线段树 Segment Tree

维护区间信息的数据结构

杨文昊

ACM-ICPC 省铜、校一

2020 年 11 月 16 日

区间信息

- 线段树是算法竞赛中非常常用的维护区间信息的数据结构，与它作用相似的还有树状数组。
- 线段树可以在 $O(logN)$ 的时间复杂度内实现单点修改、区间修改、区间查询（区间求和，求区间最大值，求区间最小值）等操作。
- 线段树维护的信息，需要满足可加性，即能以可以接受的速度合并信息和修改信息，包括在使用懒惰标记时，标记也要满足可加性。

线段树的基本结构

- 线段树将每个长度不为 1 的区间划分成左右两个区间递归求解，把整个线段划分为一个树形结构，通过合并左右两区间信息来求得该区间的信息。这种数据结构可以方便的进行大部分的区间操作。有大小为 5 的数组，$a = \{10, 11, 12, 13, 14\}$，现在我们需要把它转化为线段树。

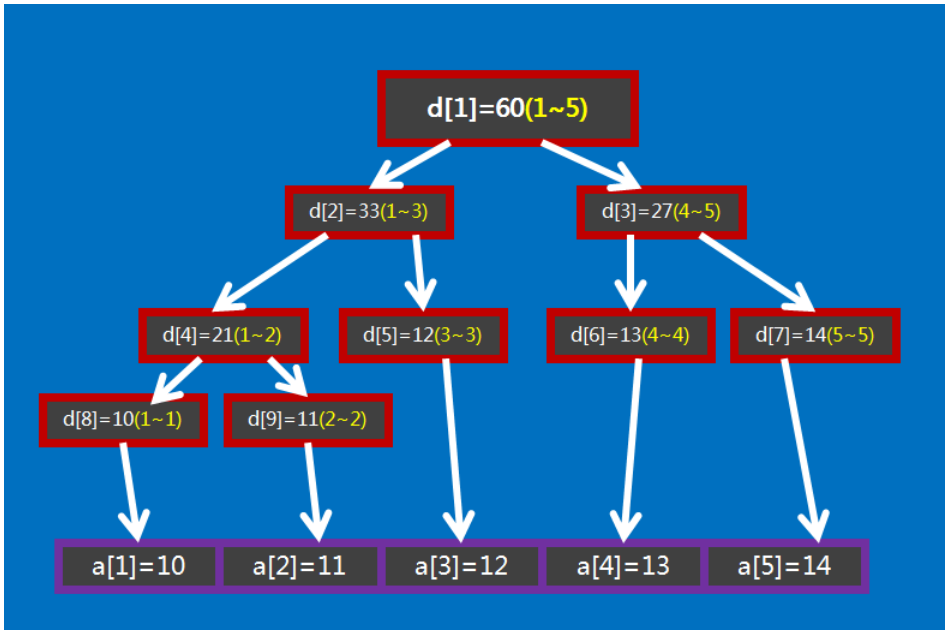

图: 线段树基本结构

维护区间的数组 d

- 数组 d 维护指定区间的所有数之和，如图所示，$d[1]$ 维护的区间为 $[1,5]$，即 $d1 = 10+11+12+13+14 = 60$，而左结点 $d[2]$ 维护区间为 $[1,3]$，而右结点 $d[3]$ 维护区间为 $[4,5]$。以此类推，如果 $d[i]$ 维护区间为 $[s,t]$，那么左结点维护区间为 $[s, \frac{s+t}{2}]$，右结点维护区间为 $[\frac{s+t}{2}+1, t]$。

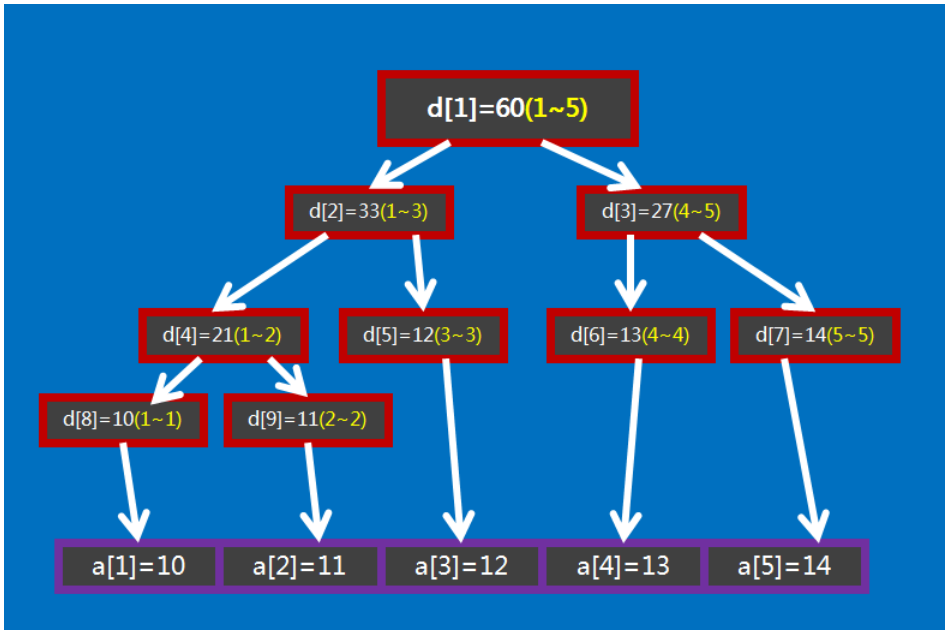

图: 线段树基本结构

建立线段树

- 很显然，我们需要利用递归的思想去建立我们想要的线段树，那对于这个递归问题，我们如何去寻找它所对应的递归边界呢？即当 $d[i]$ 所维护的区间 $[s, t]$ 满足条件 $s = t$ 时，则不可继续分解区间和，我们把这个作为递归边界。

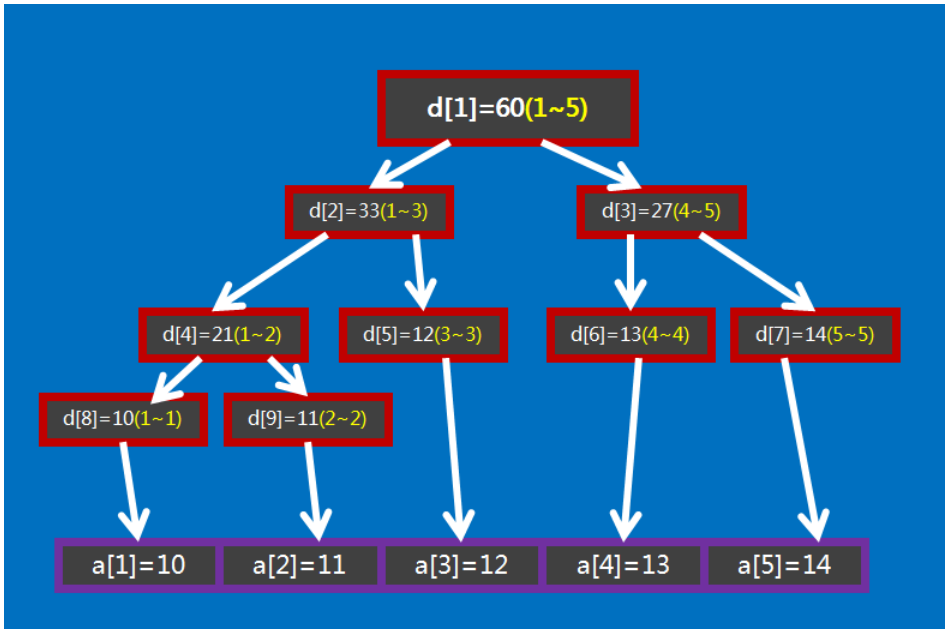

图: 线段树基本结构

C++ 代码

```
void build(int s,int t,int p)
{
    if (s == t)
    {
        d[p] = a[s];
        return;
    }
    else
    {
        int m = (s + t) / 2;
        build(s, m, p * 2);
        build(m + 1, t, p * 2 + 1);
        d[p] = d[p * 2] + d[p * 2 + 1];
    }
    }
```

图: C++ 代码实现

线段树区间求和

- 如果我们需要求出区间 $[3,5]$ 的和，根据线段树的基本结构，我们可以把这个区间拆分为 $[3,3]d[5]+[4,5]d[3]$ 来求出区间和。

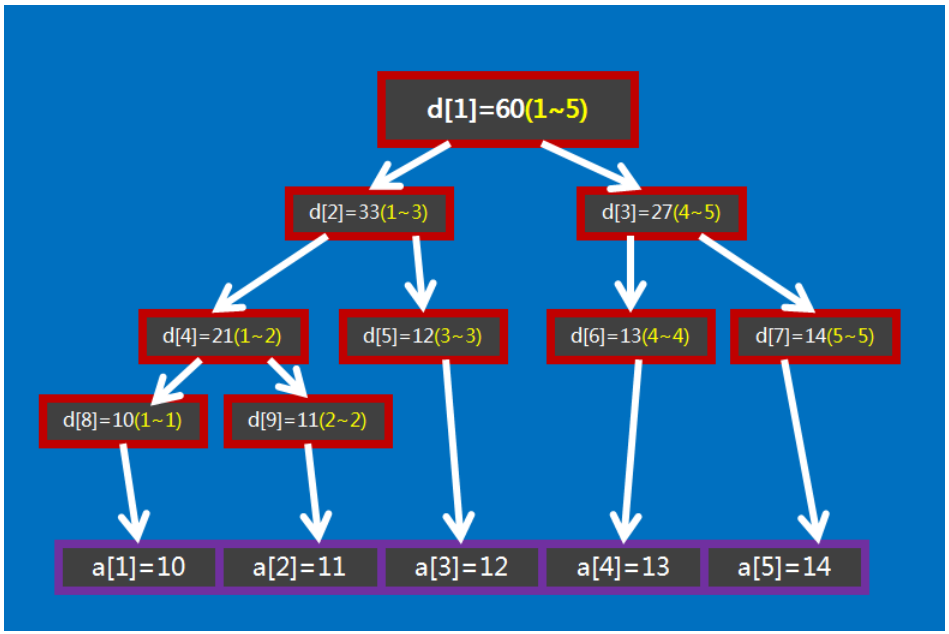

图: 线段树基本结构

```cpp
ll getsum(int l, int r, int s, int t, int p)
{
    if (l <= s && r >= t) return d[p];
    int m = (s + t) / 2;
    ll sum = 0;
    if (l <= m) sum += getsum(l,r,s,m,p*2);
    if (r > m) sum += getsum(l,r,m+1,t,p*2+1);
    return sum;
    }
```

图: C++ 代码实现

懒惰标记

- 如果要求修改区间 $[l, r]$，把所有包含在区间 $[l, r]$ 中的节点都遍历一次、修改一次，时间复杂度无法承受。我们这里要引入一个叫做"懒惰标记"的东西。
- 懒惰标记是个非常有意思的东西，如果现在我们想让区间 $[3, 5]$ 每个数都加上 2，我们将会如何操作?

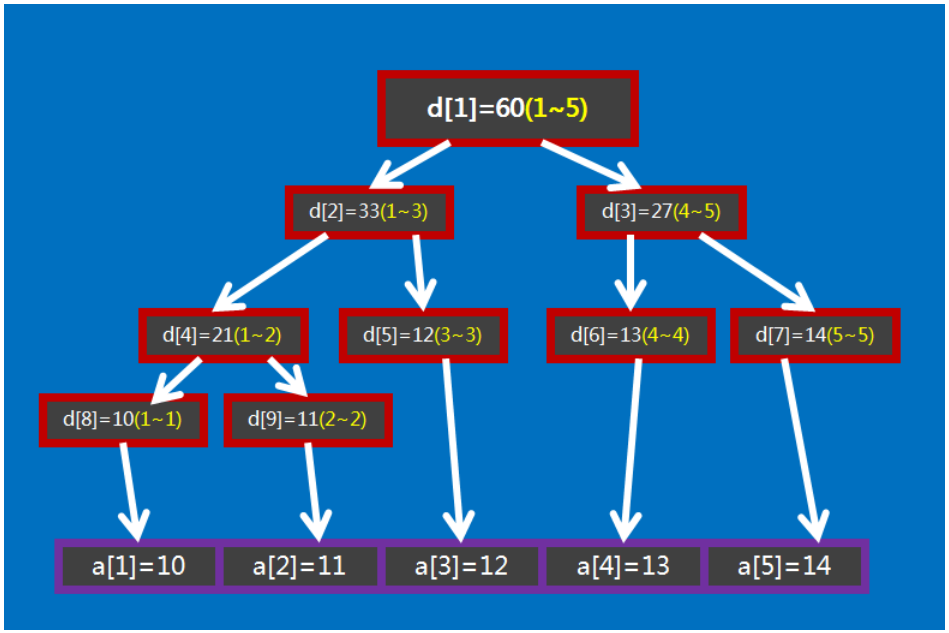

图: 线段树基本结构

C++ 代码实现

```
void update(int l, int r, int c, int s, int t, int p)
{
    if (l <= s && r >= t)
    {
        d[p] += (t - s + 1) * c;
        b[p] += c;
        return;
    }
    int m = (s + t) / 2;
    if (b[p] && s != t)
    {
        d[p * 2] += b[p] * (m - s + 1);
        d[p * 2 + 1] += b[p] * (t - m);
        b[p * 2] += b[p];
        b[p * 2 + 1] += b[p];
        b[p] = 0;
    }
    if (l <= m) update(l, r, c, s, m, p * 2);
    if (r > m) update(l, r, c, m + 1, t, p * 2 + 1);
    d[p] = d[p * 2] + d[p * 2 + 1];
    }
```

图: C++ 代码实现区间修改

区间修改成某个特定值

```
void update(int l, int r, int c, int s, int t, int p)
{
    if (l <= s && r >= t)
    {
        d[p] = (t - s + 1) * c;
        b[p] = c;
        return;
    }
    int m = (s + t) / 2;
    if (b[p] && s != t)
    {
        d[p * 2] = b[p] * (m - s + 1);
        d[p * 2 + 1] = b[p] * (t - m);
        b[p * 2] = b[p];
        b[p * 2 + 1] = b[p];
        b[p] = 0;
    }
    if (l <= m) update(l, r, c, s, m, p * 2);
    if (r > m) update(l, r, c, m + 1, t, p * 2 + 1);
    d[p] = d[p * 2] + d[p * 2 + 1];
    }
```

图: C++ 代码实现区间修改 2

线段树区间求和 (新)

```cpp
ll getsum(int l, int r, int s, int t, int p)
{
    if (l <= s && r >= t) return d[p];
    int m = (s + t) / 2;
    if (b[p] && s != t)
    {
        d[p * 2] += b[p] * (m - s + 1);
        d[p * 2 + 1] += b[p] * (t - m);
        b[p * 2] += b[p];
        b[p * 2 + 1] += b[p];
        b[p] = 0;
    }
    ll sum = 0;
    if (l <= m) sum += getsum(l,r,s,m,p*2);
    if (r > m) sum += getsum(l,r,m+1,t,p*2+1);
    return sum;
}
```

图: C++ 代码实现区间更新后的求和

一些优化

- 下传懒惰标记，pushdown() 函数
- 堆式建树
- 猫树、李超线段树

模板题

- P3372 【模板】线段树 1
- P3373 【模板】线段树 2

P1471 方差

根据方差的公式来进行推导

$$
\begin{aligned}
s^2 &= \frac{1}{n}\sum_{i=1}^{n}(a_i-\bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}(a_i^2-2a_i\bar{x}-\bar{x}^2) \\
&= \frac{1}{n}(\sum_{i=1}^{n}a_i^2-2\bar{x}\sum_{i=1}^{n}a_i+n\bar{x}^2) \\
&= \frac{1}{n}(\sum_{i=1}^{n}a_i^2-\frac{(\sum_{i=1}^{n}a_i)^2}{n}) = \frac{1}{n}\sum_{i=1}^{n}a_i^2-\frac{(\sum_{i=1}^{n}a_i)^2}{n^2} \\
&= \frac{1}{n}\sum_{i=1}^{n}a_i^2-\bar{x}^2
\end{aligned}
\tag{1}
$$

2020ACM-ICPC 陕西赛区 B 题

- n 个杯子，装有不同容量的水
- 在 n 杯水中，选两个杯子，将剩余 n-2 的杯子中水进行倒出或倒入操作，让它们的水等于两个杯子中水的任意一杯。
- 求出使倒出倒入水量最小的选择。
- 例如：1 2 2 3 3 5

机器学习中的线段树

- KD-Tree(K-Dimension Tree) 是一种对 k 维空间中的实例点进行存储以便对其进行快速检索的树形数据结构。
- 自动驾驶中实施三维重建 (SLAM) 的问题，所常用的深感摄像头采集的是环境的点云信息，而这些庞大的点云信息为了能优化我们搜索的速度，我们通常采用 KD-Tree 类似的方式去存储。

Q&A