HW1

Page12：2.1-3，2.1-4
Page16：2.2-3

2.1

2.1-3 考虑以下查找问题：
输入：n 个数的一个序列 $A=\langle a_1, a_2, \cdots, a_n\rangle$ 和一个值 v。
输出：下标 i 使得 $v=A[i]$ 或者当 v 不在 A 中出现时，v 为特殊值 NIL。
写出线性查找的伪代码，它扫描整个序列来查找 v。使用一个循环不变式来证明你的算法是正确的。确保你的循环不变式满足三条必要的性质。

```
Search(A,v):
#input: a sequence of n numbers A=<a1,a2,...,an> and a value v
#output:index i such that v=A[i] or the special value NIL if v does not appear in
A
    for(i=1 to A.length)
        if A[i]==v
            return i
    return NIL
```

循环不变式

在for循环的每次迭代开始时，数组A[1...i-1]由不同于v的元素组成。

初始化

最初子数组为空数组，循环不变式成立。

保持

在每一步，我们知道A[I..i-1]不包含v。我们将其与A[i]进行比较。如果相同，返回i，否则继续下一步。我们已经确保A[1..i-1]不包含v并且A[i]与v不同，因此这一步保留了不变量。

终止

当i>A.length时循环终止。由于i增加1并且i>A.length,我们知道A中的所有元素都已经检查完毕，发现v不在其中，因此，返回NIL。

2.1-4 考虑把两个 n 位二进制整数加起来的问题，这两个整数分别存储在两个 n 元数组 A 和 B 中。这两个整数的和应按二进制形式存储在一个 $(n+1)$ 元数组 C 中。请给出该问题的形式化描述，并写出伪代码。

输入：存储两个n位二进制数的n元数组A[1..n], B[1..n]

输出：存储A, B对应二进制数的和的n+1元数组C[1..n+1]

伪码：

```
BinaryAdd(A, B):
# 输入：A[1..n]，B[1..n]。A[1]存二进制整数最低位，A[n]存二进制整数最高位;数组B同理。
# 输出：C[1..n+1]。C[1]存整数和最低位，C[n+1]存整数和最高位
    #初始化
    for(i=1 to n+1):
        C[i]=0
    # carry表进位
    carry = 0
    for(i=1 to n):
        C[i] = (A[i]+B[i]+carry) % 2
        carry = (A[i]+B[i]+carry) / 2
    # 最后要将C[n+1]置为carry
    C[n+1] = carry
```

- 如果用carry变量，则最后要注意加上C[n+1]=carry。
- 不用carry变量，用C[i+1]存进位值也可以。

法2：如果if-else分类讨论，则要考虑sum=0, 1, 2, 3的情况。

2.2

2.2-3 再次考虑线性查找问题(参见练习 2.1-3)。假定要查找的元素等可能地为数组中的任意元素，平均需要检查输入序列的多少元素？最坏情况又如何呢？用 Θ 记号给出线性查找的平均情况和最坏情况运行时间。证明你的答案。

2.1-3 考虑以下**查找问题**：
输入：n 个数的一个序列 $A=\langle a_1, a_2, \cdots, a_n\rangle$ 和一个值 v。
输出：下标 i 使得 $v=A[i]$ 或者当 v 不在 A 中出现时，v 为特殊值 NIL。
写出**线性查找**的伪代码，它扫描整个序列来查找 v。使用一个循环不变式来证明你的算法是正确的。确保你的循环不变式满足三条必要的性质。

1. 平均需要检查：

   v出现的位置求个期望：n个元素每个出现的概率都是$1/n$，故期望位置$\Sigma_{i=1}^{n} i/n = (n+1)/2$

   最坏需要检查n个元素

2. 平均和最坏都是 $\theta(n)$。证明：由 *1* ，平均和最坏都要检查$\theta(n)$量级的元素，一次检查耗时$\theta(1)$.