Homework 6

和泳毅 PB19010450

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。

答:

（1）令$I=\{i|i\text{不是2的幂}, i\leq n, i\in N\}$。

$$\begin{aligned}\sum_{i=1}^{n}c(i) &= \sum_{i=1}^{\lfloor\lg n\rfloor}2^i+\sum_{i\in I}1\\ &\leq \sum_{i=1}^{\lfloor\lg n\rfloor}2^i+n\\ &= 2^{1+\lfloor\lg n\rfloor}-1+n\\ &\leq 2n-1+n\\ &\leq 3n\end{aligned}$$

所以这n个操作序列的最坏情况时间为$O(n)$，摊还代价为$O(n)/n=O(1)$。

（2）

	i严格为2的幂	其他
实际代价c_i	i	1
摊还代价$\hat{c}_i$	3	3

当i不是2的幂时，花费1个代价，存入2个信用。当i是2的幂时，花费3个代价与$i-3$个信用。对第一个2的幂：2，已有3个信用，花费实际代价后保持信用非负。此后任意两个相邻的2的幂：2^k与2^{k+1}，相差$2^{k+1}-2^k-1=2^k-1$个数，期间存入信用$2(2^k-1)=2^{k+1}-2$，大于在2^{k+1}上的花费$2^{k+1}-3$。所以保持信用非负。

所以总摊还代价为$O(n)$，总实际代价也是$O(n)$，摊还代价为$O(n)/n=O(1)$。

(3)

定义势函数$\Phi(D_0)=0,\Phi(D_i)=2i-2^{1+\lfloor\lg i\rfloor}$。

则有

$$\Phi(D_n)=2n-2^{1+\lfloor\lg n\rfloor}\geq 2n-2n=0=\Phi(D_0)$$

对操作1：

$$\hat{c}_1=c_1+\Phi\left(D_1\right)-\Phi\left(D_0\right)=1+2-2-0=1$$

对其他不是2的幂的操作：

$$\hat{c}_i=c_i+\Phi\left(D_i\right)-\Phi\left(D_{i-1}\right)=1+2i-2^{1+\lfloor\lg i\rfloor}-\left(2(i-1)-2^{1+\lfloor\lg(i-1)\rfloor}\right)=3$$

对是2的幂的操作：

$$\begin{aligned}\hat{c}_i&=c_i+\Phi\left(D_i\right)-\Phi\left(D_{i-1}\right)\\&=i+2i-2^{1+j}-\left(2(i-1)-2^{1+j-1}\right)\\&=i+2i-2i-2i+2+i\\&=2\end{aligned}$$

每个操作的摊还代价都为$O(1)$，n个操作的总摊还代价是$O(n)$。

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？

答：

（1）$T(n)=132464T(n/68)+\Theta(n)$，

由主定理：$T(n)=\Theta(n^{\log_{68}132464})\approx\Theta(n^{2.795128})$。

（2）$T(n)=143664T(n/70)+\Theta(n)$，

由主定理，$T(n)=\Theta(n^{\log_{70}143664})\approx\Theta(n^{2.795162})$。

（3）$T(n)=155424T(n/72)+\Theta(n)$，

由主定理，$T(n)=\Theta(n^{\log_{72}155424})\approx\Theta(n^{2.795147})$。

$Strassen$算法渐近运行时间为$\Theta(n^{\lg 7})\approx\Theta(n^{2.807355})$，最佳的是用132464次乘法操作完成$68\times 68$的矩阵相乘。

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$n_1n_2\dots n_d=n$。定义$d$维离散傅里叶变换如下：

$$y_{k_1,k_2,\ldots,k_d}=\sum_{j_1=0}^{n_1-1}\sum_{j_2=0}^{n_2-1}\cdots\sum_{j_d=0}^{n_d-1}a_{j_1,j_2,\ldots,j_d}\,\omega_{n_1}^{j_1k_1}\omega_{n_2}^{j_2k_2}\ldots\omega_{n_d}^{j_dk_d}$$

其中$0\le k_1\le n_1, 0\le k_2\le n_2,\ldots,0\le k_d\le n_d$。

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$n/n1$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$n/n2$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 $n/n3$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$O(nlgn)$，与d无关。

证明：

a.

$$\begin{aligned}y_{k_1,\ldots,k_d}&=\sum_{j_1=0}^{n_1-1}\cdots\sum_{j_d=0}^{n_d-1}a_{j_1,\ldots j_d}\,\omega_{n_1}^{j_1k_1}\ldots\omega_{n_d}^{j_dk_d}\\&=\sum_{j_d=0}^{n_d-1}\cdots\sum_{j_1=0}^{n_1-1}a_{j_1,\ldots j_d}\,\omega_{n_1}^{j_1k_1}\ldots\omega_{n_d}^{j_dk_d}\\&=\sum_{j_d=0}^{n_d-1}\cdots\sum_{j_2=0}^{n_2-1}\left(\sum_{j_1=0}^{n_1-1}a_{j_1,\ldots j_d}\,\omega_{n_1}^{j_1k_1}\right)\omega_{n_2}^{j_2k_2}\ldots\omega_{n_d}^{j_dk_d}\end{aligned}$$

括号内的内容是一维的DFT，需要对外部的每个索引计算，即计算$n_2\ldots n_d=n/n_1$次。计算完后求和符号就消去一个，如此往复可以减少维度直到第d维度。

b.

$$\begin{aligned}y_{k_1,\ldots,k_d}&=\sum_{j_d=0}^{n_d-1}\cdots\sum_{j_2=0}^{n_2-1}\left(\sum_{j_1=0}^{n_1-1}a_{j_1,\ldots j_d}\,\omega_{n_1}^{j_1k_1}\right)\omega_{n_2}^{j_2k_2}\ldots\omega_{n_d}^{j_dk_d}\\&=\sum_{j_d=0}^{n_d-1}\cdots\sum_{j_1=0}^{n_1-1}\left(\sum_{j_2=0}^{n_2-1}a_{j_1,\ldots j_d}\,\omega_{n_2}^{j_2k_2}\right)\omega_{n_1}^{j_1k_1}\ldots\omega_{n_d}^{j_dk_d}\\&=\ldots\end{aligned}$$

可以任意交换求和的顺序，因为求和的指标没有出现在不同求和符号的边界中。

c.

沿着第k维计算每个DFT的时间是$O(n_k\lg(n_k))$，并且因为我们只需要执行最多$n/(\Pi_{j\le k}n_j)$次，所以在k维上的运行时间最多为$O(n/(\Pi_{j<k}n_j)\lg(n_k))$。不妨设$n_k\ge 2$：

$$
\begin{aligned}
\sum_{k=1}^{d} n/\left(\prod_{j<k} n_j\right)\lg(n_k) &\le \lg(n)\sum_{k=1}^{d} n/\left(\prod_{j<k} n_j\right) \\
&\le \lg(n)\sum_{k=1}^{d} n/2^{k-1} \\
&< n\lg(n)
\end{aligned}
$$

所以计算一个d维的DFT的总时间是$O(nlgn)$，与d无关。