算法设计与分析

总复习

内容框架

- 算法基础
- 分治与递归
- 动态规划
- 贪心
- 回溯与分支限界
- 随机与近似算法

算法 设计与分析

一. 算法基础

◆ 算法的特性

输入>0，输出>=0

有限性，可行性，确定性

◆ 算法设计要求

正确性，可读性，鲁棒性，

高效率低存储

◆ 算法设计与分析阶段的主要任务

设计阶段：设计算法解决给定问题。更一般的，研究解决某类问题的一般规律和方法。

分析阶段：分析算法时空复杂度。更一般的，研究算法质量评价标准。

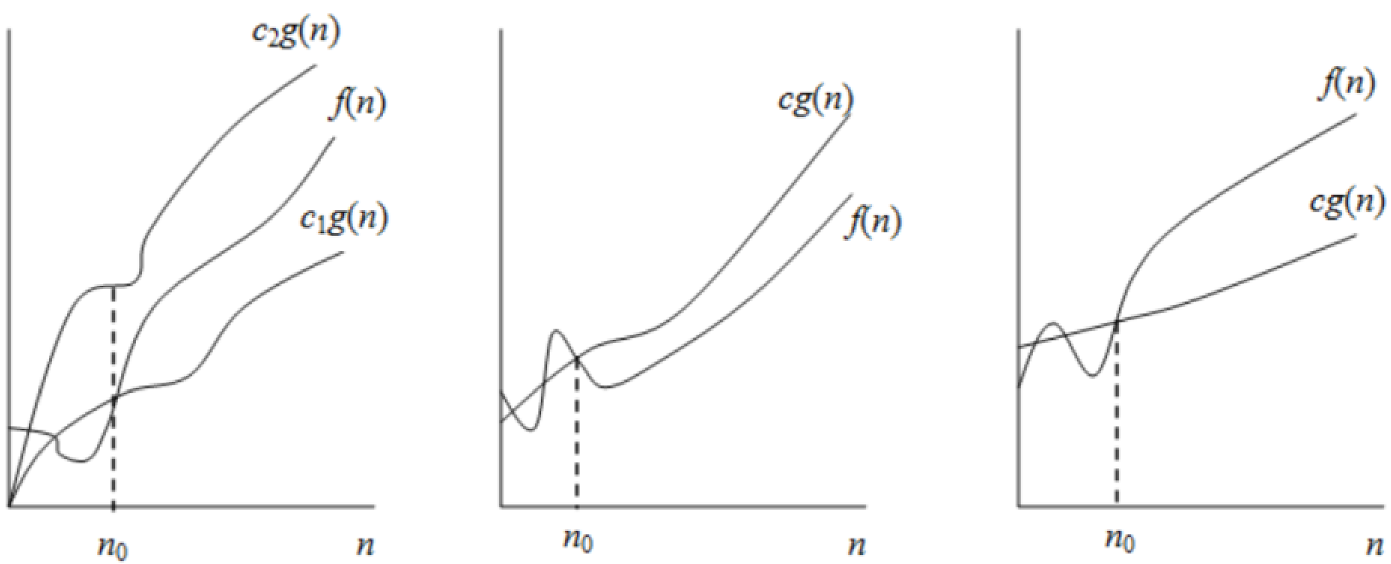
一. 算法基础

◆ 算法时间复杂度分析

大O： 描述算法时间增长率的上界

大Ω： 表述算法时间增长率的下界

大Θ： 同阶



(a) $f(n)=\Theta(g(n))$ (b) $f(n)=O(g(n))$ (c) $f(n)=\Omega(g(n))$

一. 算法基础

阶的证明——极限法

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=C\begin{cases}\text{若}c\neq 0,\text{则}f\text{和}g\text{同阶，}f=\theta(g)\\ \text{若}c=0,\text{则}f\text{和}g\text{不同阶，}f=O(g),\text{但}g\text{不是}\theta(f)\\ \text{若}c=\infty,\text{则}f=\Omega(g)\text{，}f\text{是}g\text{的高阶}\end{cases}$$

洛必达法则：

$$\text{若}\lim_{n\to\infty}f(n)=\lim_{n\to\infty}g(n)=\infty,\text{则}\lim_{n\to\infty}\frac{f(n)}{g(n)}=\lim_{n\to\infty}\frac{f^{'}(n)}{g^{'}(n)}$$

一. 算法基础

练习：证明n与$\log_2 n$的函数阶关系

证明：设f(n)=n, g(n)=$\log_2 n$

$$\lim_{n->\infty}\frac{f(n)}{g(n)}=\lim\frac{\mathrm{n}}{\log_2 n}=\lim_{n->\infty}\frac{1}{1/(\mathrm{n}\ln 2)}=\infty$$

所以，f(n)=Ω(g(n))，即n=Ω($\log_2 n$)，$\log_2 n$是n的下界

也能说，$\log_2 n$=O(n)，n是$\log_2 n$的上界。

二. 递归与分治

很多分治法算法都是采用递归实现的，但有些也可以不用递归实现。

- 递归是从编程的角度考虑的，将一个大问题转化为若干个相似的小问题求解。
- 分治法是一种算法设计策略，既可以采用递归来实现，也可以采用非递归来实现。
- 分治法的思路与递归吻合，所以很多分治法算法是采用递归来实现。

二. 递归与分治

能够用递归解决的问题应该满足的条件：

- ✓ 需要解决的问题可以转化为一个或多个子问题来求解，而这些子问题的求解方法与原问题完全相同，只是在规模上不同。
- ✓ 递归调用的次数必须是有限的，即有可以结束递归的条件来终止递归。

二. 递归与分治

递归算法分析：

✓替换法（递推）

✓递归树法

✓主方法

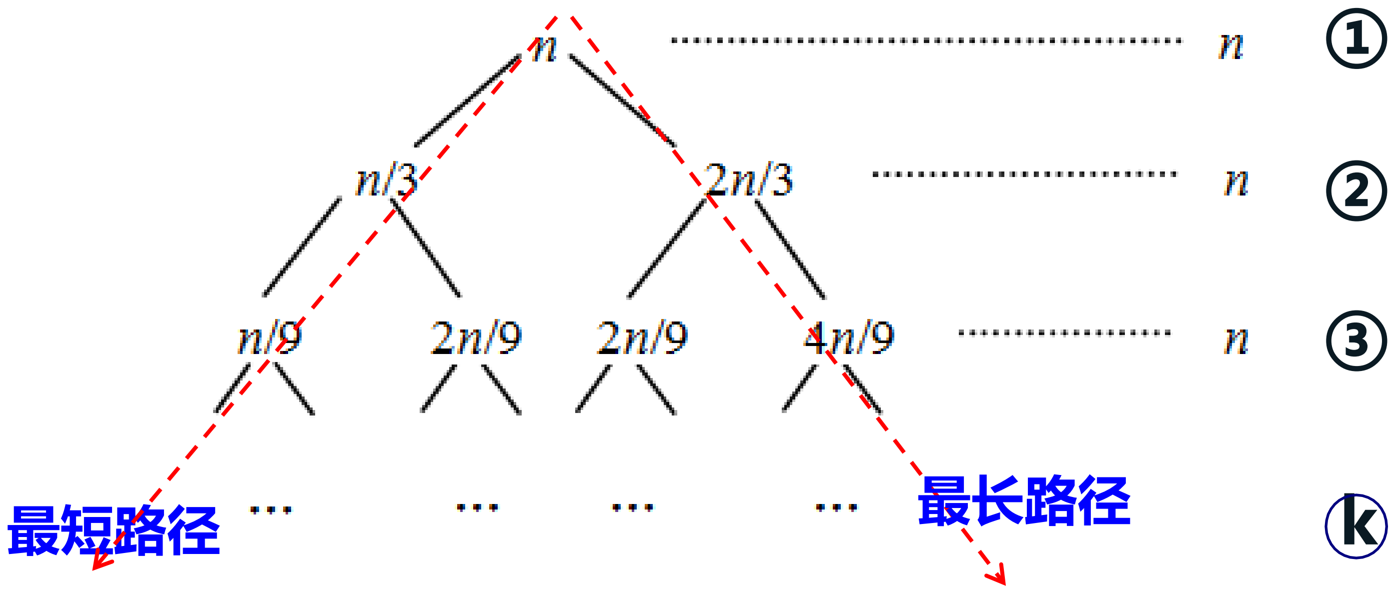
$$\begin{cases} T(n) = aT(\frac{n}{b}) + f(n) & n > 1 \\ T(1) = 1 & n = 1 \end{cases}$$

函数f(n) 与 $n^{\log_b a}$ 进行比较，

- 如果函数 $n^{\log_b a}$ 比函数f(n) 更大，则T (n) = $O(n^{\log_b a})$。
- 如果函数 $n^{\log_b a}$ 和函数f(n) 一样大，则T (n) = $O(n^{\log_b a}\log_2 n)$。
- 如果函数 $n^{\log_b a}$ 比函数f(n) 小，则T (n) =0 (f(n)) 。

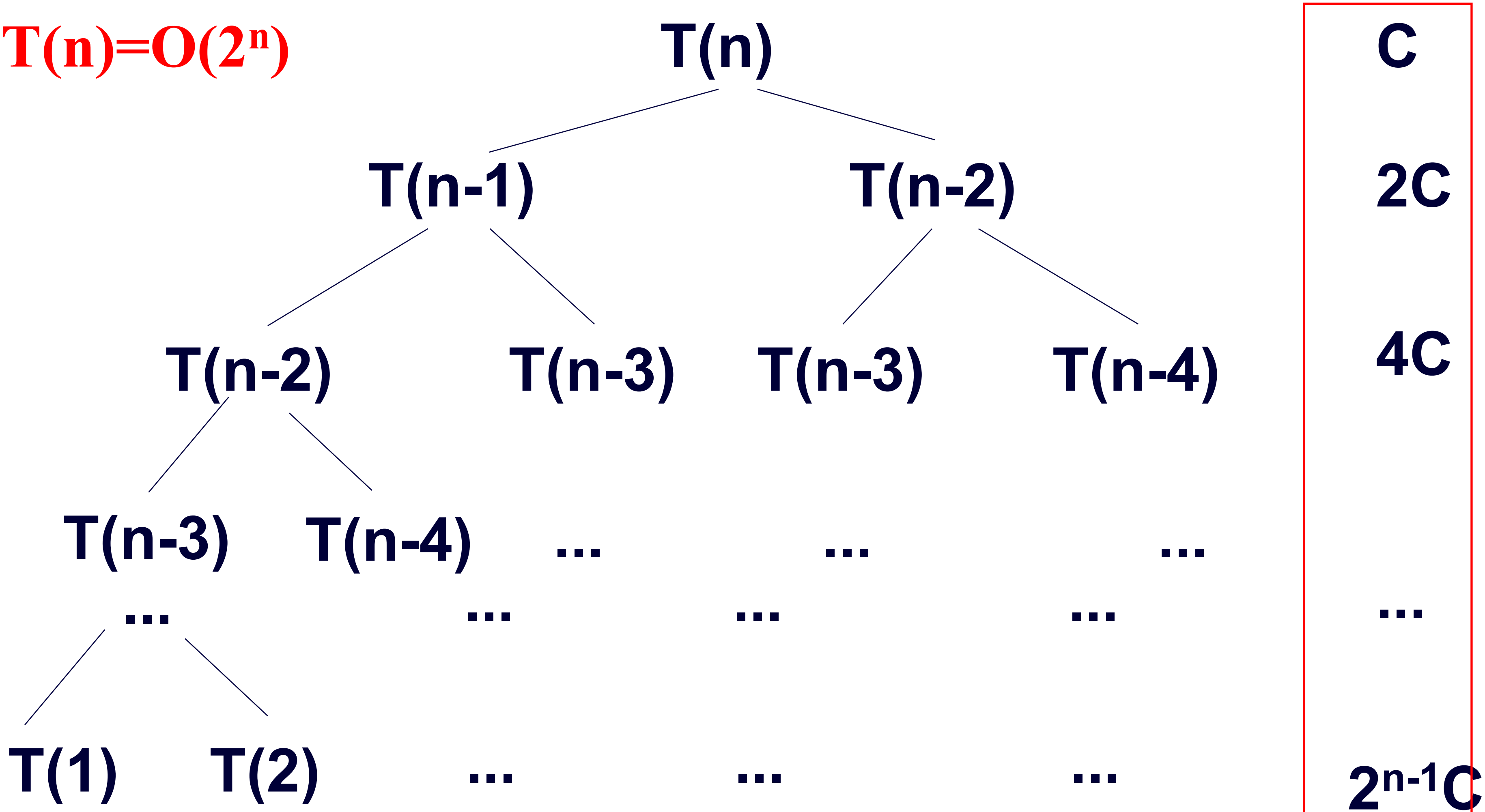
✓递归树法

$$T(n)=1 \quad 当n=1$$
$$T(n)=T(n/3)+T(2n/3)+n \quad 当n>1$$

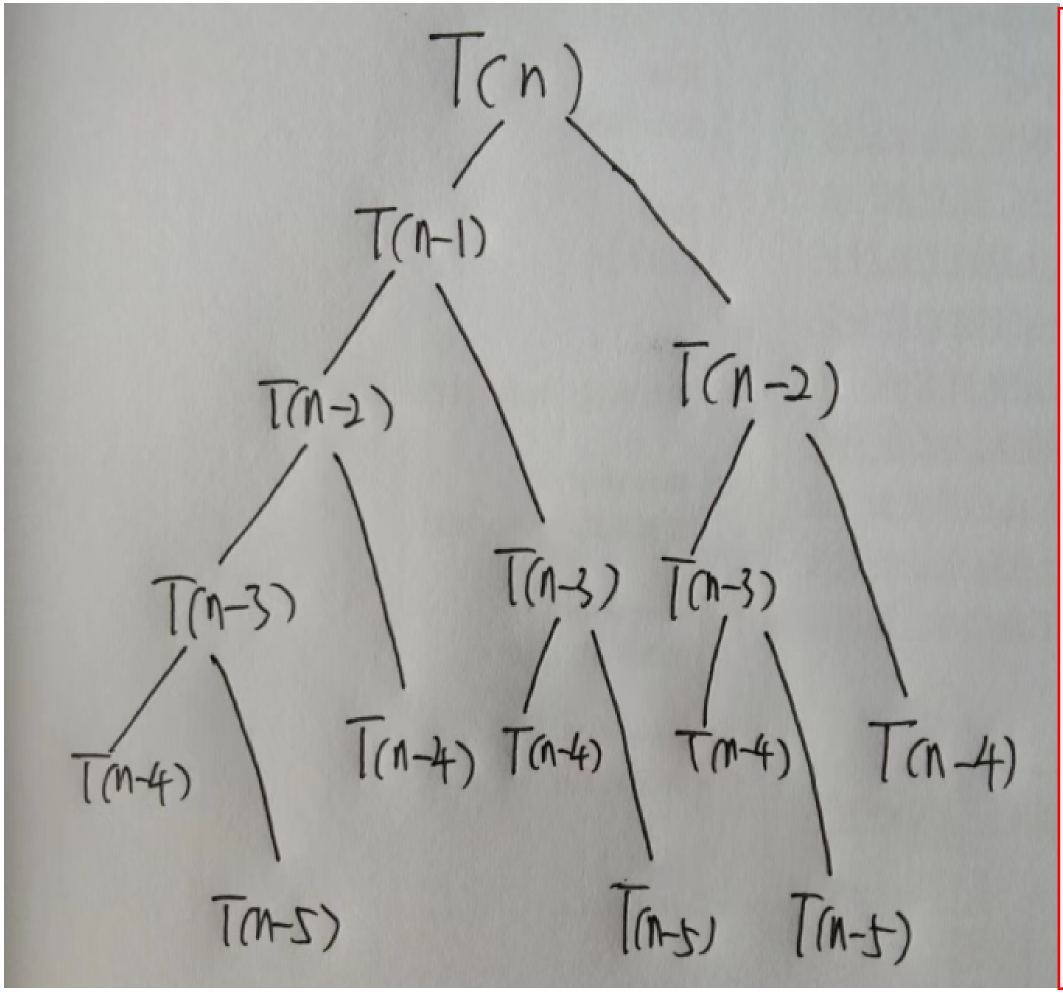


$$T(n)=1 \qquad 当n=1 \ 或 \ n=2$$
$$T(n)=T(n-1)+T(n-2)+C \qquad 当n>2$$

$T(n)=O(2^n)$



$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] \qquad S_n = F_{n+2} - 1$$



1个

1个

2个

3个

5个

...

T(n) = $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ **=O(1.618^n)**

二. 递归与分治

✓ 适用分治解决的问题特点:

（1）原问题可以分解为若干规模较小的相同问题，即该问题具有最优子结构性质。(前提)

（2）子问题缩小到一定的程度就可以容易地解决。(大部分问题都满足)（递归出口）

（3）子问题的解可以容易的合并为原问题的解。(关键)

（4）各子问题相互独立，即子问题之间不包含公共子问题。(会影响效率)

✓ 分治法步骤：分解-》求解-》合并

二. 递归与分治

讲过的例子：

- 递归：简单选择排序、全排列、二叉树构建
- 排序：快速排序、归并排序（自顶向下，自底向上）
- 查找：n个数中求最大/最小值（自顶向下，自底向上）；
  第k小元素；查找两序列中位数
- 组合：大整数乘法；最大连续子序列和；棋盘覆盖

关键：

- 如何将大问题分解为相同子问题
- 如何将子问题求解结果合并为大问题求解结果

三. 动态规划（填表法）

能采用DP求解的问题，一般具有3个性质：

（1）最优子结构：问题的最优解，包含子问题的最优解。（基本条件）。

（2）无后效性（马尔科夫性）：即某阶段状态一旦确定，就不受这个状态以后决策的影响。

（3）有重叠子问题（重叠子结构）：子问题之间不独立，一个子问题在下一阶段决策中可能被多次使用到。（不必要，但关乎效率）

三. 动态规划

动态规划方法分类：

- ◆自底向上的动态规划
- ◆自顶向下的动态规划：递归+备忘录

三. 动态规划DP

讲过的例子：

- 走台阶、最少张数钞票找零、求解组合数
- 整数分解
- 矩阵连乘问题(最佳次序)
- 0-1背包问题
- 资源分配

- 最大连续子序列和
- 最长递增子序列
- 最长公共子序列
- 编辑距离

三. 动态规划DP

可以先考虑穷举思路，发现隐藏其中的“最优子结构”和“重复子问题”，再寻找问题的递推关系。

动态规划 ≠ 穷举 ≠ 分治

步骤：

①将大问题转换为小问题（状态转移），分析问题最优子结构

②定义问题状态和状态之间的关系（状态转移方程，确定dp含义）。

③以自底向上或自顶向下（备忘录法）方式填表或求解。

④根据计算最优值时得到的信息，构造问题最优解。

四. 贪心算法

✓贪心算法通过做一系列的贪心选择（当前看起来最好的选择），给出某一问题的最优解。对算法中的每一个决策点做出当时看起来最佳的选择。

✓适合贪心求解的问题:

贪心选择性质：问题的整体最优解可以通过一系列局部最优选择，即贪心选择来达到。

最优子结构性质：一个大问题的最优解包含着它子问题的最优解

✓贪心法不能保证问题总能得到最优解。

四. 贪心算法

讲过的例子：

- 汽车加油问题
- 活动安排问题/蓄栏保留问题/区间相交问题
- 购买股票/摇摆序列
- 多机调度
- 移除k个数字
- 跳跃游戏

五-六. 回溯与分支限界

□ 解空间树结构

子集树

排列树

m叉树

扩展结点：一个正在产生儿子的结点（某时刻只有1个）

活结点：一个自身已经生成，但儿子还没有全部生成的结点

死结点：一个所有儿子已经生成的结点

算法 设计与分析

五-六.回溯与分支限界

- **回溯：深度遍历解空间**
- **分支限界：广度遍历解空间**
  - **队列式分支限界**
  - **优先队列式分支限界**
- **时间复杂度高，一般需要设计剪枝函数**

  约束函数：剪去不满足约束条件的子树

  限界函数：剪去不能得到最优解的子树

算法 设计与分析

五-六.回溯与分支限界

讲过的例子：

回溯

- 子集树：0-1背包问题、最大团
- 排列树：n后问题、哈密顿回路、TSP
- m叉树：n皇后问题、地图着色

分支限界

- 0-1背包
- 单源最短路径
- 八数码问题
- 流水作业调度问题

七、随机算法

- **数值随机算法：**常用于数值问题的求解，这类算法所得到的往往是近似解，而且近似解的精度随计算时间的增加不断提高。（求PI，求积分、求不规则图形面积等）
- **舍伍德（Sherwood）算法：**引入随机性消除算法的时间复杂性与输入实例间的这种联系，降低最坏情况发生概率。总能求得问题的一个解，且所求得的解总是正确的。（快排、有序表查找等）

七、随机算法

- 拉斯维加斯（Las Vegas）算法。随机选择一步进行。可能找不到解，也可能很快找到解。但一旦找到解，一定是正确的解。需要多次运行算法。用以改善平均情况。（求重复元素，N后问题，挑钥匙）
- 蒙特卡罗（Monte Carlo）算法。又称计模拟法、随机抽样技术。能够求得问题的一个解，但这个解未必是正确（最优）的。算法运行时间越长（抽样次数越多），得到正确解的概率越大。用以解决难题。（不规则图形面积等，大数的素数判定，主元素判定，挑苹果）

七、随机算法

- 拉斯维加斯（Las Vegas）算法。
- 蒙特卡罗（Monte Carlo）算法。

共同点：找到正确解的概率随着它所用的计算时间的增加而提高

不同点：

七、随机算法

- 给定n个元素（n为偶数）的序列A，一半为0一半为1，想要找到一个元素为1的下标。

```
repeat:
    k = RandInt(n)
    if A[k] = 1, return k
return "Failed"
```

拉斯维加斯

```
repeat 300 times:
    k = RandInt(n)
    if A[k] = 1, return k
return "Failed"
```

蒙特卡洛

七、随机算法

- 有一个1000000的整数集合，求其中位数。随机抽取m<1000000，将它们的中位数近似看做这个集合的中位数。随着m的增大，这个结果是正确解的概率也在增加。但除非m=1000000，否则无法知道近似结果知否真实。

蒙特卡洛

- 拉斯维加斯算法，也是随机选择或采样。运行（采样）次数越多，得到正确解的概率逐渐增大。但过程中如果得到了正确解，则结束。

期末考试

一、简答题（每题5分，共20分）

二、选择题（每空1分，共10分）

三、填空题（每空2分，共20分）

四、算法构造题（每小题10分，共30分）

五、算法设计(每小题10分，共20分)