

Homework 6

1. 假定我们对一个数据结构执行一个由 n 个操作组成的操作序列，当 i 严格为 2 的幂时，第 i 个操作的代价为 i ，否则代价为 1。（1）使用聚合分析确定每个操作的摊还代价。（2）使用核算法确定每个操作的摊还代价。（3）使用势能法确定每个操作的摊还代价。

2. V.Pan 发现一种方法，可以用 132464 次乘法操作完成 68×68 的矩阵相乘，发现另一种方法，可以用 143664 次乘法操作完成 70×70 的矩阵相乘，还发现一种方法，可以用 155424 次乘法操作完成 72×72 的矩阵乘法。当用于矩阵乘法的分治算法时，上述哪种方法会得到最佳的渐近运行时间？与 Strassen 算法相比，性能如何？

3. 我们可以将一维离散傅里叶变换 (DFT) 推广到 d 维上。这时输入是一个 d 维的数组 $A = (a_{j_1, j_2, \dots, j_d})$ ，维数分别为 n_1, n_2, \dots, n_d ，其中 $n_1 n_2 \dots n_d = n$ 。定义 d 维离散傅里叶变换如下：

$$y_{k_1, k_2, \dots, k_d} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \dots \sum_{j_d=0}^{n_d-1} a_{j_1, j_2, \dots, j_d} \omega_{n_1}^{j_1 k_1} \omega_{n_2}^{j_2 k_2} \dots \omega_{n_d}^{j_d k_d}$$

其中 $0 \leq k_1 < n_1, 0 \leq k_2 < n_2, \dots, 0 \leq k_d < n_d$ 。

a. 证明：我们可以依次在每个维度上计算一维的 DFT 来计算一个 d 维的 DFT。也就是说，首先沿着第 1 维计算 n/n_1 个独立的一维 DFT。然后，把沿着第 1 维的 DFT 结果作为输入，我们计算沿着第 2 维的 n/n_2 个独立的一维 DFT。利用这个结果作为输入，我们计算沿着第三维的 n/n_3 个独立的一维 DFT，如此下去，直到第 d 维。

b. 证明：维度的次序并无影响，于是可以通过在 d 个维度的任意顺序中计算一维 DFT 来计算一个 d 维的 DFT。

c. 证明：如果采用计算快速傅里叶变换计算每个一维的 DFT，那么计算一个 d 维的 DFT 的总时间是 $O(n \lg n)$ ，与 d 无关。