**一: 引例**

|  |
| --- |
| 题目一： 10000个正整数，编号1到10000，用A[1],A[2],A[10000]表示。 修改：无 统计：1.编号从L到R的所有数之和为多少？ 其中1<= L <= R <= 10000. |

方法一：对于统计L,R ，需要求下标从L到R的所有数的和，从L到R的所有下标记做[L..R],问题就是对A[L..R]进行求和。

这样求和，对于每个询问，需要将(R-L+1)个数相加。

方法二：更快的方法是求前缀和,令 S[0]=0, S[k]=A[1..k] ，那么，A[L..R]的和就等于S[R]-S[L-1]，

这样，对于每个询问，就只需要做一次减法，大大提高效率。

|  |
| --- |
| 题目二： 10000个正整数，编号从1到10000，用A[1],A[2],A[10000]表示。 修改：1.将第L个数增加C （1 <= L <= 10000） 统计：1.编号从L到R的所有数之和为多少？ 其中1<= L <= R <= 10000. |

再使用方法二的话，假如A[L]+=C之后，S[L],S[L+1],,S[R]都需要增加C,全部都要修改，见下表。

|  |  |  |
| --- | --- | --- |
|  | 方法一 | 方法二 |
| A[L]+=C | 修改1个元素 | 修改R-L+1个元素 |
| 求和A[L..R] | 计算R-L+1个元素之和 | 计算两个元素之差 |

从上表可以看出，方法一修改快，求和慢。 方法二求和快，修改慢。

那有没有一种结构，修改和求和都比较快呢？答案当然是线段树。

**二: 综述**

假设有编号从1到n的n个点，每个点都存了一些信息，用[L,R]表示下标从L到R的这些点。

线段树的用处就是，对编号连续的一些点进行修改或者统计操作，修改和统计的复杂度都是O(log2(n)).

线段树的原理，就是，将[1,n]分解成若干特定的子区间(数量不超过4\*n),然后，将每个区间[L,R]都分解为

少量特定的子区间，通过对这些少量子区间的修改或者统计，来实现快速对[L,R]的修改或者统计。

由此看出，用线段树统计的东西，必须符合**区间加法**，否则，不可能通过分成的子区间来得到[L,R]的统计结果。

**符合区间加法的例子：**

数字之和——总数字之和 = 左区间数字之和 + 右区间数字之和

最大公因数(GCD)——总GCD = gcd( 左区间GCD , 右区间GCD );

最大值——总最大值=max(左区间最大值，右区间最大值)

**三：原理**

**（注：由于线段树的每个节点代表一个区间，以下叙述中不区分节点和区间，只是根据语境需要，选择合适的词）**

线段树本质上是维护下标为1,2,..,n的n个按顺序排列的数的信息，所以，其实是“点树”，是维护n的点的信息，至于每个点的数据的含义可以有很多，

在对线段操作的线段树中，每个点代表一条线段，在用线段树维护数列信息的时候，每个点代表一个数，但本质上都是每个点代表一个数。以下，在讨论线段树的时候，区间[L,R]指的是下标从L到R的这(R-L+1)个数，而不是指一条连续的线段。只是有时候这些数代表实际上一条线段的统计结果而已。

线段树是将每个区间[L,R]分解成[L,M]和[M+1,R] (其中M=(L+R)/2 这里的除法是整数除法，即对结果下取整)直到 L==R 为止。

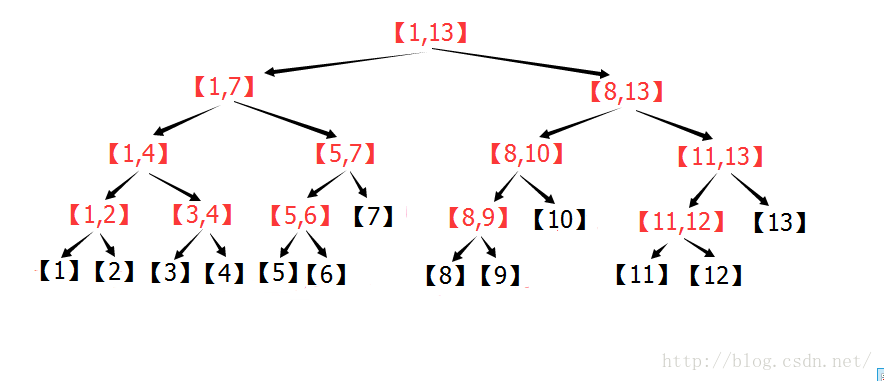
开始时是区间[1,n] ,通过递归来逐步分解，假设根的高度为1的话，树的最大高度为（n>1）。

线段树对于每个n的分解是唯一的，所以n相同的线段树结构相同，这也是实现可持久化线段树的基础。

下图展示了区间[1,13]的分解过程：

上图中，每个区间都是一个节点，每个节点存自己对应的区间的统计信息。

**四: 区间划分与线段树的建立**



void build(int l,int r,int cur)

{

int m;

sum[cur]=0,laz[cur]=0;

if(l==r) return;

m=(l+r)/2;

build(l,m,2\*cur);

build(m+1,r,2\*cur+1);

pushup(cur);

}

**五: pushup与pushdown**

void pushup(int cur)

{

sum[cur]=sum[2\*cur]+sum[2\*cur+1];

}

void pushdown(int l,int r,int cur)

{

int m;

if(laz[cur]!=0)

{

m=(l+r)/2;

sum[2\*cur]+=(m-l+1)\*laz[cur];

laz[2\*cur]+=laz[cur];

sum[2\*cur+1]+=(r-m)\*laz[cur];

laz[2\*cur+1]+=laz[cur];

laz[cur]=0;

}

}

1. pushup与pushdown是必须的吗?
2. 多种lazy标记的次序

**六: 基本点与区间操作模板**

void updateI(int tar,int val,int l,int r,int cur)//点更新

{

int m;

if(l==r)

{

sum[cur]+=val;

laz[cur]+=val;

return;

}

m=(l+r)/2;

if(tar<=m) updateI(tar,val,l,m,2\*cur);

else updateI(tar,val,m+1,r,2\*cur+1);

pushup(cur);

}

void updateII(int pl,int pr,int val,int l,int r,int cur)//点查询

{

int m;

if(pl<=l&&r<=pr)

{

sum[cur]+=(r-l)\*val;

laz[cur]+=val;

return;

}

pushdown(cur,l,r);

m=(l+r)/2;

if(pl<=m) updateII(pl,pr,val,l,m,2\*cur);

if(pr>m) updateII(pl,pr,val,m+1,r,2\*cur+1);

pushup(cur);

}

int queryI(int tar,int l,int r,int cur)//点查询

{

int m,res;

if(l==r) return sum[cur];

m=(l+r)/2,res=0;

pushdown(cur,l,r);

if(tar<=m) return queryI(tar,l,m,2\*cur);

else queryI(tar,m+1,r,2\*cur+1);

}

int queryII(int pl,int pr,int l,int r,int cur)//区间查询

{

int m,res;

if(pl<=l&&r<=pr) return sum[cur];

pushdown(cur,l,r);

m=(l+r)/2,res=0;

if(pl<=m) res+=queryII(pl,pr,l,m,2\*cur);

if(pr>m) res+=queryII(pl,pr,m+1,r,2\*cur+1);

}

**树状数组**

**int lowbit(int x)**

**{**

**return x&-x;**

**}**

**void update(int pos,int val)**

**{**

**while(pos<=n)**

**{**

**c[pos]+=val;**

**pos+=lowbit(pos);**

**}**

**}**

**int query(int pos)**

**{**

**int ans=0;**

**while(pos>0)**

**{**

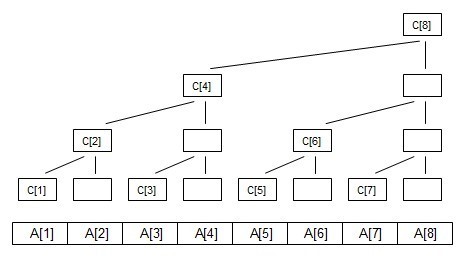
**ans+=c[pos];**

**pos-=lowbit(pos);**

**}**

**return ans;**

**}///单点修改区间查询，最基本最常用**



C[1]=A[1];

C[2]=A[1]+A[2];

C[3]=A[3];

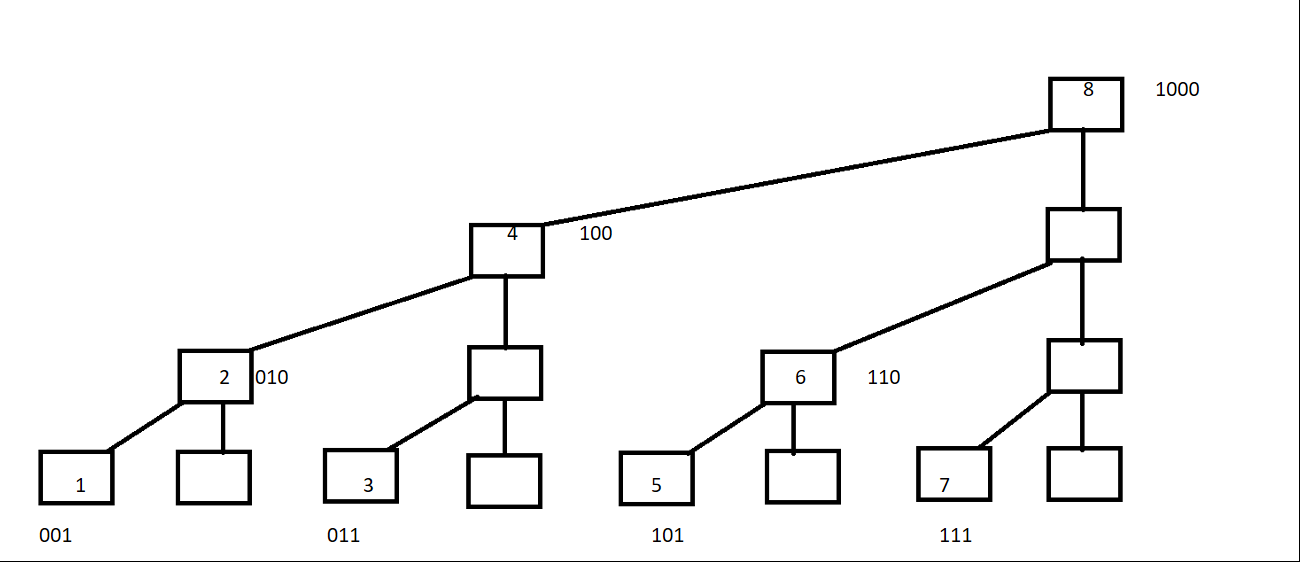
C[4]=A[1]+A[2]+A[3]+A[4];

C[5]=A[5];

C[6]=A[5]+A[6];

C[7]=A[7];

C[8]=A[1]+A[2]+A[3]+A[4]+A[5]+A[6]+A[7]+A[8];



**Lowbit(x)**

**Update(pos,val)**

**Query(pos)**

**二维**

**int lowbit(int x) {return x&(-x);}**

**int update(int x,int y,int val)**

**{**

**for(int i=x;i<=N;i+=lowbit(i))**

**for(int j=y;j<=N;j+=lowbit(j))**

**sum[i][j]+=val;**

**}**

**int query(int x,int y)**

**{**

**int ans=0;**

**for(int i=x;i>0;i-=lowbit(i))**

**for(int j=y;j>0;j-=lowbit(j))**

**ans+=sum[i][j];**

**return ans;**

**}**