

[back](#)

[test_2](#)

这里有一个公式
$$\sqrt{\frac{2}{3}}\lim_{x\rightarrow\infty}\{\frac{x}{e^x}=0\}$$

这里有一段代码

```
#include <con>
#include <iostream>
#include <set>
int main()
{
    std::set<int> s;
    for (int i=1;i<=1000;i++)
        s.insert(i*i%1234);
    for (auto i:s)
        std::cout<<i<<std::endl;
    return 0;
}
```

这里有一个表格：

算法	时间复杂度	空间复杂度
冒泡排序	$O(n^2)$	$O(1)$
堆排序	$O(n\log n)$	$O(1)$

```
#include<cstdio> #include<map> #include<vector> #include<cstring> #include<algorithm>
#define _for(i,a,b) for(int i=(a);i<(b);i++) #define _rep(i,a,b) for(int i=(a);i<=(b);i++) #define
mem(a,b) memset(a,b,sizeof(a)) using namespace std; typedef long long LL; typedef
vector<int>::iterator iter; int read_int(){
```

```
int t;
scanf("%d",&t);
return t;
```

```
} void space(LL x){
```

```
printf("%lld ",x);
```

```
} void enter(LL x){
```

```
printf("%lld\n",x);
```

```
} const int MAXN=1e5+5; namespace Tree{
```

```
int init_val;
int
lef[MAXN<<2], rig[MAXN<<2], maxv[MAXN<<2], maxc[MAXN<<2], secv[MAXN<<2], lazy[MAX
N<<2];
LL sum[MAXN<<2];
```

```
void push_up(int k) {
    sum[k]=(sum[k<<1]+sum[k<<1|1]);
    if(maxv[k<<1]==maxv[k<<1|1]) {
        maxv[k]=maxv[k<<1];
        maxc[k]=maxc[k<<1]+maxc[k<<1|1];
        secv[k]=max(secv[k<<1],secv[k<<1|1]);
    }
    else if(maxv[k<<1]>maxv[k<<1|1]) {
        maxv[k]=maxv[k<<1];
        maxc[k]=maxc[k<<1];
        secv[k]=max(secv[k<<1],maxv[k<<1|1]);
    }else {
        maxv[k]=maxv[k<<1|1];
        maxc[k]=maxc[k<<1|1];
        secv[k]=max(maxv[k<<1],secv[k<<1|1]);
    }
}

void push_tag(int k,int v) {
    if(v>=maxv[k]) return;
    sum[k]-=1LL*maxc[k]*(maxv[k]-v);
    maxv[k]=lazy[k]=v;
}

void push_down(int k) {
    if(~lazy[k]) {
        push_tag(k<<1,lazy[k]);
        push_tag(k<<1|1,lazy[k]);
        lazy[k]=-1;
    }
}

void build(int k,int L,int R) {
    lef[k]=L,rig[k]=R,lazy[k]=-1;
    int M=L+R>>1;
    if(L==R) {
        sum[k]=maxv[k]=init_val;
        secv[k]=-1;
        maxc[k]=1;
        return;
    }
    build(k<<1,L,M);
    build(k<<1|1,M+1,R);
    push_up(k);
}

void init(int n){
    init_val=n;
    build(1,1,n);
}
```

```

void update(int k,int L,int R,int v) {
    if(maxv[k]<=v) return;
    if(L<=lef[k]&&rig[k]<=R&&secv[k]<v) {
        return push_tag(k,v);
    }
    push_down(k);
    int mid=lef[k]+rig[k]>>1;
    if(mid>=L) update(k<<1,L,R,v);
    if(mid<R) update(k<<1|1,L,R,v);
    push_up(k);
}

```

```

} int p[MAXN]; LL f[MAXN]; vector<int> vec[MAXN]; void solve(){

```

```

int n=read_int();
_rep(i,1,n)
p[read_int()]=i;
_rep(i,1,n){
    vec[i].clear();
    vec[i].push_back(0);
    for(int j=i;j<=n;j+=i)
        vec[i].push_back(p[j]);
    sort(vec[i].begin(),vec[i].end());
}
Tree::init(n);
for(int i=n;i;i--){
    for(int j=0;j+2<vec[i].size();j++)
        Tree::update(1,vec[i][j]+1,vec[i][j+1],vec[i][j+2]-1);
    f[i]=1LL*n*n-Tree::sum[1];
}
f[n+1]=0;
_rep(i,1,n)
enter(f[i]-f[i+1]);

```

```

} int main(){ freopen("test.in","r",stdin); int T=read_int(); while(T--){ solve(); } return 0; }

```

From:

<https://wiki.cvbbacm.com/> - **CVBB ACM Team**

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:test



Last update: **2021/08/28 17:05**