|  |  |
| --- | --- |
| KD\_TREE  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133 | */\**  *寻找目标最近的num个点对*    *首先通过二叉树搜索（比较待查询节点和分裂节点的分裂维的值，小于等于就进入左子树分支，等于就进入右子树分支直到叶子结点），顺着“搜索路径”很快能找到最近邻的近似点，也就是与待查询点处于同一个子空间的叶子结点；然后再回溯搜索路径，并判断搜索路径上的结点的其他子结点空间中是否可能有距离查询点更近的数据点，如果有可能，则需要跳到其他子结点空间中去搜索（将其他子结点加入到搜索路径）。重复这个过程直到搜索路径为空。*  *\*/*  #include<iostream>  #include<cstring>  #include<cstdio>  #include<algorithm>  #include<queue>  **using** **namespace** std;  **const** **int** maxn=5e4+50;  **const** **int** K=6;  **int** n,m;  **int** num;  **struct** point{  **int** p[K];  **int** split;  **bool** leaf;  **long** **long** dis;  **void** Print(){  **for** (**int** i=0;i<m;i++) printf("%d%c",p[i],i==m-1?'\n':' ');  }  **bool** **operator**<(**const** point& b)**const**{  **return** dis<b.dis;  }  **void** Input(){  **for** (**int** i=0;i<m;i++) scanf("%d",&p[i]);  }  }a[maxn],tar;  priority\_queue<point> q;  **int** cmp\_NO;  **bool** cmp(**const** point&a,**const** point&b){  **return** a.p[cmp\_NO]<b.p[cmp\_NO];  }  **void** Build(**int** l,**int** r,**int** d){  **int** mid=(l+r)>>1;  cmp\_NO=d;  nth\_element(a+l,a+mid,a+r+1,cmp);  a[mid].split=d;  **if** (l==r){  a[l].leaf=1;  **return** ;  }  a[mid].leaf=0;  **if** (l!=mid) Build(l,mid-1,(d+1)%m);  **if** (r!=mid) Build(mid+1,r,(d+1)%m);  }  **long** **long** Dist(point a1,point a2){  **long** **long** ret=0;  **for** (**int** i=0;i<m;i++) ret+=(a1.p[i]-a2.p[i])\*(a1.p[i]-a2.p[i]);  **return** ret;  }  **void** Find\_kd(**int** l,**int** r){  **if** (l>r) **return** ;  **int** mid=(l+r)>>1;  a[mid].dis=Dist(a[mid],tar);  **if** (l==r){  **if** (q.size()<num){  q.push(a[l]);  }  **else** **if** (q.top().dis>a[l].dis){  q.pop();  q.push(a[l]);  }  **return** ;  }  **long** **long** t=(tar.p[a[mid].split]-a[mid].p[a[mid].split]);  **if** (t<0){  Find\_kd(l,mid-1);  **if** (q.size()<num){  q.push(a[mid]);  Find\_kd(mid+1,r);  }  **else**{  **if** (a[mid].dis<q.top().dis){  q.pop();  q.push(a[mid]);  }  **if** (t\*t<q.top().dis){  Find\_kd(mid+1,r);  }  }  }  **else**{  Find\_kd(mid+1,r);  **if** (q.size()<num){  q.push(a[mid]);  Find\_kd(l,mid-1);  }  **else**{  **if** (a[mid].dis<q.top().dis){  q.pop();  q.push(a[mid]);  }  **if** (t\*t<q.top().dis){  Find\_kd(l,mid-1);  }  }  }  }  **void** Print(){  **if** (q.empty()) **return** ;  point tmp=q.top();  q.pop();  Print();  tmp.Print();  }  **int** main(){  **while** (scanf("%d%d",&n,&m)!=EOF){  **for** (**int** i=1;i<=n;i++) a[i].Input();  Build(1,n,0);  **int** t;  scanf("%d",&t);  **while** (!q.empty()) q.pop();  **while** (t--){  tar.Input();  scanf("%d",&num);  Find\_kd(1,n);  printf("the closest %d points are:**\n**",num);  Print();  }  }  **return** 0;  } |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139 | *kd\_tree(带插入删除，左右子树不平衡时就rebuild)*  *#include<iostream>*  *#include<cstring>*  *#include<cstdio>*  *#include<algorithm>*  *#include<vector>*  *using namespace std;*  *const int maxn=5e4+50;*  *const double mul=0.7;*  *int n,Q,W,H;*  *vector<int> q;*  *int ans;*  *struct point{*  *int x,y,id,p[2];*  *void trans(){*  *p[0]=x+y;*  *p[1]=x-y;*  *}*  *}p[maxn],tp[maxn];*  *struct node{*  *point mid;*  *int siz;*  *bool exist;*  *node \*ls,\*rs;*  *node(){siz=exist=0;}*  *bool judge(){*  *return !siz || ls->siz>siz\*mul ||rs->siz>siz\*mul;*  *}*  *}pool[maxn\*5],\*root,\*null=&pool[0],\*ptr[maxn];*  *int tot,tot2,tp\_num;*  *int cmp\_NO;*  *bool cmp(const point&a,const point&b){*  *return a.p[cmp\_NO]<b.p[cmp\_NO] || (a.p[cmp\_NO]==b.p[cmp\_NO] && a.id<b.id);*  *}*  *node\* Build(int l,int r,int d){*  *if (l>r) return null;*  *node \*p=tot2?ptr[tot2--]:&pool[++tot];*  *cmp\_NO=d;*  *int mid=(l+r)>>1;*  *nth\_element(tp+l,tp+mid,tp+r+1,cmp);*  *p->exist=1;*  *p->mid=tp[mid];*  *p->ls=Build(l,mid-1,!d);*  *p->rs=Build(mid+1,r,!d);*  *p->siz=p->ls->siz+p->rs->siz+1;*  *return p;*  *}*  *void dfs(node\* root){*  *if (root==null) return ;*  *if (root->exist){*  *ptr[++tot2]=root;*  *tp[++tp\_num]=root->mid;*  *}*  *dfs(root->ls);*  *dfs(root->rs);*  *}*  *void Rebuild(node\* &root,int d){*  *tp\_num=0;*  *dfs(root);*  *root=Build(1,tp\_num,d);*  *}*  *void Get\_vec(node\* &root,int d,int lx,int rx,int ly,int ry,bool flg){*  *if (root->siz==0) return ;*  *if (root->exist && lx<= root->mid.p[0] && root->mid.p[0]<=rx && ly<=root->mid.p[1] && root->mid.p[1]<=ry){*  *root->exist=0;*  *root->siz--;*  *q.push\_back(root->mid.id);*  *}*  *bool tag=root->judge();*  *if (!d?lx<=root->mid.p[0]:ly<=root->mid.p[1]) Get\_vec(root->ls,!d,lx,rx,ly,ry,flg|tag);*  *if (!d?rx>=root->mid.p[0]:ry>=root->mid.p[1]) Get\_vec(root->rs,!d,lx,rx,ly,ry,flg|tag);*  *root->siz=root->ls->siz+root->rs->siz+root->exist;*  *if (tag && !flg) Rebuild(root,d);*  *}*  *void Insert(node\* &root,int d,point a,bool flg){*  *if (root==null){*  *root=&pool[++tot];*  *root->mid=a;*  *root->ls=null;*  *root->rs=null;*  *root->siz=1;*  *root->exist=1;*  *return ;*  *}*  *root->siz++;*  *bool tag=root->judge();*  *if (a.p[d]<root->mid.p[d] || (a.p[d]==root->mid.p[d] && a.id<root->mid.id)) Insert(root->ls,!d,a,flg|tag);*  *else Insert(root->rs,!d,a,flg|tag);*  *root->siz=root->ls->siz+root->rs->siz+root->exist;*  *if (tag && !flg) Rebuild(root,d);*  *}*  *int main(){*  *int T;*  *scanf("%d",&T);*  *for (int cas=1;cas<=T;cas++){*  *tot=tot2=0;*  *printf("Case #%d:\n",cas);*  *scanf("%d%d%d%d",&n,&Q,&W,&H);*  *for (int i=1;i<=n;i++){*  *scanf("%d%d",&p[i].x,&p[i].y);*  *p[i].id=i;*  *p[i].trans();*  *tp[i]=p[i];*  *}*  *root=Build(1,n,0);*  *while (Q--){*  *int x,y,E,a,b,c,d,e,f,lx,rx,ly,ry;*  *scanf("%d%d%d%d%d%d%d%d%d",&x,&y,&E,&a,&b,&c,&d,&e,&f);*  *lx = x + y-E;*  *ly = x - y-E;*  *rx = x + y+E;*  *ry = x - y+E;*    *q.clear();*  *Get\_vec(root,0,lx,rx,ly,ry,0);*  *for (auto i:q){*  *long long xi=p[i].x;*  *long long yi=p[i].y;*  *p[i].x= ((xi\*a) + (yi\*b) + (1ll\*i\*c)) % W;*  *p[i].y= ((xi\*d) + (yi\*e) + (1ll\*i\*f)) % H;*  *p[i].trans();*  *Insert(root,0,p[i],0);*  *}*  *}*  *for (int i=1;i<=n;i++) printf("%d %d\n",p[i].x,p[i].y);*  *}*  *return 0;*  *}* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165 | */\**  *树分治*  *注意map中的一些细节*  *map{1,2,3,4,5}*  *iterator it=lower\_bound(3); (\*it=3)*  *reverser\_iterator rit=reverse\_iterator(it); (\*rit=2)*  *it=rit.base(); (\*it=3)*  *it=((++rit).base()); (\*it=2)*    *题目大意：给出一棵树有1≤n≤105个节点，每个节点有个权值1≤ai≤105，求一个由节点构成的*  *最长权值不降连续子串，且串的最大值和最小值的差diff≤D,1≤D≤105，样例数T≤10。*  *思路：用树分治比较方便。考虑这条路径经过根和不经过根两种情况；不经过根就到其子树中去找，*  *经过根就得先维护一个上升串（离根越远值越小），然后在下降串中每次到上升串中二分即可。*  *在上升串中，如果dep[u]≥dep[v]&&val[u]≥val[v]，那么u在任何时候都不会比v差，那么v就可以删除掉。*  *\*/*  *#include<map>*  *#include<cstring>*  *#include<cstdio>*  *#include<cstring>*  *#include<iostream>*  *#include<vector>*  *#include<algorithm>*  *using namespace std;*  *const int maxn=1e5+50;*  *const int maxm=maxn\*2;*  *int head[maxn],ver[maxm],nex[maxm],tot;*  *int n,D;*  *int a[maxn];*  *bool del[maxn];*  *int ans;*  *vector<int> nodes;*  *int maxBranch[maxn],siz[maxn];*  *map<int,int> up;*  *map<int,int>:: iterator it;*  *map<int,int>:: reverse\_iterator rit;*  *vector<int> son;*  *void add(int x,int y){*  *ver[++tot]=y;*  *nex[tot]=head[x];*  *head[x]=tot;*  *}*  *//寻找中心根*  *void Dfs\_siz(int u,int f){*  *nodes.push\_back(u);*  *siz[u]=1;*  *maxBranch[u]=0;*  *for (int i=head[u];i;i=nex[i]){*  *if (ver[i]!=f && !del[ver[i]]){*  *Dfs\_siz(ver[i],u);*  *siz[u]+=siz[ver[i]];*  *maxBranch[u]=max(maxBranch[u],siz[ver[i]]);*  *}*  *}*  *}*  *int Get\_root(int u){*  *nodes.clear();*  *Dfs\_siz(u,-1);*  *int rt=u;*  *for (auto v: nodes){*  *maxBranch[v]=max(maxBranch[v],siz[u]-siz[v]);*  *if (maxBranch[v]<maxBranch[rt]) rt=v;*  *}*  *return rt;*  *}*  *// 根据题目具体对答案的处理*  *void Update(int val,int len){*  *it=up.lower\_bound(val);*  *if (it!=up.end()&& it->second>=len) return ;*  *it=up.upper\_bound(val);*  *rit=map<int,int>::reverse\_iterator(it);*  *while (rit!=up.rend() && rit->second<=len) rit++;*  *up.erase(rit.base(),it);*  *up[val]=len;*  *}*  *void Work\_up(int u,int f,int len){*  *Update(a[u],len);*  *for (int i=head[u];i;i=nex[i]) if (!del[ver[i]] && ver[i]!=f && a[ver[i]]<=a[u]){*  *Work\_up(ver[i],u,len+1);*  *}*  *}*  *void Work\_down(int u,int f,int len){*  *it=up.lower\_bound(a[u]-D);*  *if (it==up.end()) return ;*  *ans=max(ans,it->second+len+1);*  *for (int i=head[u];i;i=nex[i]) if (!del[ver[i]] && ver[i]!=f && a[ver[i]]>=a[u]){*  *Work\_down(ver[i],u,len+1);*  *}*  *}*  *void \_work(int u){*  *up.clear();*  *up[a[u]]=0;*  *for (auto v:son){*  *if (a[v]>=a[u] && !del[v]) Work\_down(v,u,1);*  *if (a[v]<=a[u] && !del[v]) Work\_up(v,u,1);*  *}*  *}*  *void Work(int u){*  *son.clear();*  *for (int i=head[u];i;i=nex[i]) if (!del[ver[i]]) son.push\_back(ver[i]);*  *\_work(u);*  *reverse(son.begin(),son.end());*  *\_work(u);*  *}*  *void Solve(int u){*  *u=Get\_root(u);*  *Work(u);*  *del[u]=1;*  *for (int i=head[u];i;i=nex[i]) if (!del[ver[i]]){*  *Solve(ver[i]);*  *}*  *}*  *int main(){*  *int T;*  *int cas=0;*  *scanf("%d",&T);*  *while (T--){*  *scanf("%d%d",&n,&D);*  *tot=0;*  *for (int i=1;i<=n;i++) head[i]=0;*  *for (int i=1;i<=n;i++) scanf("%d",&a[i]);*  *for (int i=1;i<n;i++){*  *int u,v;*  *scanf("%d%d",&u,&v);*  *add(u,v);*  *add(v,u);*  *}*  *for (int i=1;i<=n;i++){*  *del[i]=0;*  *}*  *ans=1;*  *Solve(1);*  *printf("Case #%d: %d\n",++cas,ans);*  *}*  *return 0;*  *}*  */\**  *3*  *3 1*  *1 2 3*  *1 2*  *2 3*  *3 2*  *1 2 3*  *1 2*  *2 3*  *7 4*  *1 3 4 5 7 8 3*  *1 3*  *1 4*  *2 3*  *3 5*  *5 7*  *6 3*  *\*/* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247 | *//Treap*  *#include<cstring>*  *#include<iostream>*  *#include<cstdio>*  *#include<cstdlib>*  *using namespace std;*  *#define MAXN 50005*  *struct t\_node{*  *int root,l,r;*  *}t[MAXN\*5];*  *struct treap\_node{*  *int l,r,ls,rs,nums,dat,pri;*  *}treap[MAXN\*25];*  *int tot,a[MAXN],n,m;*  *bool flag,hx;*  *char ch;*  *void read(int &x){*  *hx=0;*  *while ((ch=getchar())&& (ch>'9' || ch<'0')){*  *if (ch=='-')hx=1;*  *};*  *x=ch-'0';*  *while ((ch=getchar())&& (ch<='9' && ch>='0'))x=x\*10+ch-'0';*  *if (hx)x=-x;*  *}*  *void left(int &root){*  *int temp=treap[root].l;*  *treap[root].l=treap[temp].r;*  *treap[root].ls=treap[temp].rs;*  *treap[temp].r=root;*  *treap[temp].rs+=treap[root].rs+treap[root].nums;*  *root=temp;*  *}*  *void right(int &root){*  *int temp=treap[root].r;*  *treap[root].r=treap[temp].l;*  *treap[root].rs=treap[temp].ls;*  *treap[temp].l=root;*  *treap[temp].ls+=treap[root].ls+treap[root].nums;*  *root=temp;*  *}*  *void insert(int &root,int x){*  *if (!root){*  *root=++tot;*  *treap[root].dat=x;*  *treap[root].nums=1;*  *treap[root].l=treap[root].r=treap[root].ls=treap[root].rs=0;*  *treap[root].pri=rand();*  *return ;*  *}*  *if (treap[root].dat==x){*  *treap[root].nums++;*  *return ;*  *}*  *if (x<treap[root].dat){*  *insert(treap[root].l,x);*  *treap[root].ls++;*  *if (treap[root].pri>treap[treap[root].l].pri){*  *left(root);*  *}*  *}*  *else{*  *insert(treap[root].r,x);*  *treap[root].rs++;*  *if (treap[root].pri>treap[treap[root].r].pri){*  *right(root);*  *}*  *}*  *}*  *void del(int &root,int x){*  *if (treap[root].dat==x){*  *if (treap[root].nums) treap[root].nums--;*  *if (!treap[root].nums){*  *if (!treap[root].l){*  *root=treap[root].r;*  *}*  *else if (!treap[root].r){*  *root=treap[root].l;*  *}*  *else{*  *if (treap[treap[root].l].pri<treap[treap[root].r].pri){*  *left(root);*  *del(treap[root].r,x);*  *}*  *else{*  *right(root);*  *del(treap[root].l,x);*  *}*  *}*  *}*  *return ;*  *}*  *if (x<treap[root].dat){*  *del(treap[root].l,x);*  *treap[root].ls--;*  *}*  *else{*  *del(treap[root].r,x);*  *treap[root].rs--;*  *}*  *}*  *int ask\_rank(int root,int x){*  *if (!root) return 0;*  *if (treap[root].dat==x){*  *flag=1;*  *return treap[root].ls;*  *}*  *if (x<treap[root].dat){*  *return ask\_rank(treap[root].l,x);*  *}*  *return ask\_rank(treap[root].r,x)+treap[root].ls+treap[root].nums;*  *}*  *void build(int p,int l,int r){*  *t[p].l=l;t[p].r=r;t[p].root=0;*  *for (int i=l;i<=r;i++) insert(t[p].root,a[i]);*  *if (l==r) return ;*  *int mid=(l+r)>>1;*  *build(p\*2,l,mid);*  *build(p\*2+1,mid+1,r);*  *}*  *int ask(int p,int l,int r,int k){*  *if (t[p].l>=l && t[p].r<=r){*  *return ask\_rank(t[p].root,k);*  *}*  *int ans=0;*  *if (t[p\*2].r>=l) ans+=ask(p\*2,l,r,k);*  *if (t[p\*2+1].l<=r) ans+=ask(p\*2+1,l,r,k);*  *return ans;*  *}*  *int ask\_pre(int root,int x){*  *if (!root) return 0;*  *if (x>treap[root].dat){*  *return max(treap[root].dat,ask\_pre(treap[root].r,x));*  *}*  *else{*  *return ask\_pre(treap[root].l,x);*  *}*  *}*  *int ask\_succ(int root,int x){*  *if (!root) return 2000000000;*  *if (x<treap[root].dat){*  *return min(treap[root].dat,ask\_succ(treap[root].l,x));*  *}*  *else{*  *return ask\_succ(treap[root].r,x);*  *}*  *}*  *int work(int l,int r,int k){*  *int now,temp,ans;*  *now=t[1].root;*  *flag=0;*  *temp=ask(1,l,r,treap[now].dat);*  *while (now){*  *if (k>=temp+1){*  *if (flag) ans=treap[now].dat;*  *now=treap[now].r;*  *if (now){*  *flag=0;*  *temp=ask(1,l,r,treap[now].dat);*  *}*  *}*  *else{*  *now=treap[now].l;*  *if (now){*  *flag=0;*  *temp=ask(1,l,r,treap[now].dat);*  *}*  *}*  *}*  *return ans;*  *}*  *void change(int p,int x,int y){*  *del(t[p].root,a[x]);*  *insert(t[p].root,y);*  *if (t[p].l==t[p].r) return ;*  *if (x<=t[p\*2].r) change(p\*2,x,y);*  *else change(p\*2+1,x,y);*  *}*  *int work\_pre(int p,int l,int r,int x){*  *if (t[p].l>=l && t[p].r<=r){*  *return ask\_pre(t[p].root,x);*  *}*  *int ans=0;*  *if (t[p\*2].r>=l) ans=max(ans,work\_pre(p\*2,l,r,x));*  *if (t[p\*2+1].l<=r) ans=max(ans,work\_pre(p\*2+1,l,r,x));*  *return ans;*  *}*  *int work\_succ(int p,int l,int r,int x){*  *if (t[p].l>=l && t[p].r<=r){*  *return ask\_succ(t[p].root,x);*  *}*  *int ans=2000000000;*  *if (t[p\*2].r>=l) ans=min(ans,work\_succ(p\*2,l,r,x));*  *if (t[p\*2+1].l<=r) ans=min(ans,work\_succ(p\*2+1,l,r,x));*  *return ans;*  *}*  *int main(){*  *int i,opt,l,r,k,x,y;*  *scanf("%d%d",&n,&m);*  *for (i=1;i<=n;i++){*  *scanf("%d",&a[i]);*  *}*  *build(1,1,n);*  *while (m--){*  *scanf("%d",&opt);*  *switch (opt){*  *case 1:{//1.查询k在区间内的排名*  *read(l);read(r);read(k);*  *printf("%d\n",ask(1,l,r,k)+1);*  *break;*  *}*  *case 2:{//2.查询区间内排名为k的值*  *read(l);read(r);read(k);*  *printf("%d\n",work(l,r,k));*  *break;*  *}*  *case 3:{//3.修改某一位值上的数值*  *read(x);read(y);*  *change(1,x,y);*  *a[x]=y;*  *break;*  *}*  *case 4:{//4.查询k在区间内的前驱(前驱定义为小于x，且最大的数)*  *read(l);read(r);read(k);*  *printf("%d\n",work\_pre(1,l,r,k));*  *break;*  *}*  *case 5:{//5.查询k在区间内的后继(后继定义为大于x，且最小的数)*  *read(l);read(r);read(k);*  *printf("%d\n",work\_succ(1,l,r,k));*  *break;*  *}*  *}*  *}*  *return 0;*  *}* |