi[i] = i - 1;
12          }
13          for (int j = 0; j < p_sz && (d = i * prime[j]) < p_max; ++j) {
14              vis[d] = 1;
15              if (i % prime[j] == 0) {
16                  phi[d] = phi[i] * prime[j];
17                  break;
18              }
19              else phi[d] = phi[i] * (prime[j] - 1);
20          }
21      }
22  }
```

扩展欧几里得

- 求 $ax + by = gcd(a,b)$ 的一组解
- 如果 a 和 b 互素，那么 x 是 a 在模 b 下的逆元
- 注意 x 和 y 可能是负数

```
1   //ax+by=gcd(a,b) 是否有解 有解解就是 x
2   bool ex_gcd(int a, int b, int& x, int& y) {
3       if(b == 0) {
4           x = 1;
5           y = 0;
6           return a;
```

```
7          }
8          int d = ex_gcd(b, a%b, x, y);
9          int temp = x;
10         x = y;
11         y = temp - a/b * y;
12         return d;
13     }
14     //同余方程 ax+by=c 即 ax=c(mod b) 是否有解
15     bool CongruenceEquation(int a, int b,int c, int& x, int& y) {
16         int d = ex_gcd(a,b,x,y);
17         if(c%d != 0) return 0;
18         int k = c / d;
19         x *= k;
20         y *= k;
21         return 1;
22     }
```

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时, $f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时, $g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \ge c$ or $b \ge c$ 时, $h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$; 否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

逆元

- 如果 p 不是素数, 使用拓展欧几里得

- 前置模板: 快速幂 / 扩展欧几里得

```
1      int n;
2      ll mod = (int)(1e9+7);
3      inline ll qpow(ll base, ll rk) {
4          ll ans = 1;
5          ll now = (base % mod + mod) % mod;
6          for(; rk; rk >>= 1) {
7              if(rk & 1) {ans *= now; ans %= mod;}
8              now *= now; now %= mod;
9          }
10         return ans;
11     }
12
13     //费马小定理求单个数逆元 mod 必须素数
14     ll get_inv(ll x) {return qpow(x, mod - 2);}
15
16     int inv_n[maxn];
17     //线性求 n 个数的逆元
18     void init_n() {
19         inv_n[1] = 1;
20         for(int i = 2; i <= n; i++) {
21             inv_n[i] = 1ll * (mod - mod / i) * inv_n[mod % i] % mod;
22         }
23     }
24
25     ll a[maxn], inv[maxn], s[maxn], sv[maxn];//原数组 逆元 前缀积 逆元前缀积
26     //线性求任意 n 个数的逆元
27     void init_any() {
28         s[0] = 1;
29         for(int i = 1; i <= n; i++) s[i] = s[i-1] * a[i] % mod;
30         sv[n] = qpow(s[n], mod-2);
31         for(int i = n; i >= 1; i--) sv[i-1] = sv[i] * a[i] % mod;
32         for(int i = 1; i <= n; i++) inv[i] = sv[i] * s[i-1] % mod;
33     }
```

- 预处理阶乘及其逆元

```
1  LL invf[M], fac[M] = {1};
2  void fac_inv_init(LL n, LL p) {
3      FOR (i, 1, n)
4          fac[i] = i * fac[i - 1] % p;
5      invf[n - 1] = bin(fac[n - 1], p - 2, p);
6      FORD (i, n - 2, -1)
7          invf[i] = invf[i + 1] * (i + 1) % p;
8  }
```

快速幂

- 如果模数是素数，则可在函数体内加上 `n %= MOD - 1;`（费马小定理）

```
1  ll FastPowerMod(ll x, ll n)
2  {
3      if(!n) return 1;
4      ll t = 1;
5      while(n)
6      {
7          if(n&1) t *= x;
8          n >>= 1;
9          x *= x;
10         x = (x + mod) % mod;
11         t = (t + mod) % mod;
12     }
13     return t;
14 }
```

质因数分解

- 前置模板：素数筛

- 带指数

```
1  LL factor[30], f_sz, factor_exp[30];
2  void get_factor(LL x) {
3      f_sz = 0;
4      LL t = sqrt(x + 0.5);
5      for (LL i = 0; pr[i] <= t; ++i)
6          if (x % pr[i] == 0) {
7              factor_exp[f_sz] = 0;
8              while (x % pr[i] == 0) {
9                  x /= pr[i];
10                 ++factor_exp[f_sz];
11             }
12             factor[f_sz++] = pr[i];
13         }
14     if (x > 1) {
15         factor_exp[f_sz] = 1;
16         factor[f_sz++] = x;
17     }
18 }
```

多项式

FWT

```
//快速沃尔什变换 位运算卷积
//https://www.luogu.com.cn/problem/P4717
#include<bits/stdc++.h>
using namespace std;

// #define int long long
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcod
typedef long long ll;
```

```cpp
typedef unsigned long long ull;
const ll maxn = 2e5 +7;
const ll maxm = 2e5 +7;
const ll inf = 0x3f3f3f3f;
const ll mod = 998244353;//1e9+7
const ll inv2 = 499122177;
const ll
Cor[2][2] = {{1,0},{1,1}},
Cand[2][2] = {{1,1},{0,1}},
Cxor[2][2] = {{1,1},{1,mod-1}},
ICor[2][2] = {{1,0},{mod-1,1}},
ICand[2][2] = {{1,mod-1},{0,1}},
ICxor[2][2] = {{inv2, inv2},{inv2,mod-inv2}};

void FWT(ll *F, const ll c[2][2], int n) {
    for(int len = 1; len < n; len <<= 1)
        for(int p = 0; p < n; p += len*2)
            for(int i = p; i < p +len; i++) {
                ll sav = F[i];
                F[i] = (c[0][0]*F[i]+c[0][1]*F[i+len])%mod;
                F[i+len] = (c[1][0]*sav+c[1][1]*F[i+len])%mod;
            }
}

void bitmul(ll *F, ll *G, const ll C[2][2], const ll IC[2][2], int n) {
    FWT(F, C, n); FWT(G, C, n);
    for(int i = 0; i < n; i++) F[i] = F[i]*G[i]%mod;
    FWT(F, IC, n);
}

int n;
ll f[maxn], g[maxn], a[maxn], b[maxn];
signed main()
{
    // freopen("D:/c++source file/intxt/in.txt","r",stdin);
    ios :: sync_with_stdio(0);
    cin.tie(0);

    int n; cin>>n; n = (1<<n);
    for(int i = 0; i < n; i++) cin>>f[i];
    for(int i = 0; i < n; i++) cin>>g[i];
    memcpy(a,f,sizeof(ll)*n);memcpy(b,g,sizeof(ll)*n);
    bitmul(a,b, Cor, ICor, n);
    for(int i = 0; i < n; i++) cout<<a[i]<<" "; cout<<endl;

    memcpy(a,f,sizeof(ll)*n);memcpy(b,g,sizeof(ll)*n);
    bitmul(a,b, Cand, ICand, n);
    for(int i = 0; i < n; i++) cout<<a[i]<<" "; cout<<endl;

    memcpy(a,f,sizeof(ll)*n);memcpy(b,g,sizeof(ll)*n);
    bitmul(a,b, Cxor, ICxor, n);
    for(int i = 0; i < n; i++) cout<<a[i]<<" "; cout<<endl;

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

公式

调和级数部分和

$$S = \frac{1}{1} + \frac{1}{2} + \ldots\ldots + \frac{1}{n} \quad S = ln(n) + eulr + \frac{1}{2n} \quad elur = 0.57721\,56649\,01532\,86060$$

一些数论公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p) + \phi(p)} \pmod{p}$
- $\mu^2(n) = \sum_{d^2|n} \mu(d)$
- $\sum_{d|n} \varphi(d) = n$
- $\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2)$, 其中 ω 是不同素因子个数
- $\sum_{d|n} \mu^2(d) = 2^{\omega(d)}$

一些数论函数求和的例子

- $\sum_{i=1}^{n} i[gcd(i,n) = 1] = \frac{n\varphi(n) + [n=1]}{2}$
- $\sum_{i=1}^{n} \sum_{j=1}^{m} [gcd(i,j) = x] = \sum_d \mu(d) \lfloor \frac{n}{dx} \rfloor \lfloor \frac{m}{dx} \rfloor$
- $\sum_{i=1}^{n} \sum_{j=1}^{m} gcd(i,j) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{d|gcd(i,j)} \varphi(d) = \sum_d \varphi(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $S(n) = \sum_{i=1}^{n} \mu(i) = 1 - \sum_{i=1}^{n} \sum_{d|i, d<i} \mu(d) \stackrel{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^{n} S(\lfloor \frac{n}{t} \rfloor)$
  - 利用 $[n=1] = \sum_{d|n} \mu(d)$
- $S(n) = \sum_{i=1}^{n} \varphi(i) = \sum_{i=1}^{n} i - \sum_{i=1}^{n} \sum_{d|i, d<i} \varphi(i) \stackrel{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^{n} S(\frac{n}{t})$
  - 利用 $n = \sum_{d|n} \varphi(d)$
- $\sum_{i=1}^{n} \mu^2(i) = \sum_{i=1}^{n} \sum_{d^2|n} \mu(d) = \sum_{d=1}^{\lfloor \sqrt{n} \rfloor} \mu(d) \lfloor \frac{n}{d^2} \rfloor$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} gcd^2(i,j) = \sum_d d^2 \sum_t \mu(t) \lfloor \frac{n}{dt} \rfloor^2$
  $\stackrel{x=dt}{=} \sum_x \lfloor \frac{n}{x} \rfloor^2 \sum_{d|x} d^2 \mu(\frac{x}{d})$
- $\sum_{i=1}^{n} \varphi(i) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} [i \perp j] - 1 = \frac{1}{2} \sum_{i=1}^{n} \mu(i) \cdot \lfloor \frac{n}{i} \rfloor^2 - 1$

斐波那契数列性质

- $F_{a+b} = F_{a-1} \cdot F_b + F_a \cdot F_{b+1}$
- $F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}, F_2 + F_4 + \cdots + F_{2n} = F_{2n+1} - 1$
- $\sum_{i=1}^{n} F_i = F_{n+2} - 1$
- $\sum_{i=1}^{n} F_i^2 = F_n \cdot F_{n+1}$
- $F_n^2 = (-1)^{n-1} + F_{n-1} \cdot F_{n+1}$
- $gcd(F_a, F_b) = F_{gcd(a,b)}$
- 模 n 周期（皮萨诺周期）
  - $\pi(p^k) = p^{k-1} \pi(p)$
  - $\pi(nm) = lcm(\pi(n), \pi(m)), \forall n \perp m$
  - $\pi(2) = 3, \pi(5) = 20$
  - $\forall p \equiv \pm 1 \pmod{10}, \pi(p)|p-1$
  - $\forall p \equiv \pm 2 \pmod 5, \pi(p)|2p+2$

一些组合公式

- 错排公式: $D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n!(\frac{1}{2!} - \frac{1}{3!} + \cdots + (-1)^n \frac{1}{n!}) = \lfloor \frac{n!}{e} + 0.5 \rfloor$
- 卡塔兰数（n 对括号合法方案数，n 个结点二叉树个数，$n \times n$ 方格中对角线下方的单调路径数，凸 $n+2$ 边形的三角形划分数，n 个元素的合法出栈序列数）: $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

中国剩余定理

- 无解返回 -1
- 前置模板: 扩展欧几里得

```
1   LL CRT(LL *m, LL *r, LL n) {
2       if (!n) return 0;
3       LL M = m[0], R = r[0], x, y, d;
4       FOR (i, 1, n) {
5           d = ex_gcd(M, m[i], x, y);
6           if ((r[i] - R) % d) return -1;
7           x = (r[i] - R) / d * x % (m[i] / d);
8           // 防爆 LL
9           // x = mul((r[i] - R) / d, x, m[i] / d);
10          R += x * M;
11          M = M / d * m[i];
12          R %= M;
13      }
14      return R >= 0 ? R : R + M;
15  }
```

博弈

- Nim 游戏：每轮从若干堆石子中的一堆取走若干颗。先手必胜条件为石子数量异或和非零。
- 阶梯 Nim 游戏：可以选择阶梯上某一堆中的若干颗向下推动一级，直到全部推下去。先手必胜条件是奇数阶梯的异或和非零（对于偶数阶梯的操作可以模仿）。
- Anti-SG：无法操作者胜。先手必胜的条件是：
  - SG 不为 0 且某个单一游戏的 SG 大于 1。
  - SG 为 0 且没有单一游戏的 SG 大于 1。
- Every-SG：对所有单一游戏都要操作。先手必胜的条件是单一游戏中的最大 step 为奇数。
  - 对于终止状态 step 为 0
  - 对于 SG 为 0 的状态，step 是最大后继 step +1
  - 对于 SG 非 0 的状态，step 是最小后继 step +1
- 树上删边：叶子 SG 为 0，非叶子结点为所有子结点的 SG 值加 1 后的异或和。

尝试：

- 打表找规律
- 寻找一类必胜态（如对称局面）
- 直接博弈 dp

图论

最短路

DIJKSTRA

```
1   int n, m, s;
2   vector<pair<int, int>> e[maxn]; // v, w 到达点 长度
3   int dis[maxn]; //距离
4   bool vis[maxn]; //是否访问过
5   void Dijk() {
6       mst(vis);
7       for(int i = 0; i <= n +7; i++) dis[i] = inf;
8       dis[s] = 0;
9       priority_queue<pair<int, int>> q; // u,dis
10      q.push(make_pair(s,0));
11      while(!q.empty()) {
12          int u = q.top().first, d = q.top().second;
13          q.pop();
14          for(auto y : e[u]) {
15              int v = y.first, w = y.second;
16              int td = d + w;
17              if(td < dis[v]) {
18                  dis[v] = td;
19                  q.push(make_pair(v,td));
20              }
21          }
22      }
23  }
```

SPFA+ 判负环

```cpp
int n, m, s;
vector<pair<int, int>> e[maxn]; //to w 边终点，边权
int dis[maxn], cnt[maxn], vis[maxn]; // 0-(n-1)

bool spfa(int s) { // 返回true如果没有负环 否则返回false
    for(int i = 0; i < n; i++) {
        dis[i] = inf;
    }
    queue<int> q;
    dis[s] = 0; vis[s] = 1;
    q.push(s);
    while(!q.empty()) {
        int u = q.front();
        q.pop(); vis[u] = 0;
        for(auto y : e[u]) {
            int v = y.first, w = y.second;
            if(dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                cnt[v] = cnt[u] + 1;  // 记录最短路经过的边数
                if (cnt[v] >= n) return false;
                // 在不经过负环的情况下，最短路至多经过 n - 1 条边
                // 因此如果经过了多于 n 条边，一定说明经过了负环
                if (!vis[v]) q.push(v), vis[v] = 1;
            }
        }
    }
    return true;
}
```

Floyd

初始化

```cpp
int n, m;
int dis[maxn][maxn];

for(int i = 0; i < maxn; i++) { //初始化dis
    for(int j = 0; j < maxn; j++) {
        dis[i][j] = inf;
    }
}
for(int i = 0; i < n; i++) dis[i][i] = 0;//初始化dis
```

输入

```cpp
for(int i = 1; i <= m; i++) {
    int u, v, w;
    cin>>u>>v>>w;
    dis[u][v] = w;
}
```

计算

```cpp
for(int k = 0; k < n; k++) { //先枚举中转点
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
        }
```

```
        }
    }
```

判负环

```
for(int i = 0; i < n; i++) {
    if(dis[i][i] < 0) { // 如果自己到自己是负数则有负环
        cout<<"NEGATIVE CYCLE"<<endl;
        return 0;
    }
}
```

输出

```
for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
        //注意不可达到判断不能直接==inf,要设置一个上界
        if(dis[i][j]>1e10) cout<<"INF";
        else cout<<dis[i][j];
        cout<<(j==n-1?"\n":" ");
    }
}
```

LCA

倍增

输入 N，M，S：树的结点个数、询问的个数和树根结点的序号。

接下来 N-1 行是边，M 行询问最近公共祖先是谁。

```
5 5 4
3 1
2 4
5 1
1 4
2 4
3 2
3 5
1 2
4 5
```

```
1   int N, M, S;
2   int fa[maxn][31], dep[maxn];
3   //fa[i][j]：第 i 个点的第 2^j 个祖先
4   vector<int> G[maxn];
5
6   void dfs(int root,int fno) {
7       fa[root][0] = fno;
8       dep[root] = dep[fno] + 1;
9       //初始化 fa
10      for(int i = 1; i < 31; ++i) {
11          fa[root][i] = fa[fa[root][i-1]][i-1];
12      }
13      //遍历
14      for(auto y : G[root]) {
15          if(y == fno) continue;
16          dfs(y, root);
17      }
18  }
19
20  int lca(int x, int y) {
21      if(dep[x] > dep[y]) swap(x,y);
22      int tem = dep[y] - dep[x];
23      for(int i = 0; tem; ++i, tem >>= 1)
24      {
```

```
25            if(tem&1) y = fa[y][i];
26        }
27        if(y == x) return x;
28        for(int i = 30; i >= 0 && y != x; --i) {
29            if(fa[x][i] != fa[y][i]) {
30                x = fa[x][i];
31                y = fa[y][i];
32            }
33        }
34        return fa[x][0];
35    }
36
37    int main() {
38        cin>>N>>M>>S;
39        for(int i = 1; i < N; i++) {
40            int x, y;
41            cin>>x>>y;
42            G[x].push_back(y);
43            G[y].push_back(x);
44        }
45        dfs(S, 0);
46        for(int i = 1; i <= M; i++) {
47            int x, y;
48            cin>>x>>y;
49            cout<<lca(x,y)<<endl;
50        }
51        return (0);
52    }
```

欧拉路径/回路

判别法:

- 对于无向图 G, G 是欧拉图当且仅当 G 是连通的且没有奇度顶点。(欧拉回路

- 对于无向图 G, G 是半欧拉图当且仅当 G 是连通的且 G 中恰有个或个奇度顶点。(欧拉路径

- 对于有向图 G, G 是欧拉图当且仅当 G 的所有顶点属于同一个强连通分量且每个顶点的入度和出度相同。(欧拉回路

- 对于有向图 G, G 是半欧拉图当且仅当: (欧拉路径

    - 如果将 G 中的所有有向边退化为无向边时, 那么 G 的所有顶点属于同一个连通分量。
    - 最多只有一个顶点的出度与入度差为 1。
    - 最多只有一个顶点的入度与出度差为 1。
    - 所有其他顶点的入度和出度相同。

题:

无向图找欧拉路径,输出字典序最小的路径。

```
1   int n, m;
2   int e[maxn][maxn], du[maxn]; //邻接矩阵 点的度
3   stack<int> ans; //访问点的顺序
4
5   void dfs(int u) {
6       for(int i = 1; i <= n; i++) {
7           if(e[u][i]) {
8               e[u][i]--; e[i][u]--;
9               dfs(i);
10          }
11      }
12      ans.push(u);
13  }
14
15  signed main() {
16      cin>>m;
17      for(int i = 1; i <= m; i++) {
18          int u, v; cin>>u>>v;
19          n = max(n, u); n = max(n, v);
20          e[u][v]++;
21          e[v][u]++;
```

```
22          du[u]++; du[v]++;
23      }
24      int s = 1;
25      for(int i = 1; i <= n; i++) {
26          if(du[i]%2==1) {s = i; break;}
27      }
28      dfs(s);
29      while(ans.size()) {
30          cout<<ans.top()<<endl;
31          ans.pop();
32      }
33      return (0);
34  }
```

强连通分量与 2-SAT

题意:

2*n 个人 m 个关系。每个 2*i-1 和 2*i 之间要选一个。

u 和 v 不能共存。

3 2
1 3
2 4

输出方案或者不存在则 NIE

1
4
5

```
1   //2-SAT
2   //https://www.luogu.com.cn/problem/P5782
3   #include<bits/stdc++.h>
4   using namespace std;
5
6   // #define int long long
7   typedef long long ll;
8   typedef unsigned long long ull;
9   #define mst(a) memset(a,0,sizeof(a))
10  #define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcodeforce++)
11  const ll maxn = 2e5 +7;
12  const ll maxm = 2e5 +7;
13  const ll inf = 0x3f3f3f3f;
14  const ll mod = 100000007;
15
16  int n, m;//n 人 1~n(1&2 同党) m 关系
17  int oth(int x) {return x%2?x+1:x-1;}//另一个人
18  vector<int> e[maxn];//存边
19
20  int dfn[maxn], low[maxn], dfncnt;
21  int tj_stack[maxn], in_stack[maxn], tp;
22  int scc[maxn], scc_cnt;
23  void tarjan(int rt) {
24      dfn[rt] = low[rt] = ++dfncnt;
25      tj_stack[++tp] = rt, in_stack[rt] = 1;
26      for(int i = 0; i < e[rt].size(); i++)
27      {
28          int y = e[rt][i];
29          if(!dfn[y])
30          {
31              tarjan(y);
32              low[rt] = min(low[rt], low[y]);
33          }
34          else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
35      }
36      if(dfn[rt] == low[rt])
37      {
38          ++scc_cnt;
```

```
39          while(tj_stack[tp] != rt)
40          {
41              scc[tj_stack[tp]] = scc_cnt;
42              in_stack[tj_stack[tp]] = 0;
43              tp--;
44          }
45          scc[tj_stack[tp]] = scc_cnt;
46          in_stack[tj_stack[tp]] = 0;
47          tp--;
48      }
49  }
50
51  void init()
52  {
53      mst(dfn);
54      mst(low);
55      dfncnt=0;
56      tp = 0;
57      for(int i = 1; i <= n; i++) e[i].clear();
58      mst(scc);
59      scc_cnt = 0;
60      mst(tj_stack);
61      mst(in_stack);
62  }
63
64  int main()
65  {
66      // freopen("D:/c++source file/intxt/in.txt","r",stdin);
67      ios :: sync_with_stdio(0);
68      cin.tie(0);
69
70      while(cin>>n>>m)
71      {
72          n *= 2;
73          init();
74
75          for(int i = 1; i <= m; i++)
76          {
77              int u, v;
78              cin>>u>>v;  //交叉连边
79              e[u].push_back(oth(v));
80              e[v].push_back(oth(u));
81          }
82          for(int i = 1; i <= n; i++)
83          {
84              if(!dfn[i]) tarjan(i);  //求全部强连通分量
85          }
86          int f = 0;
87          for(int i = 1; i <= n; i+=2)
88          {
89              if(scc[i] == scc[oth(i)])  //不存在合法解
90              {
91                  cout<<"NIE"<<endl;
92                  f = 1;
93                  break;
94              }
95          }
96          if(f) continue;
97          for(int i = 1; i <= n; i+=2) // 输出一个解
98          {
99              cout<<(scc[i] > scc[oth(i)] ? oth(i) : i)<<endl;
100         }
101     }
102     //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
103     return (0);
104     /*
105     先 tarjan 缩点 然后检查行不行 然后直接输出
106     */
107 }
```

拓扑排序

```
1   int n, m;
2   int deg[maxn]; //入度
3   vector<int> e[maxn];
4   vector<int> ans;
5
6   bool toposort() { // 0-indexed
7       queue<int> q;
8       for(int i = 0; i < n; i++) {
9           if(deg[i]==0) q.push(i);
10      }
11      while(!q.empty()) {
12          int u = q.front();
13          q.pop();
14          ans.push_back(u);
15          for(auto y : e[u]) {
16              --deg[y];
17              if(deg[y]==0) q.push(y);
18          }
19      }
20      if(ans.size()==n) return true;
21      else return false;
22  }
```

```
    cin>>n>>m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin>>u>>v;
        e[u].push_back(v);
        deg[v]++;
    }
    if(toposort()) {
        for(auto y : ans) cout<<y<<endl;
    }
    else {
        cout<<"None"<<endl; //题目DAG必能拓扑排序
    }
```

连通性相关

割点和割边（无向图

tarjan 同时处理割点和桥

```
int n, m;
vector<int> e[maxn];
int dfn[maxn], low[maxn], vis[maxn];
//访问顺序 不过fa最低访问dfn 记录
int iscut[maxn];
int isbridge[maxn], father[maxn];

void tarjan(int u, int fa, int dep) { //割点割边同时处理
    father[u] = fa;
    vis[u] = 1;
    int child = 0;
    low[u] = dfn[u] = dep;
    for(auto v : e[u]) {
        if(v != fa && vis[v] == 1) {
            low[u] = min(low[u], dfn[v]);
        }
        if(vis[v] == 0) {
            tarjan(v, u, dep + 1);
            child++;
```

```
            low[u] = min(low[u], low[v]);
            if( (fa==-1 && child>1) || (fa!=-1 && low[v] >= dfn[u])) {
                iscut[u] = true;
            }
            if(low[v] > dfn[u]) {
                isbridge[v] = true; //割边 会重复所以要isbridge[]记录
            }
        }
    }
    vis[u] = 2;
}
```

割点 main 函数:

```
    cin>>n>>m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin>>u>>v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    for(int i = 0; i < n; i++) {
        if(!dfn[i]) {
            tarjan(i,-1,1);
        }
    }
    for(int i = 0; i < n; i++) {
        if(iscut[i])
            cout<<i<<endl;
    }
```

割边 main 函数:

```
    cin>>n>>m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin>>u>>v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    for(int i = 0; i < n; i++) {
        if(!dfn[i]) {
            tarjan(i,-1,1);
        }
    }
    vector<pair<int, int>> ans;
    for(int i = 0; i < n; i++) {
        if(isbridge[i]) {
            if(i<father[i])
                ans.push_back(make_pair(i,father[i]));
            else
                ans.push_back(make_pair(father[i],i));
        }
    }
    sort(ans.begin(), ans.end());
    for(auto y : ans) {
        cout<<y.first<<" "<<y.second<<endl;
    }
```

强连通分量缩点（有向图

```
1   int n, m;
2   vector<int> e[maxn];//存边
3
4   int dfn[maxn], low[maxn], dfncnt;
5   int tj_stack[maxn], in_stack[maxn], tp;
6   int scc[maxn], scc_cnt;
7   void tarjan(int rt) {
8       dfn[rt] = low[rt] = ++dfncnt;
9       tj_stack[++tp] = rt, in_stack[rt] = 1;
10      for(int i = 0; i < e[rt].size(); i++)
11      {
12          int y = e[rt][i];
13          if(!dfn[y])
14          {
15              tarjan(y);
16              low[rt] = min(low[rt], low[y]);
17          }
18          else if(in_stack[y]) low[rt] = min(low[rt], dfn[y]);
19      }
20      if(dfn[rt] == low[rt])
21      {
22          ++scc_cnt;
23          while(tj_stack[tp] != rt)
24          {
25              scc[tj_stack[tp]] = scc_cnt;
26              in_stack[tj_stack[tp]] = 0;
27              tp--;
28          }
29          scc[tj_stack[tp]] = scc_cnt;
30          in_stack[tj_stack[tp]] = 0;
31          tp--;
32      }
33  }
```

```
1   void init()
2   {
3       mst(dfn);
4       mst(low);
5       dfncnt=0;
6       tp = 0;
7       for(int i = 1; i <= n; i++) e[i].clear();
8       mst(scc);
9       scc_cnt = 0;
10      mst(tj_stack);
11      mst(in_stack);
12  }
```

```
//例题  判断是否在同一个强连通分量
    cin>>n>>m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin>>u>>v;
        e[u].push_back(v);
    }
    for(int i = 0; i < n; i++) {
        if(!dfn[i]) tarjan(i);
    }
    int q; cin>>q;
    for(int i = 1; i <= q; i++) {
        int a, b; cin>>a>>b;
        cout<<(scc[a]==scc[b])<<endl;
    }
```

最小生成树

```
1   //唯一
2   struct edge{
```

```cpp
    int from, to, val;
}e[maxn];
int n, m, ans;
int fa[maxn];
bool cmp(edge a, edge b){return a.val < b.val;}
void init()
{
    for(int i = 0; i <= n; i++)
    {
        fa[i] = i;
    }
    ans = 0;
}
int find(int x) {return fa[x]==x?x:(fa[x] = find(fa[x]));}
bool uniKruskal()
{
    int sum1 = 0, sum2 = 0;//已使用的 可能使用的
    int p = 0;//相同指针
    int flag = 0, num = 0;
    for(int i = 1; i <= m + 1; i++)
    {
        if(p<i)
        {
            if(sum1 != sum2)
            {
                flag = 1;
                break;
            }
            sum1 = 0, sum2 = 0;
            for(int j = i; j <= m+1; j++)
            {
                if(e[j].val != e[i].val)
                {
                    p = j-1;
                    break;
                }
                if(find(e[j].from) != find(e[j].to))
                    ++sum2;
            }
        }
        if(i>m) break;
        int x = find(e[i].from);
        int y = find(e[i].to);
        if(x != y && num != n-1)
        {
            num++;
            sum1++;
            // merge(x, y);
            fa[x] = fa[y];
            ans += e[i].val;
        }
    }
    if(flag) return false;
    else return true;
}

int main()
{
    freopen("D:/c++source file/intxt/in.txt","r",stdin);
    // ios :: sync_with_stdio(0);
    // cin.tie(0);
    cf
    {
        cin>>n>>m;
        init();
        for(int i = 1; i <= m; i++) cin>>e[i].from>>e[i].to>>e[i].val;
        sort(e+1,e+1+m, cmp);
        if(uniKruskal()) cout<<ans<<endl;
        else cout<<"Not Unique!"<<endl;
    }
```

```
75      //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
76      return (0);
77  }
```

网络流

dinic 最大流

```
struct node{
    int to, nxt;
    LL val;
}e[maxm<<1];
int n, m; //在memset分层的深度用到了n,m没用到
int head[maxn];
int cnt, S, T; //cnt=0 head初始化-1为了i^1是对边
inline void AddEdge(int u, int v, int w) {
    e[cnt].to = v;
    e[cnt].val = w;
    e[cnt].nxt = head[u];
    head[u] = cnt++;
    //对边
    e[cnt].to = u;
    e[cnt].val = 0;
    e[cnt].nxt = head[v];
    head[v] = cnt++;
}
int dep[maxn], q[maxn], l, r;

bool BFS() { //图分层
    memset(dep, 0, (n + 1) * sizeof(int)); //要先初始化 且n+1
    q[l = r = 1] = S;
    dep[S] = 1; //0是空置 是不访问标志数
    while(l <= r) {
        int u = q[l++];
        for(int p = head[u]; ~p; p = e[p].nxt) {
            int v = e[p].to;
            if(e[p].val && !dep[v]) {
                dep[v] = dep[u] + 1;
                q[++r] = v;
            }
        }
    }
    return dep[T]; // 0就是搜不到
}
LL DFS(int u, LL in /*搜索点，入流支持*/) { // 一个分层图的阻塞流
    if(u == T)
        return in;
    LL out = 0;
    //                  这个 and 剪了十几倍
    for(int p = head[u]; ~p && in; p = e[p].nxt) {
        int v = e[p].to;
        LL w = e[p].val;
        if(w && dep[v] == dep[u]+1) {
            LL res = DFS(v, min(in, w));
            e[p].val -= res;
            e[p^1].val +=  res;
```

```
            in -= res;
            out += res;
        }
    }
    if(out == 0) //强力的剪枝
        dep[u] = 0; //不搜不能到汇点的点
    return out;
}

LL maxflow() {
    cnt = 1;
    LL ans = 0;
    while(BFS())
        ans += DFS(S, infLL);
    return ans;
}



signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif
    IOS;

    read(n); read(m); read(S); read(T);
    memset(head, -1, (n+1) * sizeof(int));
    for(int i = 1; i <= m; i++) {
        int u, v, w;
        read(u); read(v); read(w);
        AddEdge(u, v, w);
    }
    cout<<maxflow()<<endl;

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

dinic+spfa 最小费用最大流

```
int head[maxn], cnt;
int ret = 0;
struct edge{
    int to, nxt, w, cost;
}e[maxm];
void add(int u, int v, int w, int c) {
    e[cnt].to = v;
    e[cnt].w = w;
    e[cnt].cost = c;
    e[cnt].nxt = head[u];
    head[u] = cnt++;
}
void addEdge(int u, int v, int w, int c) {
    add(u, v, w, c);
    add(v, u, 0, -c);
}
```

```cpp
int n, m;
int vis[maxn], dis[maxn], cur[maxn]; // cur是head的复制 加速dfs
//////////////////////

bool spfa(int s, int t) {
    memset(dis, 0x3f, sizeof(dis));
    memcpy(cur, head, sizeof(head));
    queue<int> q;
    q.push(s); dis[s] = 0; vis[s] = 1;
    while(!q.empty()) {
        int u = q.front();
        q.pop(); vis[u] = 0;
        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].to;
            if(e[i].w && dis[v] > dis[u] + e[i].cost) {
                dis[v] = dis[u] + e[i].cost;
                if(!vis[v])
                    q.push(v), vis[v] = 1;
            }
        }
    }
    return dis[t] != inf;
}

int dfs(int u, int t, int flow) {
    if(u == t) return flow;
    vis[u] = 1;
    int ans = 0;
    for(int &i = cur[u]; ~i && ans < flow; i = e[i].nxt) {
        int v = e[i].to;
        if(!vis[v] && e[i].w && dis[v] == dis[u] + e[i].cost) {
            int x = dfs(v, t, min(e[i].w, flow - ans));
            if(x) {
                ret += x * e[i].cost;
                e[i].w -= x;
                e[i^1].w += x;
                ans += x;
            }
        }
    }
    vis[u] = 0;
    return ans;
}

int mcmf(int s, int t) {
    int res = 0;
    while(spfa(s, t)) {
        int x;
        while((x = dfs(s, t, infLL))) {
            res += x;
        }
    }
    return res;
}

signed main() {
```

```
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif
    IOS;

    int s, t;
    cin>>n>>m>>s>>t;
    memset(head, -1, (n+1) * sizeof(int));
    while(m--) {
        int u, v, w, c;
        cin>>u>>v>>w>>c;
        addEdge(u,v,w,c);
    }
    int x = mcmf(s, t);
    printf("%d %d\n", x, ret);

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

EK+spfa 最小费用最大流

```
int n, m, S, T;
int head[maxn], cnt;
int incf[maxn], dis[maxn], pre[maxn];
bool vis[maxn];
//incf是最大入流 pre存来的边号
struct edge{
    int to, nxt, fl, cost;
}e[maxm];
void add(int u, int v, int c, int w) {
    e[cnt].to = v;
    e[cnt].fl = c;
    e[cnt].cost = w;
    e[cnt].nxt = head[u];
    head[u] = cnt++;
}
void addE(int u, int v, int c, int w) {
    add(u, v, c, w);
    add(v, u, 0, -w);
}

bool spfa() {
    queue<int> q;
    memset(dis, 0x3f, sizeof(dis));
    memset(incf, 0, sizeof(incf));
    q.push(S); dis[S] = 0; incf[S] = inf;
    while(!q.empty()) {
        int u = q.front();
        q.pop();
        vis[u] = false;

        for(int i = head[u]; ~i; i = e[i].nxt) {
            int v = e[i].to;
            if(e[i].fl && dis[v] > dis[u] + e[i].cost) {
                dis[v] = dis[u] + e[i].cost;
                pre[v] = i;
```

```cpp
                incf[v] = min(e[i].fl, incf[u]);
                if(!vis[v]) {
                    vis[v] = true;
                    q.push(v);
                }
            }
        }
    }
    return incf[T] > 0;
}

void EK(int &flow, int &cost) {
    flow = cost = 0;
    while(spfa()) {
        int t = incf[T];
        flow += t; cost += t * dis[T];
        for(int i = T; i != S; i = e[pre[i]^1].to) {
            e[pre[i]].fl -= t;
            e[pre[i]^1].fl += t;
        }
    }
}

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif
    IOS;

    cin>>n>>m>>S>>T;
    memset(head, -1, sizeof(head));
    for(int i = 1; i <= m; i++) {
        int u, v, c, w;
        cin>>u>>v>>c>>w;
        addE(u, v, c, w);
    }
    int flow, cost;
    EK(flow, cost);
    cout<<flow<<" "<<cost<<endl;

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

杂项

前向星存边

```cpp
int n;
int head[maxn], cnt = 0;
struct edge{
    int to;
    int w;
    int next;
}e[maxn<<2];

void add(int u, int v, int w) {
```

```
    e[++cnt].to = v;
    e[cnt].w = w;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void iterate_edge(int u) { //遍历u的点
    for(int i = head[u]; i; i=e[i].next) {


    }
}
```

计算几何

几角排序

```
struct point{
    int x, y;
    double k;
    void calk() {k = atan(1.0*y/x);}
    bool operator < (const point b) const {
        if(x == 0) {
            if(b.x > 0) {return true;}
            else if(b.x == 0) {return ( (b.y<0) > (y<0) );}
            else if(b.x < 0) {return y > 0;}
        }
        else if(b.x == 0) {
            if(x > 0) {return false;}
            else if(x == 0) {return ( (b.y<0) > (y<0) );}
            else if(x < 0) {return b.y < 0;}
        }
        else { //  都不是0
            if(x*b.x<0) return x < b.x; //两边
            //else return y / x < b.y / b.x;
            else return y * b.x < b.y * x;
            //else return k<b.k;
        }
        return 1;
    }
} a[maxn];
```

基本

```
#include<bits/stdc++.h>
using namespace std;

#define EPS (1e-10)
#define int long long
//#define lson (rt<<1)
//#define rson ((rt<<1)+1)
//#define mid ((l+r)>>1)
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcod
typedef long long ll;
const ll maxn = 2e5 +7;
const ll maxm = 2e5 +7;
```

```cpp
const ll inf = 0x3f3f3f3f;
const ll mod = 1e9 +7;
/////////////////////////

#define EPS (1e-10)
#define equals(a, b) (fabs((a)-(b)) < EPS)

//点和向量
class Point{
public:
    double x, y;

    Point(double x = 0, double y = 0) : x(x), y(y) {}

    Point operator + (Point p) {return Point(x+p.x,y+p.y);}
    Point operator - (Point p) {return Point(x-p.x,y-p.y);}
    Point operator * (double a) {return Point(a*x,a*y);}
    Point operator / (double a) {return Point(x/a,y/a);}

    double abs() {return sqrt(norm());}
    double norm() {return x * x + y * y;}

    bool operator < (const Point &p) const {
        return x != p.x ? x < p.x : y < p.y;
    }

    bool operator == (const Point &p) const {
        return fabs(x - p.x) < EPS && fabs(y - p.y) < EPS;
    }
    void ShowPoint() {cout<<x<<" "<<y<<endl;}
};
typedef Point Vector;

// 线段和线
struct Segment{
    Point p1, p2;
};
typedef Segment Line;

//圆和多边形
class Circle {
public:
    Point c;
    double r;
    Circle(Point c = Point(), double r = 0.0) : c(c), r(r){}
};

typedef vector<Point> Polygon;

//函数

double dot(Vector a, Vector b) { //点乘
    return a.x * b.x + a.y * b.y;
}
double cross(Vector a, Vector b) {
    return a.x*b.y - a.y*b.x;
```

```
}


/////////////////////////
int n;

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif




    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

平面最近点对（分治）

```
//https://onlinejudge.u-aizu.ac.jp/courses/library/4/CGL/5/CGL_5_A
//平面最近点对 分治
#include<bits/stdc++.h>
using namespace std;

#define EPS (1e-10)
#define int long long
//#define lson (rt<<1)
//#define rson ((rt<<1)+1)
//#define mid ((l+r)>>1)
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcod
typedef long long ll;
const ll maxn = 2e5 +7;
const ll maxm = 2e5 +7;
const ll inf = 0x3f3f3f3f;
const ll mod = 1e9 +7;

int n;

struct point{
    double x, y;
    int id;
};

bool cmp_x(point a, point b) {
    if(a.x==b.x) return a.y < b.y;
    else return a.x<b.x;}
bool cmp_y(point a, point b) {
    if(a.y==b.y) return a.x<b.x;
    else return a.y<b.y;}

vector<point> a;
double mindist = 1e20;
int ansa, ansb;
void upd_ans(const point &a, const point &b) {
    double dist = sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y) + 0.0);
```

```cpp
        if(dist < mindist) {
            mindist = dist; ansa = a.id, ansb = b.id;
        }
    }
}
void csp(int l, int r) { //求[l, r]的最近点对
    if(r - l <= 3) {
        for(int i = l; i <= r; i++) {
            for(int j = i + 1; j <= r; j++) upd_ans(a[i], a[j]);
        }
        sort(a.begin()+l,a.begin()+r+1,cmp_y);
        return ;
    }

    int m = (r+l) / 2;
    double midx = a[m].x;
    csp(l,m); csp(m+1,r);
    inplace_merge(a.begin()+l,a.begin()+m+1,a.begin()+r+1, cmp_y);

    static point t[maxn];
    int tsz = 0;
    for (int i = l; i <= r; ++i) {
        if (abs(a[i].x - midx) < mindist) {
            for (int j = tsz - 1; j >= 0 && a[i].y - t[j].y < mindist; --j) {
                upd_ans(a[i], t[j]);
            }
            t[tsz++] = a[i];
        }
    }
}

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif

    cin>>n;
    for(int i = 0; i < n; i++) {
        point p; cin>>p.x>>p.y; p.id = i; a.push_back(p);
    }
    sort(a.begin(),a.begin()+n,cmp_x);
    csp(0,n-1);
    printf("%.10lf\n",mindist);

    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

全板子

```cpp
#include<bits/stdc++.h>
using namespace std;

//#define lson (rt<<1)
//#define rson ((rt<<1)+1)
//#define mid ((l+r)>>1)
#define mst(a) memset(a,0,sizeof(a))
#define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcod
```

```cpp
typedef long long ll;
const ll maxn = 2e5 +7;
const ll maxm = 2e5 +7;
const ll inf = 0x3f3f3f3f;
const ll mod = 1e9 +7;
//////////////////////////////////////////////////////////
// COUNTER CLOCKWISE
static const int CCW_COUNTER_CLOCKWISE = 1;
static const int CCW_CLOCKWISE = -1;
static const int CCW_ONLINE_BACK = 2;
static const int CCW_ONLINE_FRONT = -2;
static const int CCW_ON_SEGMENT = 0;

//Intercsect Circle & Circle
static const int ICC_SEPERATE = 4; // 不相交
static const int ICC_CIRCUMSCRIBE = 3; // 外接
static const int ICC_INTERSECT = 2; //相交
static const int ICC_INSCRIBE = 1; //内切
static const int ICC_CONTAIN = 0; //包含

#define EPS (1e-10)
#define equals(a, b) (fabs((a)-(b)) < EPS)

//点和向量
class Point{
public:
    double x, y;

    Point(double x = 0, double y = 0) : x(x), y(y) {}

    Point operator + (Point p) {return Point(x+p.x,y+p.y);}
    Point operator - (Point p) {return Point(x-p.x,y-p.y);}
    Point operator * (double a) {return Point(a*x,a*y);}
    Point operator / (double a) {return Point(x/a,y/a);}

    double abs() {return sqrt(norm());}
    double norm() {return x * x + y * y;}

    bool operator < (const Point &p) const {
        return x != p.x ? x < p.x : y < p.y;
    }

    bool operator == (const Point &p) const {
        return fabs(x - p.x) < EPS && fabs(y - p.y) < EPS;
    }
    void ShowPoint() {cout<<x<<" "<<y<<endl;}
};
typedef Point Vector;

double norm(Vector a) {
    return a.x * a.x + a.y * a.y;
}
double abs(Vector a) {
    return sqrt(norm(a));
}
// 线段和线
```

```cpp
struct Segment{
    Point p1, p2;
    Segment() {}
    Segment(Point a, Point b): p1(a), p2(b) {}
};
typedef Segment Line;

//圆和多边形
class Circle {
public:
    Point c;
    double r;
    Circle(Point c = Point(), double r = 0.0) : c(c), r(r){}
};

typedef vector<Point> Polygon;

///////////////函数///////////////////////////////////////////

double dot(Vector a, Vector b) { //点乘
    return a.x * b.x + a.y * b.y;
}

double cross(Vector a, Vector b) {//叉乘
    return a.x*b.y - a.y*b.x;
}

Point project(Segment s, Point p) { // 投影
    Vector base = s.p2 - s.p1;
    double r = dot(p-s.p1, base) / base.norm();
    return s.p1 + base * r;
}

Point reflect(Segment s, Point p) { // 对称点
    return p + (project(s,p)-p) * 2.0;
}

double arg(Vector p) {return atan2(p.y, p.x);} // 角度
Vector polar(double a, double r) {return Point(cos(r) * a, sin(r) * a);} // 极坐标转直角坐标

int ccw(Point p0, Point p1, Point p2) { // p0-p1和p0-p2的方向 顺逆时针 大小
    Vector a = p1 - p0;
    Vector b = p2 - p0;
    //a.Show(); b.Show(); cout<<"|||"<<endl;
    if(cross(a, b) > EPS) {return CCW_COUNTER_CLOCKWISE;} // COUNTER_CLOCKWISE
    if(cross(a, b) < -EPS) {return CCW_CLOCKWISE;} // CLOCKWISE
    if(dot(a, b) < -EPS) {return CCW_ONLINE_BACK;} // ONLINE_BACK
    if(a.norm() < b.norm()) {return CCW_ONLINE_FRONT;} // ONLINE_FRONT
    if(a.norm() > b.norm()) {return CCW_ON_SEGMENT;} // ON_SEGMENT

    return 0;
}
bool convexPolygon(Polygon p) { //凸多边形
    int n = p.size();
    for(int i = 0; i < n; i++) {
        Vector v1, v2;
```

```
            v1 = p[(i+1)%n] - p[i];
            v2 = p[(i+2)%n] - p[(i+1)%p.size()];
            if(cross(v1,v2) < 0) return false;
        }
        return true;
}
bool intersect(Point p1, Point p2, Point p3, Point p4) { //p1-p2 p3-p4相交
        return (ccw(p1, p2, p3) * ccw(p1, p2, p4) <= 0 &&
                ccw(p3, p4, p1) * ccw(p3, p4, p2) <= 0);
}
bool intersect(Segment s1, Segment s2) { // 线段相交
        return intersect(s1.p1, s1.p2, s2.p1, s2.p2);
}
Point getCrossPoint(Segment s1, Segment s2) { // 线段交点
        Vector base = s2.p2 - s2.p1;
        double d1 = abs(cross(base, s1.p1 - s2.p1));
        double d2 = abs(cross(base, s1.p2 - s2.p1));
        double t = d1 / (d1 + d2);
        return s1.p1 + (s1.p2 - s1.p1) * t;
}
pair<Point, Point> getCrossPoints(Circle c, Line l) { //线和圆交点
        //assert(intersect(c, l));
        Vector pr = project(l,c.c);
        Vector e = (l.p2 - l.p1) / abs(l.p2 - l.p1);
        double base = sqrt(c.r * c.r - norm(pr - c.c));
        return make_pair(pr + e * base, pr - e * base);
}
pair<Point, Point> getCrossPoints(Circle c1, Circle c2) { // 圆和圆交点
        double d = abs(c1.c - c2.c);
        double cosv = (c1.r*c1.r+d*d-c2.r*c2.r) / (2*c1.r*d);
        if( abs(abs(cosv) - 1) < EPS ) cosv = 1.0 * (cosv < 0 ? -1 : 1);
        double a = acos(cosv);
        double t = arg(c2.c - c1.c);
        return make_pair(c1.c + polar(c1.r, t+a), c1.c + polar(c1.r, t-a));
}
Point getCrossPointLL(Line l1,Line l2){ //直线交点
  double a=cross(l1.p2-l1.p1,l2.p2-l2.p1);
  double b=cross(l1.p2-l1.p1,l1.p2-l2.p1);
  if(abs(a)<EPS&&abs(b)<EPS) return l2.p1; //共线
  return l2.p1+(l2.p2-l2.p1)*(b/a);
}
int contains(Polygon g, Point p) { // 多边形包含点 IN-2 ON-1 OUT-0
        int n = g.size();
        bool x = false;
        for(int i = 0; i < n; i++) {
            Point a = g[i] - p, b = g[(i+1)%n] - p;
            if( abs(cross(a, b)) < EPS && dot(a, b) < EPS) return 1;
            if( a.y > b.y ) swap(a, b);
            if( a.y < EPS && EPS < b.y && cross(a, b) > EPS) x = !x;
        }
        return (x ? 2 : 0);
}
Polygon andrewScan(Polygon s) { //凸包
        Polygon u, l;
        if(s.size() < 3) return s;
        sort(s.begin(), s.end());
```

```cpp
    // x最小的加到u
    u.push_back(s[0]);
    u.push_back(s[1]);
    // x最大的加到l
    l.push_back(s[s.size()-1]);
    l.push_back(s[s.size()-2]);

    //上部
    for(int i = 2; i < int(s.size()); i++) { // 注意ccw规定了是否取边上多的点
        for(int n = u.size(); n >= 2 && ccw(u[n-2], u[n-1], s[i]) == 1; n--) {
            u.pop_back();
        }
        u.push_back(s[i]);
    }
    //下部
    for(int i = s.size() - 3; i >= 0; i--) { // 注意ccw规定了是否取边上多的点
        for(int n = l.size(); n >= 2 && ccw(l[n-2], l[n-1], s[i]) == 1; n--) {
            l.pop_back();
        }
        l.push_back(s[i]);
    }

    // 从左下开始的顺时针生成序列
    reverse(l.begin(), l.end());
    for(int i = u.size() - 2; i >= 1; i--) {l.push_back(u[i]);}
    // 返回逆时针的序列
    return l;
}
Polygon convexCut(Polygon p, Line l) { // 切凸多边形返回左侧
    Polygon q;
    int n = p.size();
    for(int i = 0; i < n; i++) {
        if(ccw(l.p1, l.p2, p[i]) != -1) {
            q.push_back(p[i]);
        }
        if(ccw(l.p1, l.p2, p[i])*ccw(l.p1, l.p2, p[(i+1)%n]) < 0) {
            Line tem; tem.p1 = p[i]; tem.p2 = p[(i+1)%n];
            q.push_back(getCrossPointLL(l, tem));
        }
    }
    return q;
}
/////////////////距离/////////////////////////////////////////////
double getDistance(Point a, Point b) { // 两点距离
    return (a-b).abs();
}
double getDistanceLP(Line l, Point p) { //点和直线距离
    return abs(cross(l.p2-l.p1,p-l.p1) / abs(l.p2-l.p1));
}
double getDistanceSP(Segment s, Point p) { //点和线段距离
    if ( dot(s.p2 - s.p1, p - s.p1) < 0.0 ) return (p - s.p1).abs();
    if( dot(s.p1 - s.p2, p - s.p2) < 0.0 ) return (p - s.p2).abs();
    return getDistanceLP(s, p);
}
double getDistance(Segment s1, Segment s2) { //线段和线段距离
    if( intersect(s1, s2) ) return 0.0;
```

```
    return min(
        min(getDistanceSP(s1, s2.p1), getDistanceSP(s1, s2.p2)),
        min(getDistanceSP(s2, s1.p1), getDistanceSP(s2, s1.p2))
    );
}
double getAreaPolygon(Polygon v) { // 多边形面积 点按逆时针给出
    double res = 0.0;
    int len = v.size();
    for(int i = 0; i < len; i++) {
        res += cross(v[i], v[(i+1)%len]) / 2.0;
    }
    return res;
}
double diameter(Polygon s) { //求给定凸包的直径
    Polygon p = s;
    int n = p.size();
    if(n == 2) return abs(s[0]-s[1]);
    // i最右上点 j最左下点
    int i = 0, j = 0;
    for(int k = 0; k < n; k++) {
        if(p[i] < p[k]) i = k;
        if(!(p[j]<p[k])) j = k;
    }
    // 旋转卡壳
    double res = 0;
    int si = i, sj = j; // 记录起点
    while(i != sj || j != si) {
        res = max(res, abs(p[i] - p[j]));
        if(cross(p[(i+1)%n]-p[i],p[(j+1)%n]-p[j]) < 0.0) { // 旋转
            i = (i+1) % n;
        }
        else {
            j = (j+1) % n;
        }
    }
    return res;
}
/////////////////////////圆/////////////////////////////////////////
int intersectCC(Circle c1, Circle c2) { //圆相交
    if(c1.r<c2.r) swap(c1,c2);
    double d = abs(c1.c-c2.c);
    double r = c1.r+c2.r;
    if(d == r) {return ICC_CIRCUMSCRIBE;}
    if(d > r) {return ICC_SEPERATE;}
    if(d+c2.r==c1.r) {return ICC_INSCRIBE;}
    if(d+c2.r<c1.r) {return ICC_CONTAIN;}

    return ICC_INTERSECT;
}
Circle getIncircle(Point pa, Point pb, Point pc){ //内切圆
  Vector v1 = pb - pa;
  Vector v2 = pc - pa;
  Vector v3 = pc - pb;
  Vector v4 = pa - pb;
  Vector c1 = polar(10000.0, (arg(v1) + arg(v2))/2);
  Vector c2 = polar(10000.0, (arg(v3) + arg(v4))/2);
```

```cpp
    Point x = getCrossPointLL(Segment(pa, pa + c1), Segment(pb, pb + c2));
    double r = getDistanceLP(Segment(pa, pb), x);
    return Circle(x, r);
}
Circle getExcircle(Point pa, Point pb, Point pc){ //外接圆
    Vector v1 = (pb - pa);
    Vector v2 = (pc - pa);
    Vector v1r = Vector(-v1.y, v1.x);
    Vector v2r = Vector(-v2.y, v2.x);
    Point c1 = pa + v1/2;
    Point c2 = pa + v2/2;
    Point x = getCrossPointLL(Segment(c1, c1 + v1r), Segment(c2, c2 + v2r));
    double r = getDistance(pa, x);
    return Circle(x, r);
}
pair<Point, Point> Tangency(Circle c, Point p) { // 求圆点切线的交点 前提是有切点
    double dis = norm(p-c.c) - c.r*c.r;
    double r2 = sqrt (dis);
    Circle c2; c2.c = p; c2.r = r2;
    pair<Point, Point> res =  getCrossPoints(c, c2);
    return res;
}
vector<Point> getContact(Circle C1,Circle C2){ // 求两个圆公切线的交点 在C1上的

    vector<Point> ret;

    double p = C2.c.x-C1.c.x;
    double q = C2.c.y-C1.c.y;

    double A = p*p+q*q;

    double n_1 = (q*C1.r*(C1.r+C2.r)-p*C1.r*sqrt(A-(C1.r+C2.r)*(C1.r+C2.r)))/(A)+C1.c.y;
    double n_2 = (q*C1.r*(C1.r+C2.r)+p*C1.r*sqrt(A-(C1.r+C2.r)*(C1.r+C2.r)))/(A)+C1.c.y;

    double m_1 = (p*C1.r*(C1.r+C2.r)+q*C1.r*sqrt(A-(C1.r+C2.r)*(C1.r+C2.r)))/(A)+C1.c.x;
    double m_2 = (p*C1.r*(C1.r+C2.r)-q*C1.r*sqrt(A-(C1.r+C2.r)*(C1.r+C2.r)))/(A)+C1.c.x;

    if(A-(C1.r+C2.r)*(C1.r+C2.r) >= 0){
        ret.push_back(Point(m_1,n_1));
        ret.push_back(Point(m_2,n_2));
    }

    double n_3 = (q*C1.r*(C1.r-C2.r)-p*C1.r*sqrt(A-(C1.r-C2.r)*(C1.r-C2.r)))/(A)+C1.c.y;
    double n_4 = (q*C1.r*(C1.r-C2.r)+p*C1.r*sqrt(A-(C1.r-C2.r)*(C1.r-C2.r)))/(A)+C1.c.y;

    double m_3 = (p*C1.r*(C1.r-C2.r)+q*C1.r*sqrt(A-(C1.r-C2.r)*(C1.r-C2.r)))/(A)+C1.c.x;
    double m_4 = (p*C1.r*(C1.r-C2.r)-q*C1.r*sqrt(A-(C1.r-C2.r)*(C1.r-C2.r)))/(A)+C1.c.x;

    if(A-(C1.r-C2.r)*(C1.r-C2.r) >= 0){
        ret.push_back(Point(m_3,n_3));
        ret.push_back(Point(m_4,n_4));
    }

    return ret;
}
///////////////////////////////////////////////////////////////////////////
```

```
int n;

signed main() {
    #ifdef moyi_qwq
        freopen("D:/source file/intxt/in.txt","r",stdin);
    #endif


    //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
    return (0);
}
```

字符串

manacher

```
1   namespace Manacher {
2       //记得改 maxn 要开两倍
3       char Manacher_s[maxn];
4       int d[maxn];//变换后的字符串回文半径
5       int maxlen;//最长回文串长度
6       void manacher(char s[], int n) {
7           memset(d,0,sizeof(d));
8           int cnt = 0;
9           Manacher_s[cnt++] = '#';
10          for(int i = 0; i < n; i++) {Manacher_s[cnt++]=s[i];Manacher_s[cnt++]='#';}
11          n = strlen(Manacher_s);
12          int r = 0, p = 0;
13          for(int i = 0; i < n; i++) {
14              if(i < r) d[i] = min(d[2*p-i], r-i);
15              else d[i] = 1;
16              while(i-d[i]>=0&&i+d[i]<n&&Manacher_s[i-d[i]]==Manacher_s[i+d[i]]) d[i]++;
17              if(d[i]+i-1>r) {r=d[i]+i-1;p=i;}
18          }
19          for(int i = 0; i < n; i++) d[i]--;
20          maxlen = 0;
21          for(int i = 0; i < n; i++) {
22              if(i&1) maxlen = max(maxlen, (d[i]>>1)*2+1);
23              else maxlen = max(maxlen, d[i]);
24          }
25      }
26  }
```

哈希

内置了自动双哈希开关（小心 TLE）。

```
1   #include<bits/stdc++.h>
2   using namespace std;
3
4   typedef long long ll;
5   typedef unsigned long long ull;
6   #define mst(a,x) memset(a,x,sizeof(a))
7   #define cf int Tcodeforces, Tcodeforce;cin>>Tcodeforces;for(Tcodeforce = 1; Tcodeforce <= Tcodeforces; Tcodeforce++)
8   const ll maxn = 2e5 +7;
9   const ll inf = 0x3f3f3f3f;
10  const ll mod = 100000007;
11
12  namespace HString
13  {
14      //标号从零开始
15      const int N = 1e3 +7;
16      //字符数组长度
17      const int x = 135;
18      const int p1 = 1e9 + 7, p2 = 1e9 + 9;
19      ull xp1[N], xp2[N], xp[N];
```

```
20
21      void init_xp()
22      {
23          xp1[0] = xp2[0] = xp[0] = 1;
24          for(int i = 1; i < N; i++)
25          {
26              xp1[i] = xp1[i - 1] * x % p1;
27              xp2[i] = xp2[i - 1] * x % p2;
28              xp[i] = xp[i - 1] * x;
29          }
30      }
31
32      struct HashString
33      {
34          char s[N];//文本串
35          int length, subsize;//长度为 length 的子串 子串数组的 size
36          bool sorted;
37          //h[i] 是 i 到尾部的串的哈希值
38          ull h[N], hl[N];//h 是 i 到尾部的哈希值 hl 是子串哈希值们
39          //用 char 数组 t 初始化结构体 返回哈希值
40          ull init(const char *t)
41          {
42              if(xp[0] != 1) init_xp();
43              length = strlen(t);
44              strcpy(s, t);
45              ull res1 = 0, res2 = 0;
46              h[length] = 0;
47              for(int j = length - 1; j >= 0; j--)
48              {
49          #ifdef ENABLE_DOUBLE_HASH
50              res1 = (res1 * x + s[j]) % p1;
51              res2 = (res2 * x + s[j]) % p2;
52              h[j] = (res1 << 32) | res2;
53          #else
54              res1 = res1 * x + s[j];
55              h[j] = res1;
56          #endif
57              }
58              return h[0];
59          }
60
61          //获取子串哈希，左闭右开
62          ull get_substring_hash(int left, int right)  {
63              int len = right - left;
64      #ifdef ENABLE_DOUBLE_HASH
65              unsigned int mask32 = ~(0u);
66              ull left1 = h[left] >> 32, right1 = h[right] >> 32;
67              ull left2 = h[left] & mask32, right2 = h[right] & mask32;
68              return (((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
69                      (((left2 - right2 * xp2[len] % p2 + p2) % p2));
70      #else
71              return h[left] - h[right] * xp[len];
72      #endif
73          }
74          //获得长度为 sublen 的子串 存到 hl
75          void get_all_subs_hash(int sublen) {
76              subsize = length - sublen + 1;
77              for (int i = 0; i < subsize; ++i)
78                  hl[i] = get_substring_hash(i, i + sublen);
79              sorted = 0;
80          }
81          //排序
82          void sort_substring_hash() {
83              sort(hl, hl + subsize);
84              sorted = 1;
85          }
86          //从子串中查找哈希值 key
87          //必须先排序
88          bool match(ull key) const {
89              // if (!sorted) assert (0);
90              if (!subsize) return false;
```

```
91              return binary_search(hl, hl + subsize, key);
92          }
93      };
94  }
95  int main()
96  {
97      //freopen("D:/c++source file/intxt/in.txt","r",stdin);
98
99
100     //cerr<<"Time : "<<1000*((double)clock())/(double)CLOCKS_PER_SEC<<"ms";
101     return (0);
102 }
```

KMP

```
1  namespace Kmp {
2      //下标从 0 开始
3      int nxt[maxn];//前缀函数值
4      int cnt = 0;//可以匹配的点的数量
5      vector<int> res;//可以匹配的点的下标
6      void kmp(int a[], char s[], int n) {
7          int j = a[0] = 0;
8          for(int i = 1; i < n; i++) {
9              while(j && s[i] != s[j]) j = a[j-1];
10             a[i] = j += s[i] == s[j];
11         }
12     }
13     void compare(char s[], char mode[], int la, int lb) {
14         res.clear(); memset(nxt, 0, sizeof(nxt));
15         kmp(nxt, mode, lb);
16         int j = 0;
17         for(int i = 0; i < la; i++) {
18             while(j && s[i] != mode[j]) j = nxt[j-1];
19             if(s[i] == mode[j]) j++;
20             if(j == lb) res.push_back(i-j+1);
21         }
22         cnt = res.size();
23     }
24 };
```

- 前缀函数（每一个前缀的最长 border）

```
1  void get_pi(int a[], char s[], int n) {
2      int j = a[0] = 0;
3      FOR (i, 1, n) {
4          while (j && s[i] != s[j]) j = a[j - 1];
5          a[i] = j += s[i] == s[j];
6      }
7  }
```

- Z 函数（每一个后缀和该字符串的 LCP 长度）

```
1  void get_z(int a[], char s[], int n) {
2      int l = 0, r = 0; a[0] = n;
3      FOR (i, 1, n) {
4          a[i] = i > r ? 0 : min(r - i + 1, a[i - l]);
5          while (i + a[i] < n && s[a[i]] == s[i + a[i]]) ++a[i];
6          if (i + a[i] - 1 > r) { l = i; r = i + a[i] - 1; }
7      }
8  }
```

Trie

```
1  namespace trie {
2      int t[N][26], sz, ed[N];
3      void init() { sz = 2; memset(ed, 0, sizeof ed); }
4      int _new() { memset(t[sz], 0, sizeof t[sz]); return sz++; }
5      void ins(char* s, int p) {
6          int u = 1;
7          FOR (i, 0, strlen(s)) {
8              int c = s[i] - 'a';
```

```
 9              if (!t[u][c]) t[u][c] = _new();
10              u = t[u][c];
11          }
12          ed[u] = p;
13      }
14  }
```

AC 自动机

```
 1  const int N = 1e6 + 100, M = 26;
 2
 3  int mp(char ch) { return ch - 'a'; }
 4
 5  struct ACA {
 6      int ch[N][M], danger[N], fail[N];
 7      int sz;
 8      void init() {
 9          sz = 1;
10          memset(ch[0], 0, sizeof ch[0]);
11          memset(danger, 0, sizeof danger);
12      }
13      void insert(const string &s, int m) {
14          int n = s.size(); int u = 0, c;
15          FOR (i, 0, n) {
16              c = mp(s[i]);
17              if (!ch[u][c]) {
18                  memset(ch[sz], 0, sizeof ch[sz]);
19                  danger[sz] = 0; ch[u][c] = sz++;
20              }
21              u = ch[u][c];
22          }
23          danger[u] |= 1 << m;
24      }
25      void build() {
26          queue<int> Q;
27          fail[0] = 0;
28          for (int c = 0, u; c < M; c++) {
29              u = ch[0][c];
30              if (u) { Q.push(u); fail[u] = 0; }
31          }
32          while (!Q.empty()) {
33              int r = Q.front(); Q.pop();
34              danger[r] |= danger[fail[r]];
35              for (int c = 0, u; c < M; c++) {
36                  u = ch[r][c];
37                  if (!u) {
38                      ch[r][c] = ch[fail[r]][c];
39                      continue;
40                  }
41                  fail[u] = ch[fail[r]][c];
42                  Q.push(u);
43              }
44          }
45      }
46  } ac;
47
48  char s[N];
49
50  int main() {
51      int n; scanf("%d", &n);
52      ac.init();
53      while (n--) {
54          scanf("%s", s);
55          ac.insert(s, 0);
56      }
57      ac.build();
58
59      scanf("%s", s);
60      int u = 0; n = strlen(s);
61      FOR (i, 0, n) {
62          u = ac.ch[u][mp(s[i])];
```

```
63        if (ac.danger[u]) {
64            puts("YES");
65            return 0;
66        }
67    }
68    puts("NO");
69    return 0;
70 }
```

杂项

STL

pair 相关

```
pair<string, int> a;
a = {"qq", 1};
stack<pair<string, int>> s;
s.push({"qqq",1});
```
是可用的。

`vector<pair<int, int>> ans;`
`sort(ans.begin(), ans.end());`的时候会优先按照第一关键字排序

可以检查的 sb 点

读题

- 输入的边界情况
- 题意是否多义/模糊

输出

- inf 输出-1