任选5道题，必须包含必选题

1.1　计算模型

Problem 1.1.1 (三个数排序)
输入三个各不相同的整数：

1. 请设计一个算法将输入的三个整数排序；
2. 在最坏情况下、平均情况下你的算法分别需要进行多少次比较？(假设所有可能的输入等概率出现。)
3. 在最坏情况下将三个不同整数排序至少需要多少次比较？证明你的结论。

Problem 1.1.2 (三个数的中位数)
输入三个各不相同的整数：

1. 请设计一个寻找三个数的中位数的算法；
2. 在最坏情况下、平均情况下你的算法分别需要进行多少次比较？(假设所有可能的输入等概率出现。)
3. 在最坏情况下找出三个不同整数的中位数至少需要多少次比较？证明你的结论。

Problem 1.1.3 (集合覆盖问题)
我们定义集合的最小覆盖如下：已知全集$U = \{1, ..., n\}$。给定U的子集组成的集合族$S = \{S_1, ..., S_m\}$，找出S的最小子集T，满足$\bigcup\limits_{S_i \in T} S_i = U$。例如，全集$U = \{1, 2, 3, 4, 5\}$有下面几个子集$S_1 = \{1, 3, 5\}, S_2 = \{2, 4\}, S_3 = \{1, 4\}$和$S_4 = \{2, 5\}$，则最小覆盖为$\{\, S_1, S_2 \,\}$。

1. 找出下面算法失败的例子：首先选择S中最大的集合S_i，并从全集中将S_i中的所有元素删除；然后从S中剩余的集合中挑选最大的并从全集中删除对应元素；重复上述过程直到全集中的所有元素都被覆盖；
2. 请设计一个算法计算输入全集的一个集合覆盖，并证明你所设计算法的正确性；
3. 你所设计的算法能否保证总是得出最小覆盖？如果不能，请针对你的算法设计一个反例。

Problem 1.1.4 (换硬币问题)
我们定义换硬币问题如下：给定n个硬币，它们的面值是正整数$S = \{\, s_1, s_2, ..., s_n \,\}$；另外给定一个正整数金额值$T$。我们需要从$S$中找出若干个硬币，使得它们的面值和为$T$，或者返回不存在这样的硬币集合。我们给出两种不同的算法设计方案：

1. 依次扫描硬币$s_1, s_2, ..., s_n$，并累加金额；
2. 按面值从小到大的顺序，依次扫描硬币，并累加金额；
3. 按面值从大到小的顺序，依次扫描硬币，并累加金额；

在上述扫描过程中，如果如果金额值累积到正好为T，则返回已经扫描到的硬币；否则返回不存在。请将上述三种方案分别写成算法，并通过举反例的方式证明这三个算法的“不正确性”。

Problem 1.1.5 (子集和问题)

子集和(subset sum)问题定义如下：给定一个整数集合$S = \{s_1, s_2, ..., s_n\}$和一个目标数字$T$，要找$S$的一个子集使得子集中的整数的和为$T$。例如，$S = \{1, 2, 5, 9, 10\}$，存在子集其和为$T = 22$，但是不存子集其和为$T = 23$。请找出下列解决子集和问题的算法失败的例子：

1. S中的元素保持原本顺序，按照从左到右的顺序依次选入子集。如果当前子集的和小于T则继续挑选；否则算法结束。我们称这个算法为“first-fit”；
2. 将S中的元素按照从小到大排列，然后按照从小到大的顺序依次选入子集。如果当前子集的和小于T 则继续挑选；否则算法结束。我们称这个算法为“smallest-fit”；
3. S中的元素按照从大到小排列，然后按照从大到小的顺序依次选入子集。如果当前子集的和小于T，则继续挑选，否则算法结束。我们称这个算法为我们称这个算法为“largest-fit”。

Problem 1.1.6 (多项式计算)

HORNER算法是用来求解多项式$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ 的，请证明其正确性。

Algorithm: HORNER($A[0..n], x$)
$p := A[n]$; /* 数组$A[0..n]$存放系数$a_0, a_1, \cdots, a_n$ */
for $i := n - 1$ **downto** 0 **do**
  $p := p \cdot x + A[i]$;
return p ;

Problem 1.1.7 (整数相乘(必选))

INT-MULT算法用来计算两个非负整数y, z的乘积。

Algorithm: **int** INT-MULT(**int** y, **int** z)
if $z = 0$ **then**
  return 0;
else
  return multiply(cy,$\lfloor \frac{z}{c} \rfloor$) + y $\cdot$ (z mod c);

---

1. 令$c = 2$，请证明算法的正确性；
2. 令c为任意的一个不小于2的常数，请证明算法的正确性。

Problem 1.1.8 (平均情况时间复杂度的定义(必选))

算法的输入r为1到n之间的正整数，r取不同值的概率如下：

$$Pr\{r = i\} = \begin{cases} \frac{1}{n} & (1 \le i \le \frac{n}{4}) \\ \frac{2}{n} & (\frac{n}{4} < i \le \frac{n}{2}) \\ \frac{1}{2n} & (\frac{n}{2} < i \le n) \end{cases}$$

假设n是4的倍数。请针对输入的取值情况，分析下面算法的平均情况时间复杂度(一个operation的代价记为1)。

if $r \le \frac{n}{4}$ **then**
    perform 10 operations ;
else if $\frac{n}{4} < r \le \frac{n}{2}$ **then**
    perform 20 operations ;
else if $\frac{n}{2} < r \le \frac{3n}{4}$ **then**
    perform 30 operations ;
else
    perform n operations ;