Algorithm Assignment 7

Author Haihan Gao

Question 1

When $i == 2^k$ $cost(i) = i$ else $cost(i) = 1$

Assume k is a positive Integer and $2^{k+1} \geq n$ and $2^k \leq n$

Then $k = log_2 n$

$Totalcost = \sum_{k=1}^{k=log_2 n} k + n - log_2 n = 2^{log_2 n+1} - 1 + n - log_2 n = O(n)$

$costforeachoperation = O(1)$

Question 2

Given the virtual_cost

case	virtual_cost
i==2^k	0
else	0

for operation $i == 2^k$ We have profit 2^k

$for(i = 2^k; i \leq 2^{k+1}; i++)cost(i) = 0$

so we have to pay for the real cost for the case $else$ from profit We earn

Fortunately, when we do Operation i ,we have a profit for 2^k and this is enough to pay for the Operation between 2^k and 2^{k+1}

So the average cost is O(1)

Question 3

Assume $i = 2^j + k$ Ω(D_i)=2×k

k==0 Then $c^{`} = c_i + \Omega(D_i) - \Omega(D_{i-1}) = 2$

k!=0 Then $c^{`} = 3$

So Average cost is O(1)

Question 4

（1）将高维向量中的每一个较低的向量视为它的一个元素，向量x=$(a_1, a_2, ..., a_{d_1})$

$a_1 = b_1, b_2, ..., b_{d_2}$ 依次类推

这样只需要对低维向量进行FFT得到高维向量，再对高维向量进行FFT得到更高维向量

（2）这个求和展开之后是通项元素与单位根乘积的全排列，所以求和顺序不影响最终结果

(3) $Time = \sum_{k=1}^{k=d} n \div n_k \times O(n_1 k \times lg_2 n_k) = O(n \times lg_2 n)$