# 树状数组

树状数组实际上是一种维护区间的思想.

它所维护的不一定是区间和，凡是可以“累加”或是“合并”的区间性质都可以被维护

在一些“区间上是否存在xx”且频繁查询的场景也可能用到。在这种情况下甚至可以在小区间上使用蛮力

注意宏定义，树状数组大小给到原数组logn倍

```
#define root 1,n,1
#define lson l,mid,rt<<1
#define rson mid+1,r,rt<<1|1
#define snow int l,int r,int rt
using namespace std;
struct szsz{
    int sum;
    szsz()
    {
        sum=0;
    }
};
struct szsz z[2000005];
int a[500005];

void update(int rt)
{
    z[rt].sum=z[rt<<1].sum+z[rt<<1|1].sum;//合并
}

//建立
void build(snow)
{
    int mid=(l+r)>>1;

    if(l==r) {  z[rt].sum=a[l]; return;}

    build(lson);
    build(rson);
    update(rt);
}

//区间合并不止于加法。只要满足分配律、或者说结果与合并顺序无关都可以
//例如三角形的题你可以把合并定义为“区间上有无三角形”，这样可以加速已经确定了有三角形区间的合并

//查询[l,r],sum初始为0,tl,tr表示要查询的区间
int se(snow,int tl,int tr,int sum)//t:target
{
    int mid=(l+r)>>1;
    if(tl<=l&&r<=tr) return sum+z[rt].sum;//区间合并
```

```
    if(tl<=mid)
    {
        if(mid<tr) return sum+se(lson,tl,tr,sum)+se(rson,tl,tr,sum);//区间合并
        else return sum+se(lson,tl,tr,sum);//区间合并
    }
    else return sum+se(rson,tl,tr,sum);//区间合并
}

//修改t->v
void modify(snow,int t,int v)
{
    if(l==r) {z[rt].sum+=v; return;}
    int mid=(l+r)>>1;
    if(t>mid) modify(rson,t,v);
    else modify(lson,t,v);
    update(rt);
    return;
}
```