assignment8

Xiaoma

2022 年 12 月 5 日

题目 1.

解答.

1. 若要计算 a^n，已知当 $n = 0$ 时，$a^n = 1$，当 $n = 1$ 时，$a^n = a$，当 n 为奇数时 $a^n = (a^{\frac{n}{2}})^2 \times a$，当 n 为偶数时，$a^n = (a^{\frac{n}{2}})^2$，故可将计算 a^n 拆分为以上子问题。子问题的最坏运行时间为 $O(\log n)$。

2. 合并过程仅需将子问题得到的数字相乘，最坏运行时间为 $O(1)$。

3. 由上述可知，原问题划分出的子问题是相同的，故只需要计算一个子问题然后进行合并，故不用重复其划分子问题和合并步骤。

4. $a^n = (a^{\frac{n}{2}})(a^{\frac{n}{2}})a^{n\%2} = (a^{\frac{n}{4}}a^{\frac{n}{4}}a^{\frac{n}{2}\%2})(a^{\frac{n}{4}}a^{\frac{n}{4}}a^{\frac{n}{2}\%2})a^{n\%2} = \cdots = (\Pi_{i=1}^{\frac{n}{2}}a)(\Pi_{i=1}^{\frac{n}{2}}a)a^{n\%2}$，故该算法一定能得到正确的计算解。

5. 算法共需要递归 $\log n$ 次，故时间复杂度为 $O(\log n)$。

Algorithm 1: myPow

Input: The base : a; The exponent : n;

Output: a^n;

```
if n == 0 then
  return 1;
if n == 1 then
  return a;
temp = myPow(a, n/2);
if n%2 == 0 then
  return temp * temp;
return temp * temp * a;
```

题目 2.

解答. 设两个数据库中数字的中位数分别为 $A[a]$，$B[b]$，数据库 A 的首尾分别为 s_1, d_1，数据库 B 的首尾分别为 s_2, d_2，所以 $a = (s_1 + d_1)/2, b = (s_2 + d_2)/2$。

1. 若 $A[a] = B[b]$，则 $A[a]$ 为两个数据库的中位数。
2. 若 $a < b$
   - 若数据库 A 中元素个数为奇数，数据库 A 将 s_1 移动到 a，数据库 B 将 d_2 移动到 b
   - 若数据库 A 中元素个数为偶数，数据库 A 将 s_1 移动到 $a+1$，数据库 B 将 d_2 移动到 b
3. 若 $a > b$
   - 若数据库 A 中元素个数为奇数，数据库 A 将 d_1 移动到 a，数据库 A 将 s_2 移动到 b

- 若数据库 A 中元素个数为偶数，数据库 A 将 d_1 移动到 a，数据库 A 将 s_2 移动到 $b+1$

4. 重复 2,3 直至 1 成立，或当两数据库的首尾范围均只包含一个数时，较小的数为所求中位数。

每次迭代都舍弃一半的数据，故该算法的时间复杂度为 $O(\log n)$。

Algorithm 2: mySearch

```
Input: Database A : A; Database B : B; The size of database : n;
Output: The median of A and B;
s1 = 0, d1 = n - 1, s2 = 0, s2 = n - 1;
while s1! = d1||s2! = d2 do
    a = (s1 + d1) / 2;
    b = (s2 + d2) / 2;
    if A[a] == B[b] then
        return A[a];
    if A[a] < B[b] then
        if (s1 + d1)%2 == 0 then
            s1 = a;
            d2 = b;
        else
            s1 = a + 1;
            d1 = b;
    else
        if (s1 + d1)%2 == 0 then
            d1 = a;
            s2 = b;
        else
            d1 = a;
            s2 = b + 1;
if A[a] > B[b] then
    return B[b];
return A[a];
```

题目 3.

解答. 假设每次随机选择一个方向，若前进 i（i 递增）米后，未找到宝藏，则回到出发点选择剩余未选择的方向，将每个方向都选择一次后，重新开始循环，直到找到宝藏。

$$cost(OPT) = 2^j + \varepsilon > 2^j$$

$$cost(ON) = m(2+4+...+2^j)+2^j+\varepsilon = (m+1)2^j+\varepsilon < (m+1)cost(OPT)$$

则算法竞争比为 $O(m)$。