Alg HW8

林宸昊 PB20000034

1. a. $a^n = \begin{cases} a^{n/2} * a^{n/2}, & n\text{为偶数} \\ a^{n-1/2} * a^{n-1/2} * a, & n\text{为奇数} \end{cases}$ 经过多重拆分子问题最终变为

计算 $a*a$ 或 $a*a*a$，最坏运行时间为常量时间 $O(1)$.

(其中a为传入参数而非最初的a)

b. 合并时即将两个分立的子问题得到的结果相乘，并将得到的结果向上传递

如 $a^{n/2} * a^{n/2}$ → $a^{n/4} * a^{n/4}$ 和 $a^{n/4} * a^{n/4}$

由于每一次划分就需要一次合并，

共需 k 次划分到达 a^1，$2^k = n \Rightarrow k = \log n$.

即最坏运行时间 $O(\log n)$.

c. 如b. 划分后的子问题会被区分为左右子树，而合并过程相当于从叶子结点开始回溯至根结点，遍历所有结点间的路径，不会产生重复.

对于叶结点，可以使用 Alg(a) => return a;

对于合并过程，可以使用 Alg(b) => 将所有叶结点值相乘，然后更新自身value.

d. 幂运算可以对指数作拆分，即 $a^n = a^{n/2+n/2} = a^{n/2} \cdot a^{n/2} = \cdots = \underbrace{a \cdots\cdots a}_{n}$

分治的本质即将 a^n 分为 n 个 a 相乘，故算法正确性得证.

e. $T(n) = 2T(\lfloor n/2 \rfloor) + O(1)$ (奇数所产生的一个额外a仅带来常数代价)

由主方法有 $T(n) = O(\lg n)$

2. 算法描述：每次对两个序列的中位数进行比较，若相等，则直接返回；若不等，则删除较大数对应数据库中比中位数更大的部分以及较小数对应数据库中比中位数更小的部分，然后递归进行操作直至找到中位数。

伪代码：

```
FIND-MID (A, B)
1:  a = FIND-K(A, A.length/2)
```

```
2:  b = FIND-K(B, B.length/2)
3:  if a=b then
4:      return a;
5:  else
6:      if a > b then
7:          DELETE-MORE(A,a)
8:          DELETE-LESS(B,b)
9:      else
10:         DELETE-MORE(B,b)
11:         DELETE-MORE(A,a)
12: return FIND-MID(A,B)
```

复杂度分析：

考虑最坏情况，每两次查询都将A-B数据库规模减半，直至只剩下A-B中各1个数，查询次数 $N=2k$，$2^k=n$ = $N=O(\log n)$ 即最多 $O(\log n)$ 次查询.

3、从Path1开始走1后返回选择反方向走1，然后返回，然后选择下一条路重复该过程，全部完成后若未找到宝藏则将探索距离改为2重复以下过程直至找到宝藏，并且每一次重复所走距离以2倍增长，即 $1, 2, 4, \cdots, 2^k$ $(2^{k-1}<x<2^k)$

$$ratio = \max_S \frac{Alg(S)}{\text{offline } OPT(S)}$$

$$= \frac{4m(1+2+\cdots+2^k)-2\cdot 2^k+x}{x} = \frac{4m(x-1)-2^{k+1}+x}{x}$$

$$= O(m).$$

即竞争比 $O(m)$