数论

筛选法素数打表

```
void isPrime()
{
    for (int i = 2; i < N; i++)
        if (!a[i])
            for (int j = i + i; j < N; j += i)
                a[j] = 1;
}
```

组合数

```
//扩展欧几里得求组合数
LL fac[N];
void init()
{
    LL i;
    fac[0]=1;
    for (LL i = 1; i < N; i++)
    fac[i] = fac[i - 1] * i % MOD;
}
LL exgcd(LL a, LL b, LL &x, LL &y) {
    if (!b) {x = 1; y = 0; return a;}
    LL d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}

LL inv(LL a, LL n) {
    LL x, y;
    exgcd(a, n, x, y);
    return (x + n) % n;
}

LL C(LL n, LL m) {
    return fac[n] * inv(fac[m] * fac[n - m] % MOD, MOD) % MOD;
}

//递推求 O(n^2)
```

```
int comb[N][N];//comb[n][m]就是 C(n,m)
void init(){
    for(int i = 0; i < N; i ++){
        comb[i][0] = comb[i][i] = 1;
        for(int j = 1; j < i; j ++){
            comb[i][j] = comb[i-1][j] + comb[i-1][j-1];
            comb[i][j] %= MOD;
        }
    }
}
```

线性基

```
const int MN=60;
ll a[61],tmp[61];
bool flag;
void ins(ll x){
    for(reg int i=MN;~i;i--)
        if(x&(1ll<<i))
            if(!a[i]){a[i]=x;return;}
            else x^=a[i];
    flag=true;
}
bool check(ll x){
    for(reg int i=MN;~i;i--)
        if(x&(1ll<<i))
            if(!a[i])return false;
            else x^=a[i];
    return true;
}
ll qmax(ll res=0){
    for(reg int i=MN;~i;i--)
        res=max(res,res^a[i]);
    return res;
}
ll qmin(){
    if(flag)return 0;
    for(reg int i=0;i<=MN;i++)
        if(a[i])return a[i];
}
ll query(ll k){
    reg ll res=0;reg int cnt=0;
    k-=flag;if(!k)return 0;
    for(reg int i=0;i<=MN;i++){
        for(int j=i-1;~j;j--)
            if(a[i]&(1ll<<j))a[i]^=a[j];
        if(a[i])tmp[cnt++]=a[i];
    }
    if(k>=(1ll<<cnt))return -1;
    for(reg int i=0;i<cnt;i++)
        if(k&(1ll<<i))res^=tmp[i];
    return res;
}
```

图

链式前向星

```
//存储结构
struct Edge {
    int to; //边的终点
    int w; //边的权值
    int next; //起点相同的下一条边
} edge[M]; //M 为边数，N 为顶点数
int head[N]; //head[i]是以 i 为起点的第一条边的编号
int cnt; //记录边数
//初始化
cnt = 0;
memset(head, -1, sizeof(head));
//建图
void addEdge(int u, int v, int w)
{
    edge[cnt].to = v;
    edge[cnt].w = w;
    edge[cnt].next = head[u];
    head[u] = cnt++;
}
//遍历以 u 为起点的邻接边
for (int i = head[u]; i != -1; i = edge[i].next) {
    int to = edge[i].to; //终点
    int w = edge[i].w; //权值
}
```

Dinic 算法

```
const int maxn=300;
const int INF=0x3f3f3f3f;
struct Edge{
    int to,next,cap;
}edge[maxn*maxn];
int head[maxn],tot;
int dep[maxn],cur[maxn];

void init(){
    tot=0;
    memset(head,-1,sizeof(head));
}

void addEdge(int u,int v,int c){
    edge[tot].to=v;edge[tot].cap=c;
    edge[tot].next=head[u];head[u]=tot++;
    edge[tot].to=u;edge[tot].cap=0;
    edge[tot].next=head[v];head[v]=tot++;
}


bool bfs(int s,int t){
    memset(dep,-1,sizeof(dep));
    for(int i=1;i<=n;i++)cur[i]=head[i];
    queue<int> que;
    dep[s]=0;
    que.push(s);
    while(que.size()){
        int u=que.front();que.pop();
        for(int i=head[u];i!=-1;i=edge[i].next){
            int v=edge[i].to;
            if(edge[i].cap>0&&dep[v]==-1){
                dep[v]=dep[u]+1;
                que.push(v);
            }
        }
    }
    if(dep[t]!=-1)return true;
    return false;
}

int dfs(int now,int t,int limit){
    if(!limit||now==t)return limit;
    int flow=0,f;

    for(int i=cur[now];i!=-1;i=edge[i].next){
        cur[now]=i;
        int v=edge[i].to;
        if(dep[v]==dep[now]+1&&(f=dfs(v,t,min(limit,edge[i].cap)))){
            flow+=f;
            limit-=f;
            edge[i].cap-=f;
            edge[i^1].cap+=f;
            if(!limit)break;
        }
    }
    return flow;
}

int dinic(int s,int t){
    int maxflow=0;
    while(bfs(s,t)){
        maxflow+=dfs(s,t,INF);
    }
    return maxflow;
}
```
数据结构

Treap

```
const int maxn=1e5+5;
const int INF=2e9+7;

struct Treap {
```

```
int ch[maxn][2];//结点左右儿子
int val[maxn],dat[maxn];//基本值和优先级
int size[maxn],cnt[maxn];//子树大小，结点副本数
int tot,root;
int New(int v) {
    val[++tot]=v;
    dat[tot]=rand();//随机优先级
    size[tot]=1;
    cnt[tot]=1;
    return tot;
}
void pushup(int id) {
    size[id]=size[ch[id][0]]+size[ch[id][1]]+cnt[id];
}
void build() {
    root=New(-INF),ch[root][1]=New(INF);
    pushup(root);
}
void rotate(int& id,int d) { //id 是引用传递，d(irection)为旋转方向，0
为左旋，1 为右旋
    int temp=ch[id][d^1];
    ch[id][d^1]=ch[temp][d];
    ch[temp][d]=id;
    id=temp;
    pushup(ch[id][d]),pushup(id);
}
void insert(int& id,int v) {
    if(!id) {
        id=New(v);
        return;
    }
    if(v==val[id])cnt[id]++;
    else {
        int d=v<val[id]?0:1;
        insert(ch[id][d],v);
        if(dat[id]<dat[ch[id][d]])rotate(id,d^1);
    }
    pushup(id);
```

```
}
void remove(int&id,int v) {
    if(!id)return;
    if(v==val[id]) {
        if(cnt[id]>1) {
            cnt[id]--,pushup(id);
            return;
        }
        if(ch[id][0]||ch[id][1]) {
            if(!ch[id][1]||dat[ch[id][0]]>dat[ch[id][1]]) {
                rotate(id,1),remove(ch[id][1],v);
            } else {
                rotate(id,0),remove(ch[id][0],v);
            }
            pushup(id);
        } else id=0;
        return;
    }
    v<val[id]?remove(ch[id][0],v):remove(ch[id][1],v);
    pushup(id);
}
int getRank(int id,int v) {
    if(!id)return 0;
    if(v==val[id])return size[ch[id][0]]+1;
    else if(v<val[id])return getRank(ch[id][0],v);
    else return size[ch[id][0]]+cnt[id]+getRank(ch[id][1],v);
}
int getVal(int id,int rank) {
    if(!id)return INF;
    if(rank<=size[ch[id][0]])return getVal(ch[id][0],rank);
    else if(rank<=size[ch[id][0]]+cnt[id])return val[id];
    else return getVal(ch[id][1],rank-size[ch[id][0]]-cnt[id]);
}
int getPre(int v) {
    int id=root,pre;
    while(id) {
        if(val[id]<v)pre=val[id],id=ch[id][1];
        else id=ch[id][0];
        }
        return pre;
    }
    int getNext(int v) {
        int id = root,next;
        while(id) {
            if(val[id] > v)next = val[id],id = ch[id][0];
            else id = ch[id][1];
        }
        return next;
    }
} trp;

int main() {
    trp.build();
    int n;
    scanf("%d",&n);
    while(n--) {
        int opt,x;
        scanf("%d%d",&opt,&x);
        int ans=-INF;
        int& r=trp.root;
        trp.insert(r,x);
        trp.remove(r,x);
        ans= trp.getRank(r,x)-1;
        ans=trp.getVal(r,x+1);
        ans=trp.getPre(x);
        ans=trp.getNext(x);
    }
}
```

堆(小顶)

```
int heap[N];
int sz;
void push(int x){
    heap[++sz]=x;
    int now=sz;
    while(now){
        ll nxt=now>>1;
        if(heap[nxt]>heap[now])swap(heap[nxt],heap[now]);
        else break;
        now=nxt;
    }
}

void pop(){
    swap(heap[1],heap[sz]);
    sz--;
    int now=1;
    while(LC(now)<=sz){
        int nxt=LC(now);
        if(nxt+1<=sz&&heap[nxt+1]<heap[nxt])nxt++;
        if(heap[nxt]<heap[now])swap(heap[nxt],heap[now]);
        else break;
        now=nxt;
    }
}
```

字符串

字典树 trie

```
struct Trie{
    int ch[M][27];
    int val[M];
    int sz;
    Trie(){sz=1;memset(ch[0],0,sizeof(ch[0]));}
    int idx(char c){return c-'a';}
    void insert(char s[],int v) {
        int u=0,n=strlen(s);
        for(int i=0;i<n;i++){
            int c=idx(s[i]);
            if(!ch[u][c]){
                memset(ch[sz],0,sizeof(ch[sz]));
                val[sz]=0;
                ch[u][c]=sz++;
            }
            u=ch[u][c];
        }
        val[u]=v;
    }
}tr;
```
AC 自动机
```
queue<int> q;
struct ACauto{
    int ch[N][26],val[N],fail[N],cnt;
    void ins(char* s){
        int len=strlen(s),u=0;
        for(int i=0;i<len;i++){
            int v=s[i]-'a';
            if(!ch[u][v])ch[u][v]=++cnt;
            u=ch[u][v];
        }
        val[u]++;
    }
void getFail(){
        for(int i=0;i<26;i++){if(ch[0][i]){
            fail[ch[0][i]]=0;
            q.push(ch[0][i]);
        }}
        while(!q.empty()){
            int u=q.front();q.pop();
            for(int i=0;i<26;i++){
               if(ch[u][i])fail[ch[u][i]]=ch[fail[u]][i],q.push(ch[u][i]);
else ch[u][i]=ch[fail[u]][i];
            }
        }
    }
    int query(char* s) {
        int len=strlen(s);int u=0,ans=0;
        for(int i=0;i<len;i++){
            u=ch[u][s[i]-'a'];
            for(int t=u;t&&~val[t];t=fail[t]){
                ans+=val[t],val[t]=-1;
            }
        }
        return ans;
    }
}AC;
```

其他

输入输出挂

```
//适用于正负整数
template <class T>
inline bool scan_d(T &ret) {
    char c;
    int sgn;
    if(c=getchar(),c==EOF) return 0; //EOF
    while(c!='-'&&(c<'0'||c>'9')) c=getchar();
    sgn=(c=='-')?-1:1;
    ret=(c=='-')?0:(c-'0');
    while(c=getchar(),c>='0'&&c<='9') ret=ret*10+(c-'0');
    ret*=sgn;
    return 1;
}

inline void out(int x) {
    if(x>9) out(x/10);
    putchar(x%10+'0');
}
```