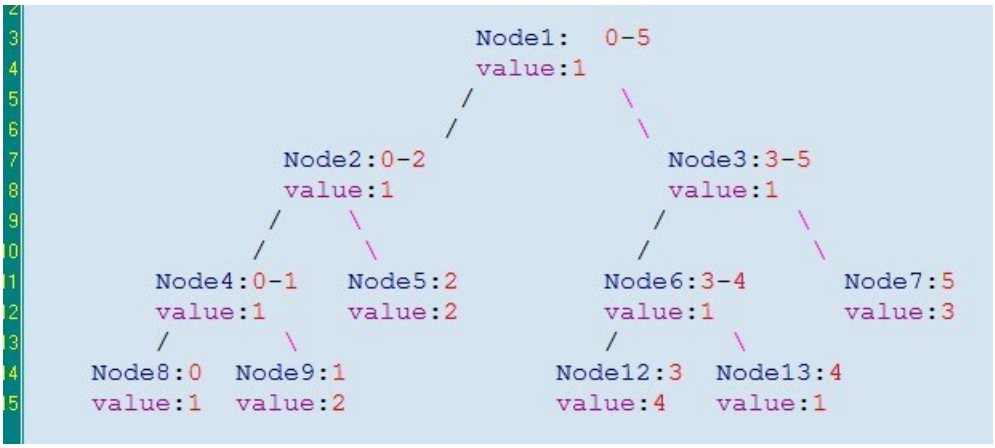
线段树

数据结构描述

线段树，类似区间树，是一个完全二叉树，它在各个节点保存一条线段（数组中的一段子数组），主要用于高效解决连续区间的动态查询问题，由于二叉结构的特性，它基本能保持每个操作的复杂度为 O(lgN)

性质 ：父亲的区间是 [a,b],(c=(a+b)/2) 左儿子的区间是 [a,c]，右儿子的区间是 [c+1,b]，线段树需要的空间为数组大小的四倍

图示



题目一

题目描述

士兵杀敌（一）

描述

南将军手下有 N 个士兵，分别编号 1 到 N，这些士兵的杀敌数都是已知的。

小工是南将军手下的军师，南将军现在想知道第 m 号到第 n 号士兵的总杀敌数，请你帮助小工来回答南将军吧。

注意，南将军可能会问很多次问题。

输入

只有一组测试数据

第一行是两个整数 N,M，其中 N 表示士兵的个数 (1<N<1000000)，M 表示南将军询问的次数 (1<M<100000)

随后的一行是 N 个整数，ai 表示第 i 号士兵杀敌数目。(0<=ai<=100)

随后的 M 行每行有两个整数 m,n，表示南将军想知道第 m 号到第 n 号士兵的总杀敌数（1<=m,n<=N)。

输出

对于每一个询问，输出总杀敌数 每个输出占一行

样例输入

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

样例输出

6 9

分析

这是一个简单哪的线段树的题只线段树的查询我们只要建立好线段树然后线段树查询就可以了

AC代码

```
#include <iostream>
#include <stdio.h>

#define MAX 1000010

using namespace std;

int array[MAX];
int segTree[MAX * 4 + 10];

void buildtree(int node,int begin,int end)
{
    if(begin==end)
    {
        segTree[node]=array[begin];
    }
    else
    {
        buildtree(node*2,begin,(end+begin)/2);
        buildtree(node*2+1,(end+begin)/2+1,end);
        segTree[node]=segTree[node*2]+segTree[node*2+1];

    }
}
int query(int node, int begin, int end, int left, int right)
{
    if(begin==left&&end==right)
    {
        return segTree[node];
    }
    int mid;
    mid=(begin+end)/2;
    if(right<=mid)
        return query(node * 2, begin, mid, left, right);
    else if(left>mid)
        return query(node*2+1,mid+1,end,left,right);
    else return query(node * 2, begin, mid, left, mid)+query(node*2+1,mid+1,end,mid + 1,right);
}
int main()
{
    int M,N;
    cin>>N>>M;
    for(int i=1;i<=N;i++)
        //cin>>array[i];
        scanf("%d",&array[i]);
    buildtree(1,1,N);
    int left,right;
    while(M--)
    {
        //cin>>left>>right;
        scanf("%d%d",&left,&right);
        //cout<<query(1,1,N,left,right)<<endl;
```

```
        printf("%d\n",query(1,1,N,left,right));
    }



    return 0;
}
```

题目二

题目描述

士兵杀敌（二）

描述

南将军手下有 N 个士兵，分别编号 1 到 N，这些士兵的杀敌数都是已知的。

小工是南将军手下的军师，南将军经常想知道第 m 号到第 n 号士兵的总杀敌数，请你帮助小工来回答南将军吧。

南将军的某次询问之后士兵 i 可能又杀敌 q 人，之后南将军再询问的时候，需要考虑到新增的杀敌数。

输入

只有一组测试数据

第一行是两个整数 N,M，其中 N 表示士兵的个数 (1<N<1000000)，M 表示指令的条数。(1<M<100000)

随后的一行是 N 个整数，ai 表示第 i 号士兵杀敌数目。(0<=ai<=100)

随后的 M 行每行是一条指令，这条指令包含了一个字符串和两个整数，首先是一个字符串，如果是字符串 QUERY 则表示南将军进行了查询操作，后面的两个整数 m,n，表示查询的起始与终止士兵编号；如果是字符串 ADD 则后面跟的两个整数 I,A(1<=I<=N,1<=A<=100), 表示第 I 个士兵新增杀敌数为 A.

输出

对于每次查询，输出一个整数 R 表示第 m 号士兵到第 n 号士兵的总杀敌数，每组输出占一行

样例输入

```
5 6
1 2 3 4 5
QUERY 1 3
ADD 1 2
QUERY 1 3
ADD 2 3
QUERY 1 2
QUERY 1 5
```

样例输出

```
6
8
8
20
```

分析

这一题是线段树的单点更新然后查询

AC代码

```
#include <stdio.h>
#include <string.h>

#define MAX 1000010

int array[MAX];
int segTree[MAX * 4 + 10];

void buildtree(int node,int begin,int end)
{
    if(begin==end)
    {
        segTree[node]=array[begin];
    }
    else
    {
        int mid = (end+begin)/2;
        buildtree(node*2,begin,mid);
        buildtree(node*2+1,mid+1,end);
        segTree[node]=segTree[node*2]+segTree[node*2+1];

    }
}
int query(int node, int begin, int end, int left, int right)
{
    if(begin==left&&end==right)
    {
        return segTree[node];
    }
    int mid;
    mid=(begin+end)/2;
    if(right<=mid)
        return query(node * 2, begin, mid, left, right);
    else if(left>mid)
        return query(node*2+1,mid+1,end,left,right);
    else return query(node * 2, begin, mid, left, mid)+query(node*2+1,mid+1,end,mid + 1,right);
}
void  addkill(int node,int begin,int end,int add,int i)
{
    if(i>=begin&&i<=end)

    {
        if(begin==end)
    {
        if(begin==i)
        segTree[node]=segTree[node]+add;
    }
    else
    {
        addkill(node*2,begin,(end+begin)/2,add,i);
        addkill(node*2+1,(end+begin)/2+1,end,add,i);
        segTree[node]=segTree[node*2]+segTree[node*2+1];
    }
    }


    return;
}
int main()
{
```

```
    int M,N;
    scanf("%d%d",&N,&M);
    for(int i=1;i<=N;i++)
        scanf("%d",&array[i]);
    buildtree(1,1,N);
    int left,right;
    char instruction[10];
    while(M--)
    {

        scanf("%s %d%d",instruction,&left,&right);
        if(instruction[0]=='A')
            {
                addkill(1,1,N,right,left);

            }
        else
        {
            printf("%d\n",query(1,1,N,left,right));
        }

    }
    return 0;
}
```

题目三

题目描述

士兵杀敌（三）

描述

南将军统率着 N 个士兵，士兵分别编号为 1~N, 南将军经常爱拿某一段编号内杀敌数最高的人与杀敌数最低的人进行比较，计算出两个人的杀敌数差值，用这种方法一方面能鼓舞杀敌数高的人，另一方面也算是批评杀敌数低的人，起到了很好的效果。

所以，南将军经常问军师小工第 i 号士兵到第 j 号士兵中，杀敌数最高的人与杀敌数最低的人之间军功差值是多少。

现在，请你写一个程序，帮小工回答南将军每次的询问吧。

注意，南将军可能询问很多次。

输入

只有一组测试数据

第一行是两个整数 N,Q，其中 N 表示士兵的总数。Q 表示南将军询问的次数。(1<N<=100000,1<Q<=1000000)

随后的一行有 N 个整数 Vi(0<=Vi<100000000)，分别表示每个人的杀敌数。

再之后的 Q 行，每行有两个正正数 m,n，表示南将军询问的是第 m 号士兵到第 n 号士兵。

输出

对于每次询问，输出第 m 号士兵到第 n 号士兵之间所有士兵杀敌数的最大值与最小值的差。

样例输入

```
5 2
1 2 6 9 3
1 2
2 4
```

样例输出

1 7

分析

这一题就是一个线段树的建树问题,然后查询

AC代码

```
#include<bits/stdc++.h>

#define MAX 1000010

using namespace std;
struct node1
{
    int ma;
    int mi;
}segtree[MAX*4+10];
int array1[MAX];
void buildtree(int node ,int begin, int end)
{
    if(begin==end)
    {
        segtree[node].ma=array1[begin];
        segtree[node].mi=array1[end];
        return;
    }
    else
    {
        buildtree(node*2,begin,(end+begin)/2);
        buildtree(node*2+1,(end+begin)/2+1,end);
        segtree[node].ma=max(segtree[2*node].ma,segtree[2*node+1].ma);
        segtree[node].mi=min(segtree[2*node].mi,segtree[2*node+1].mi);
    }


}
struct node1 query(int node, int begin, int end, int left, int right)
{
    if(begin==left&&end==right)
    {
        return segtree[node];
    }
    int mid;
    mid=(begin+end)/2;
    if(right<=mid)
        return query(node * 2, begin, mid, left, right);
    else if(left>mid)
        return query(node*2+1,mid+1,end,left,right);
    //else return query(node * 2, begin, mid, left, mid)+query(node*2+1,mid+1,end,mid + 1,right);
    else
    {
        struct node1 tmp1 = query(node * 2, begin, mid, left, mid);
        struct node1 tmp2 = query(node*2+1,mid+1,end,mid + 1,right);
        tmp1.ma = max(tmp1.ma,tmp2.ma);
        tmp1.mi = min(tmp1.mi,tmp2.mi);
    //cout<<tmp1.mi<<"  "<<tmp1.ma<<endl;
        return tmp1;

    }
```

```
}
int main()
{
    int M,N;
    cin>>N>>M;
    for(int i=1;i<=N;i++)
        scanf("%d",&array1[i]);
    buildtree(1,1,N);
// for(int i=0;i<=10;i++)
    // cout<<segtree[i].ma<<"   "<<segtree[i].mi<<endl;
    int left,right;
    while(M--)
    {
        scanf("%d%d",&left,&right);
        struct node1 tmp = query(1,1,N,left,right);
        printf("%d\n",tmp.ma-tmp.mi);
    }



    return 0;
}
```

结语

这是我对于线段树问题的一个总结，其实就是用空间来换取时间，线段树的段更新没有总结