线段树的区间修改（二）

主讲人：邓哲也

改编1

问题：有一个长度为 n 的序列，a[1]，a[2]，…，a[n]。现在执行 m 次操作，每次可以执行以下两种操作之一：

1. 将下标在区间［l，r］的数都加上 v。
2. 询问一个下标区间［l，r］中所有数的和。

改编1

和区间修改成一个数类似，我们只需要改变 tag 的定义：

tag 表示当前节点代表的区间每个数都要加上的值。

改编1-标记下传

标记下传函数：

```
void down(int l, int r, int x){
        int mid = (l + r) >> 1;
        if (tag[x] != 0){
                tag[ls] += tag[x];
                tag[rs] += tag[x];
                sum[ls] += (mid - l + 1) * tag[x];
                sum[rs] += (r - mid) * tag[x];
                tag[x] = 0;
        }
}
```

改编1-区间加

对[A, B]中的每个数都加上 v

```
void add(int A, int B, int v, int l, int r, int x) {
        if (A <= l && r <= B) {
                tag[x] += v;
                sum[x] += v * (r - l + 1);
                return;
        }
        down(l, r, x);                  // 在继续修改之前，先检查是否要下传标记
        int mid = (l + r) >> 1;
        if (A <= mid) add(A, B, v, l, mid, ls);
        if (mid < B)          add(A, B, v, mid + 1, r, rs);
        update(x);              // 回溯的时候要更新每个节点的sum，因为子节点的值
改变了
}
```

改编2

问题：有一个长度为 n 的序列，a[1]，a[2]，…，a[n]。现在执行 m 次操作，每次可以执行以下两种操作之一：

1. 将下标在区间［l，r］的数都加上 v。
2. 询问一个下标区间［l，r］中所有数的最小值。

改编2

和区间修改成一个数类似，我们只需要改变 tag 的定义：

tag 表示当前节点代表的区间每个数都要加上的值。

Min 表示当前节点代表的区间的最小值。

改编2-上传和下传函数

更新信息函数：

```
void update(int x){
	Min[x] = min(Min[ls], Min[rs]);
}
```

标记下传函数：

```
void down(int l, int r, int x){
	int mid = (l + r) >> 1;
	if (tag[x] != 0){
		tag[ls] += tag[x];
		tag[rs] += tag[x];
		Min[ls] += tag[x];
		Min[rs] += tag[x];
		tag[x] = 0;
	}
}
```

改编2-区间加

对[A, B]中的每个数都加上 v

```
void add(int A, int B, int v, int l, int r, int x) {
        if (A <= l && r <= B) {
                tag[x] += v;
                Min[x] += v;
                return;
        }
        down(l, r, x);                  // 在继续修改之前，先检查是否要下传标记
        int mid = (l + r) >> 1;
        if (A <= mid) add(A, B, v, l, mid, ls);
        if (mid < B)          add(A, B, v, mid + 1, r, rs);
        update(x);              // 回溯的时候要更新每个节点的sum，因为子节点的值
改变了
}
```

改编2-区间查询

查询[A, B]的最小值

```
int query(int A, int B, int l, int r, int x) {
        if (A <= l && r <= B)
                return Min[x];
        down(l, r, x);                  // 在继续查询之前，先检查是否要下传标记
        int mid = (l + r) >> 1, ret = 0x3F3F3F3F;
        if (A <= mid) ret = min(ret, query(A, B, l, mid, ls));
        if (mid < B)  ret = min(ret, query(A, B, mid + 1, r, rs));
        return ret;
}
```

改编3

问题：有一个长度为 n 的序列，a[1]，a[2]，…，a[n]。现在执行 m 次操作，每次可以执行以下两种操作之一：

1. 将下标在区间［l，r］的数都加上 v。
2. 询问一个下标区间［l，r］中所有数的最小值的个数。

改编3

和区间修改成一个数类似，我们只需要改变 tag 的定义：

tag 表示当前节点代表的区间每个数都要加上的值。

Min 表示当前节点代表的区间的最小值。

cnt 表示当前节点代表的区间的最小值的个数。

改编3-上传函数

更新信息函数：

```
void update(int x){
        Min[x] = min(Min[ls], Min[rs]);
        if (Min[ls] == Min[rs]){
                Min[x] = Min[ls];
                cnt[x] = cnt[ls] + cnt[rs];
        } else {
                if (Min[ls] < Min[rs])
                        Min[x] = Min[ls], cnt[x] = cnt[ls];
                else
                        Min[x] = Min[rs], cnt[x] = cnt[rs];
        }
}
```

改编3-下传函数

标记下传函数：// 此处 cnt 不用改变

```
void down(int l, int r, int x){
        int mid = (l + r) >> 1;
        if (tag[x] != 0){
                tag[ls] += tag[x];
                tag[rs] += tag[x];
                Min[ls] += tag[x];
                Min[rs] += tag[x];
                tag[x] = 0;
        }
}
```

改编3-区间加

对[A, B]中的每个数都加上 v

```
void add(int A, int B, int v, int l, int r, int x) {
        if (A <= l && r <= B) {
                tag[x] += v;
                Min[x] += v;
                return;
        }
        down(l, r, x);                  // 在继续修改之前，先检查是否要下传标记
        int mid = (l + r) >> 1;
        if (A <= mid) add(A, B, v, l, mid, ls);
        if (mid < B)          add(A, B, v, mid + 1, r, rs);
        update(x);              // 回溯的时候要更新每个节点的sum，因为子节点的值
改变了
}
```

改编3-区间查询

查询[A, B]的最小值

```
pair<int,int> query(int A, int B, int l, int r, int x) {
        if (A <= l && r <= B)
                return make_pair(Min[x], cnt[x]);
        down(l, r, x);                    // 在继续查询之前，先检查是否要下传标记
        int mid = (l + r) >> 1, ret = make_pair(0x3F3F3F3F, 0);
        if (A <= mid) ret = min(ret, query(A, B, l, mid, ls));
        if (mid < B){
                pair<int,int> tmp = min(ret, query(A, B, mid + 1, r, rs));
                if (tmp.first == ret.first) ret.second += tmp.second;
                else if(tmp.first < ret.first) ret = tmp;
        }
        return ret;
}
```

线段树总结

如果要用线段树维护一个数据结构，一定要想清楚怎么实现update函数和down函数。

也就是如何合并两个子树的信息，如何解决标记下传。

解决了这两个问题，许多问题就可以引刃而解了。

下节课再见