数据结构

线段树

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define INF 0x7fffffff
#define ll long long
const int mod = 1e9 + 7;
const int maxn = 1e6 + 100;
struct Node
{
    int left,right,pos,lazy,v;
}node[4*maxn];
int a[maxn];
int build(int left, int right, int pos=1)
{
    node[pos].left = left;
    node[pos].right  = right;
    node[pos].lazy = 0;
    if (left == right){
        node[pos].v  = a[left];
        return node[pos].v;
    }
    int mid =(left+right)>>1;
    int l = build(left,mid,2*pos);
    int r = build(mid+1,right,2*pos+1);
    return node[pos].v = max(l,r);
}
inline void push(int pos)
{
    if (!node[pos].lazy) return ;
    node[2*pos].lazy += node[pos].lazy;
    node[2*pos+1].lazy += node[pos].lazy;
    node[2*pos].v += node[pos].lazy;
    node[2*pos+1].v += node[pos].lazy;
    node[pos].lazy = 0;
}
int update(int l,int r,int v,int pos=1)
{
    if (l>node[pos].right || r<node[pos].left)
        return node[pos].v;
    if (l<=node[pos].left && node[pos].right<=r){
        node[pos].lazy += v;
        return node[pos].v+=v;
    }
    push(pos);
    int p = update(l,r,v,2*pos);
    int q = update(l,r,v,2*pos+1);
    return node[pos].v = max(p,q);
}
int query(int l,int r,int pos=1)
{
    if (l>node[pos].right || r<node[pos].left)
        return 0;
    if (l<=node[pos].left && node[pos].right<=r)
        return node[pos].v;
    push(pos);
    int p = query(l,r,2*pos);
    int q = query(l,r,2*pos+1);
    return max(p,q);
}
int main()
{
    int n,m;
```

```
    scanf("%d%d",&n,&m);
    for (int i=1;i<=n;++i)
        scanf("%d",&a[i]);
    build(1,n);
    int op;
    int x,y,v;
    while (m--){
        scanf("%d",&op);
        if (op){
            scanf("%d%d",&x,&y);
            printf("%d\n",query(x,y));
        }
        else{
            scanf("%d%d%d",&x,&y,&v);
            update(x,y,v);
        }
    }
    return 0;
}
```

主席树

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define MP(a,b) make_pair(a,b)
#define INF 0x7fffffff
#define ll long long
const int mod = 1e9 + 7;
const int maxn = 1e5 + 100;
struct Node
{
    Node *left,*right;
    int sum;
    void init(Node *l=NULL,Node *r=NULL,int s=0)
    {
        left = l,right = r,sum=s;
    }
}node[maxn*20],*root[maxn];
int num[maxn],a[maxn],p[maxn],cnt,n;
Node *modify(int left,int right,int v,Node *p)
{
    if (v<left || v>right) return p;
    Node *t = &node[cnt++];
    *t = *p;
    ++t->sum;
    if (left==right) return t;
    int mid = (left+right)/2;
    t->left = modify(left,mid,v,p->left);
    t->right = modify(mid+1,right,v,p->right);
    return t;
}
inline void build()
{
    cnt = 0;
    root[0] = &node[cnt++];
    root[0]->init(root[0],root[0],0);
    for (int i=1;i<=n;++i)
        root[i] = modify(1,n,num[i],root[i-1]);
}
int query(int left,int right,int k,Node *x,Node *y)
{
    if (left==right) return left;
    int mid = (left+right)/2;
    int sum = y->left->sum - x->left->sum;
    if (sum>=k) return query(left,mid,k,x->left,y->left);
    return query(mid+1,right,k-sum,x->right,y->right);
}
```

```
int main()
{
    int T,m;
    scanf("%d",&T);
    while (T--){
        scanf("%d%d",&n,&m);
        for (int i=1;i<=n;++i){
            scanf("%d",&a[i]);
            p[i]=a[i];
        }
        sort(p+1,p+1+n);
        for (int i=1;i<=n;++i)
            num[i] = lower_bound(p+1,p+1+n,a[i])-p;
        build();
        int l,r,k,pos;
        for (int i=0;i<m;++i){
            scanf("%d%d%d",&l,&r,&k);
            pos = query(1,n,k,root[l-1],root[r]);
            printf("%d\n",p[pos]);
        }
    }
    return 0;
}
```

二维树状数组

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define MP(a,b) make_pair(a,b)
#define INF 0x7fffffff
#define LL long long
const int MOD = 1e9 + 7;
const int maxn = 1e3 + 100;
int sum[maxn][maxn];
int n;
int lowbit(int x){return x&(-x);}
void update(int x,int y,int value)
{
    for (int i=x;i<=n;i+=lowbit(i))
        for (int j=y;j<=n;j+=lowbit(j))
            sum[i][j]+=value;
}
LL query(int x,int y)
{
    LL ret=0;
    for (int i=x;i>0;i-=lowbit(i))
        for (int j=y;j>0;j-=lowbit(j))
            ret+=sum[i][j];
    return ret;
}
int main()
{
    return 0;
}
```

字符串

KMP

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
```

```
#define MP(a,b) make_pair(a,b)
#define INF 0x7fffffff
#define LL long long
const int MOD = 1e9 + 7;
const int maxn = 1e6 + 100;
int next[maxn];
char s[maxn];
char str[maxn];
void getNext(char *s,int *next)
{
    int k=-1;
    int len=strlen(s);
    next[0]=-1;
    for (int i=1;i<len;++i){
        while (k>=0&&s[k+1]!=s[i]) k=next[k];
        if (s[k+1]==s[i]) ++k;
        next[i]=k;
    }
}
int KMPmatch(char *s,char *str,int *next)
{
    getNext(s,next);
    int len=strlen(s);
    int k=-1;
    for (int i=0;str[i];++i){
        while (k>=0&&s[k+1]!=str[i]) k=next[k];
        if (s[k+1]==str[i]) ++k;
        if (k+1==len)
            return i-len+1;
    }
}
int main()
{
    cin>>str>>s;
    cout<<KMPmatch(s,str,next)<<endl;
    return 0;
}
```

AC自动机

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define MP(a,b) make_pair(a,b)
#define INF 0x7fffffff
#define ll long long
const int mod = 1e9 + 7;
const int maxn = 1e6 + 100;
#define N 26
struct Node
{
    int count;
    Node *fail,*next[N];
    void init(int c=0,Node *f=NULL)
    {
        CL(next);
        fail = f;
        count = c;
    }
}node[maxn];
int cnt;
char s[maxn];

inline void insert(Node *p,char *s)
{
    for (int i=0;s[i];++i){
        int id = s[i]-'a';
        if (!p->next[id]){
```

```
            p->next[id] = &node[cnt++];
            p->next[id]->init();
        }
        p = p->next[id];
    }
    ++p->count;
}
inline void buildAC(Node *root)
{
    queue<Node*> Q;
    Node *p = root;
    p->fail = NULL;
    Q.push(p);
    Node *t;
    while (!Q.empty()){
        p = Q.front();
        Q.pop();
        for (int i=0;i<N;++i){
            if (!p->next[i]) continue;
            Q.push(p->next[i]);
            if (p == root) p->next[i]->fail = root;
            else {
                t = p->fail;
                while (t){
                    if (t->next[i]){
                        p->next[i]->fail = t->next[i];
                        break;
                    }
                    t = t->fail;
                }
                if (!t) p->next[i]->fail = root;
            }
        }
    }
}
inline int query(Node *root,char *s)
{
    Node *p = root;
    Node *t;
    int ans = 0;
    for (int i=0;s[i];++i){
        int id = s[i]-'a';
        while (p!=root && p->next[id]==NULL)
            p = p->fail;
        p = p->next[id];
        if (!p) p = root;
        t = p;
        while (t!=root && t->count!=-1){
            ans += t->count;
            t->count = -1;
            t=t->fail;
        }
    }
    return ans;
}
int main()
{
    int T,n;
    scanf("%d",&T);
    while (T--){
        scanf("%d",&n);
        Node * root = &node[cnt++];
        root->init();
        for (int i=0;i<n;++i){
            scanf("%s",s);
            insert(root,s);
        }
        buildAC(root);
        scanf("%s",s);
        printf("%d\n",query(root,s));
    }
    return 0;
```

```
}
```

tarjan

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define INF 0x7fffffff
#define LL long long
const int MOD = 1e9 + 7;
const int maxn = 1e3 + 100;
int dfn[maxn];//点i被访问的次序，时间戳
int low[maxn];//点i可回溯到的在栈中最早的节点
int belong[maxn];
bool inStack[maxn];//点i是否在栈中
int Index;//节点被访问次序的编号
vector<int>edge[maxn];//边
vector<int>edge2[maxn];//DAG图的边
stack<int>S;//栈
int cnt = 0;//强连通分量的总数/DAG图中点的个数
int n;//节点数
int m;//边数
void init()//初始化
{
    Index = cnt = 0;
    CL(dfn);
    CL(low);
    CL(inStack);
    CL(belong);
    for (int i = 0;i<maxn;i++){
        edge2[i].clear(); edge[i].clear();
    }
}
void tarjan(int u)
{
    dfn[u] = low[u] = ++Index;
    S.push(u);
    inStack[u] = true;
    for (int i = 0;i<edge[u].size();i++){
        int v = edge[u][i];
        if (!dfn[v]){
            tarjan(v);
            low[u] = min(low[u],low[v]);
        }
        else if (inStack[v])
            low[u] = min(low[u],dfn[v]);
    }
    if (dfn[u] ==  low[u]){
        int t;
        ++cnt;
        do{
            t = S.top();
            belong[t] = cnt;
            S.pop();
            inStack[t] = false;
        }while (t != u);
    }
}
void buildDAG()
{
    for (int i = 1;i <= n;i++){
        for (int j = 0;j<edge[i].size();j++){
            int v = edge[i][j];
            if (belong[i] != belong[v])
                edge2[belong[i]].push_back(belong[v]);
        }
    }
}
```

```
int main()
{
    return 0;
}
```

LCA

```
#include <bits/stdc++.h>
using namespace std;
#define CL(a) memset(a,0,sizeof(a))
#define Cl(a,b) memset(a,b,sizeof(a))
#define MP(a,b) make_pair(a,b)
#define INF 0x7fffffff
#define ll long long
#define ull unsigned long long
const int mod = 1e9 + 7;
const int maxn = 1e5 + 100;
vector<int> G[maxn];
int grand[maxn][17],depth[maxn];
int Log[maxn],a[maxn],b[maxn],tmp[maxn];
int n,u,v;
inline void init(int n)
{
    for (int i=1;i<=n;++i)
        G[i].clear();
    CL(depth);
    CL(grand);
}
void dfs(int from=0,int u=1)
{
    if (depth[u]) return ;
    depth[u] = depth[from]+1;
    grand[u][0] = from;
    for (int i=1;depth[u]-(1<<i) > 0; ++i)
        grand[u][i] = grand[grand[u][i-1]][i-1];
    for (int i=0;i<G[u].size();++i)
        dfs(u,G[u][i]);
}

inline int lca(int a,int b)
{
    if (depth[a]<depth[b])
        swap(a,b);

    int t=depth[a]-depth[b];
    for (int i=0;t;t>>=1,++i)
        if (t&1) a = grand[a][i];
    if (a==b) return a;

    for (int i=Log[n];i>=0;--i){
        if (grand[a][i]!=grand[b][i]){
            a = grand[a][i];
            b = grand[b][i];
        }
    }
    return grand[a][0];
}

inline bool check(int d)
{
    int cnt = 0;
    for (int i=0;i<u;++i){
        int k = a[i];
        if (depth[k]<d) continue;
        int t = depth[k]-d;
        for (int j=0;t;t>>=1,++j)
            if (t&1) k = grand[k][j];
        tmp[cnt++] = k;
    }
```

```
    if (cnt==0) return false;
    sort(tmp,tmp+cnt);
    cnt = unique(tmp,tmp+cnt)-tmp;

    for (int i=0;i<cnt;++i){
        for (int j=0;j<v;++j){
            if (lca(tmp[i],b[j]) == tmp[i])
                return true;
        }
    }
    return false;
}
int main()
{
    int m;
    for (int i=1;i<=10000;++i)
        Log[i] = Log[i/2]+1;
    while (scanf("%d%d",&n,&m)!=EOF){
        init(n);
        for (int i=1;i<n;++i){
            scanf("%d%d",&u,&v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        dfs();
        while (m--){
            scanf("%d",&u);
            for (int i=0;i<u;++i)
                scanf("%d",&a[i]);
            scanf("%d",&v);
            for (int i=0;i<v;++i)
                scanf("%d",&b[i]);
            int l=1;
            int r=n+1;
            while (l+1<r) {
                int mid = (l+r)>>1;
                if (check(mid))
                    l = mid;
                else
                    r = mid;
            }
            printf("%d\n",l);
        }
    }
    return 0;
}
```