题目一

1. 算法设计

Algorithm 1 AntiDiagonalMatrixVectorMultiply

```
function AntiDiagMatVec(a, v, n)
    res ← new array of length n
    for i = 0 to n − 1 do
        sum ← 0
        for j = 0 to n − 1 do
            k ← n − 1 − i + j
            sum ← sum + a[k] · v[j]
        end for
        res[i] ← sum
    end for
    return res
end function
```

2. 复杂度分析

对于每个 i（共 n 个），内层循环需要遍历所有 j（共 n 个），执行一次乘法和加法。因此总运算量为：

$$T(n) = n \times n = \boxed{O(n^2)}$$

该算法在时间复杂度上是最优的，因为乘积中每一项都依赖于压缩向量 a 中的一个元素，必须访问 A 的所有相关元素。

题目二

1. 算法步骤

1: **输入**: 主串 S（长度 n），模式串 P（长度 m）
2: **输出**: 所有匹配位置 L
3: 初始化匹配结果数组 $matches[0..n-m] \leftarrow 0$
4: **for** 每个字符 $c \in \Sigma$ **do**
5: 构造 $P_c[i] = \mathbb{I}(P[i] = c)$，$S_c[j] = \mathbb{I}(S[j] = c)$
6: 计算 $\hat{P}_c = \mathrm{FFT}(\mathrm{reverse}(P_c))$
7: 计算 $\hat{S}_c = \mathrm{FFT}(S_c)$
8: 计算 $cross_c = \mathrm{IFFT}(\hat{P}_c \odot \hat{S}_c)$

9: $matches \leftarrow matches + \mathrm{Re}(cross_c[m-1..n-1])$

10: **end for**

11: 计算通配符贡献 $wild = \mathrm{conv}(P_{wild}, \mathbf{1}_m)$

12: $L \leftarrow \{k \mid matches[k] + wild[k] = m\}$

13: **return** L

2. 时间复杂度分析

$$T(n) = |\Sigma| \cdot O(n \log n) + O(n) = O(n \log n)$$

3. 优化思路

- **字母表缩减**:

$$T_{opt}(n) = |unique(P)| \cdot O(n \log n)$$

- **批处理编码**:

将Σ分为$\lceil \log_2 |\Sigma| \rceil$组，复杂度降为$O(\log |\Sigma| \cdot n \log n)$

- **并行计算**:

使用p个处理器，时间降为$O(\frac{|\Sigma|}{p} n \log n)$