算法基础 HW8

PB20111686 黄瑞轩

1

若 $n=2^k, k\geq 0$，则 $a^{2^k}=a^{2^{k-1}}\cdot a^{2^{k-1}}$，即 $T(n)=T(\frac{n}{2})+O(1)$，即 $T(n)=O(\log n)$

若 $n\neq 2^k$，假设 $n=a_0+a_1\cdot 2^1+...+a_m\cdot 2^m$，这里 $a_i\in\mathbb{Z}_2, 0\leq i\leq m$，则

$$T(n)=\sum_{i=0}^{m}T(a_i\cdot 2^i)+O(m)\leq\sum_{i=0}^{m}T(2^i)+O(m) \tag{1}$$

> 不等号右侧就是最坏情况，此时恰好为 $n=2^k-1, k>0$

首先 $n\leq 2^{\lceil\log n\rceil}$，所以 $O(m)=O(\log n)$。

若我们从 $T(2^0)$ 开始进行计算任务，假设目前已经算完 $T(2^i)$，则 $T(2^{i+1})=O(1)$，因为在计算 $T(2^{i+1})$ 时 $T(2^i)$ 已经算过，所需时间为 $O(1)$（一次乘法即可算完），所以 $\sum_{i=0}^{m}T(2^i)=O(\log n)$，即 $T(n)=O(\log n)$。

2

> **示例分析**：首先，对 A 和 B 分别查询第 $\frac{1}{2}n$ 小的数，如果 $A[\frac{1}{2}n]<B[\frac{1}{2}n]$，则中位数只可能出现在 $A[\frac{1}{2}n\sim n]$ 和 $B[0\sim\frac{1}{2}n]$ 之间，问题就转化为了在这两个范围内找第 n 小的数，直到游标相等，返回。可以通过向函数传递 A、B 数组的区间上下界来完成。
>
> **两个数组大小相等使得上述算法正确性得以保证**

```
int get(Database& D, int k); // 题目提供的算法

int find(int A_L, int A_R, int B_L, int B_R) {
    int A_piviot = (A_L + A_R) >> 1;
    int B_piviot = (B_L + B_R) >> 1;
    int _test_A = get(A, A_piviot);
    int _test_B = get(B, B_piviot);
    if (_test_A == _test_B) return _test_A;
    else if (_test_A < _test_B) return find(A_piviot, A_R, B_L, B_piviot);
    else return find(A_L, A_piviot, B_piviot, B_R);
}
```

设算法时间复杂度是 $T(n)$，则

$$T(n)\leq T(\frac{1}{2}n)+O(1) \tag{2}$$

所以 $T(n)=O(\log n)$。

3

算法按下列规则运行：

1. 令 $i=1$
2. 向 path 1 ~ path m 离开汇聚点向前走至多 2^i 距离，如果这期间在某条路上找到了宝藏则算法终止，如果距离达到 2^i，则直接返回汇聚点，这一步的路程最大值为 $m\cdot 2^i\cdot 2=m\cdot 2^{i+1}$

3. $i = i + 1$，转 2

上面算法一定能得到结果，因为每条路最坏情况下都会走无穷远的距离，所有路的所有距离都会被达到（在找到宝藏之前）。

上面假设宝藏位置确定但未知，在 Offline 情况下，已知宝藏在 path k 上距离为 r 处，则 $\text{Offline OPT} = r$，而 ALG 的最大值由如下思路计算：

因为上面算法按顺序从 path 1 ~ path m 寻路，不妨假设宝藏在 path m 上，并且 $r = 2^k - 1$，这样在找到宝藏前在 path m 上就需要走

$$\sum_{i=0}^{k-1} 2 \cdot 2^i + 2^k - 1 = 2r - 1 \tag{3}$$

距离，在其他路上需要走

$$\sum_{i=0}^{k} 2 \cdot 2^i = 4r \tag{4}$$

距离，则

$$\max \text{ALG} = (m-1) \cdot 4r + 2r - 1 = (4m-2)r - 1 \tag{5}$$

竞争比为

$$ratio \leq \frac{\max \text{ALG}}{\text{Offline OPT}} = \frac{(4m-2)r-1}{r} = O(m) \tag{6}$$