Homework #5

Student name: 常芸凡 *(2020311273)*

Course: 算法分析与设计 – Professor: 王振波

Due date: *2020* 年 *10* 月 *13* 日

Master Theorem: Suppose that $T(n)$ is a function on the nonnegative integers that satisfies the recurrence

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $\frac{n}{b}$ means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Let $k = \log_b a$,

1. If $f(n) = O\left(n^{k-\epsilon}\right)$ for some constant $\epsilon > 0$, then
$$T(n) = \Theta\left(n^k\right)$$
2. If $f(n) = \Theta\left(n^k \log^p n\right)$, then $T(n) = \Theta\left(n^k \log^{p+1} n\right)$
3. If $f(n) = \Omega\left(n^{k+\epsilon}\right)$ for some constant $\epsilon > 0$ and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

1. 不妨设 $f(n) \le C_1 n^{k-\epsilon}, T(1) = C_2$，我们首先归纳证明其下界:

   显然 $T(1) \ge C_2 1^k$，设对于 $m < n$，我们都有 $T(m) \ge C_2 m^k$，则

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \ge aC_2(\frac{n}{b})^k = C_2 n^k \tag{1}$$

   从而 $T(n) \ge C_2 n^k$

   对于上界，我们先证明 $n = b^m, m \in \mathbb{Z}^+$ 的情形：

$$\begin{aligned} T(n) &= aT\left(\frac{n}{b}\right) + f(n) \\ &= a^2 T(n/b^2) + af(n/b) + f(n) \\ &= \cdots \\ &= a^m T(1) + \sum_{i=0}^{m-1} a^i f(n/b^i) \end{aligned} \tag{2}$$

而 $a^m T(1) = n^k T(1) = C_2 n^k$，

$$\begin{aligned}
\sum_{i=0}^{m-1} a^i f(n/b^i) &\le \sum_{i=0}^{m-1} a^i C_1 (\frac{n}{b^i})^{k-\epsilon} \\
&= C_1 n^{k-\epsilon} \sum_{i=0}^{m-1} b^{i\epsilon} = C_1 n^{k-\epsilon} \frac{n^\epsilon - 1}{b^\epsilon - 1} \\
&\le \frac{C_1}{b^\epsilon - 1} n^k = C_3 n^k
\end{aligned} \tag{3}$$

其中 $C_3 = \dfrac{C_1}{b^\epsilon - 1}$，于是结合式 2我们可知 $T(n) \le C_4 n^k$，其中 $C_4 = C_2 + C_3, n = b^m$

对于 $b^{m-1} < n < b^m$ 时，我们可知

$$T(n) \le T(b^m) \le C_4 b^{mk} \le C_4 b^{(\log_b n + 1)k} = C_4 a n^k$$

从而我们有 $T(n) = \Theta\left(n^k\right)$ ∎

2. 不妨设 $A_1 n^k \log^p n \le f(n) \le A_2 n^k \log^p n$。则由式 2可知,$T(n) = a^m T(1) + \sum_{i=0}^{m-1} a^i f(n/b^i)$，其中 $n = b^m$，而 $a^m T(1) = \Theta(n^k)$，

$$\begin{aligned}
\sum_{i=0}^{m-1} a^i f(n/b^i) &\le \sum_{i=0}^{m-1} a^i A_2 (\frac{n}{b^i})^k \log^p \frac{n}{b^i} \\
&= A_2 n^k \sum_{i=0}^{m-1} (\log n - i \log b)^p \\
&\le A_2 n^k \frac{\int_0^{(m+1)\log b} t^p \mathrm{d}t}{\log b} \\
&= \frac{A_2}{(p+1)\log b} (1 + \frac{1}{m})^{p+1} n^k \log^{p+1} n \\
&\le \frac{2^{p+1} A_2}{(p+1)\log b} n^k \log^{p+1} n
\end{aligned} \tag{4}$$

类似的，我们有

$$\begin{aligned}
\sum_{i=0}^{m-1} a^i f(n/b^i) &\ge A_1 n^k \sum_{i=0}^{m-1} (\log n - i \log b)^p \\
&\ge A_1 n^k \frac{\int_0^{m\log b} t^p \mathrm{d}t}{\log b} \\
&= \frac{A_1}{(p+1)\log b} n^k \log^{p+1} n \\
&\ge \frac{A_1}{(p+1)\log b} n^k \log^{p+1} n
\end{aligned} \tag{5}$$

从而 $T(n)=\Theta\left(n^k\log^{p+1}n\right), n=b^m$，对于 $b^{m-1}<n<b^m$，证明完全类似于第 1 问。 ■

3. 不妨设 $f(n)\geq B_1 n^{k+\epsilon}$，则由 2知 $T(n)=a^mT(1)+\sum_{i=0}^{m-1}a^if(n/b^i)$，其中 $n=b^m$，而 $a^mT(1)=\Theta(n^k)$，

$$\begin{aligned}\sum_{i=0}^{m-1}a^if(n/b^i)&\leq\sum_{i=1}^{m-1}a^i(\frac{c}{a})^if(n)\\&\leq\frac{1}{1-c}f(n)\end{aligned}\tag{6}$$

$$\sum_{i=0}^{m-1}a^if(n/b^i)\geq\sum_{i=1}^{m-1}a^i(\frac{c}{a})^if(n)\geq f(n)\tag{7}$$

从而 $T(n)=\Theta\left(f(n)\right), n=b^m$，对于 $b^{m-1}<n<b^m$，证明完全类似于第 1 问。 ■

5.3 Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n\log n)$ invocations of the equivalence tester.

我们把问题分成规模相似的两部分 A,B，其中 $|A|=\lceil\frac{n}{2}\rceil,|B|=\lfloor\frac{n}{2}\rfloor$，对于子问题 A,B 我们可以递归求解。

而把 A,B 链接起来，我们只需要注意到如下事实：若在 $A+B$ 中有半数以上相同，那么一定在 A 或 B 中也有半数以上相同。所以我们只需要取出 A 中超过半数的 (若存在) 那个账户与 B 中的元素逐一比较，并记下总数 (A 中的加上 B 中的)；取出 B 中超过半数的 (若存在) 那个账户与 A 中的元素逐一比较，并记下总数 (A 中的加上 B 中的)，若这两个总数有一个不小于 $\lceil\frac{n}{2}\rceil$，则返回该账户和该总数。

我们假设 n 个银行卡为 $A(1),\cdots,A(n)$，min,max 为下标，$Find(min,max)$ 为找 min,max 之间有没有超过 $n/2$ 等价账户，如果有返回该等价账户和数目。算法如算法 1。

可见如果设计算量为 $T(n)$，则有 $T(n)=2T(\frac{n}{2})+O(n)$，于是由主定理可知，$T(n)=O(n\log n)$ ■

function FIND(min, max)
  $n = max - min$
  if $n = 0$ **then return** 账户 $A(min)$,1
  end if
  $k = \lfloor \frac{n+1}{2} \rfloor$
  (A, t_1)=FIND$(min, min + k)$,(B, t_2)=FIND$(min + k + 1, max)$
  if $A \neq None$ **then** 将 A 与集合 $A(min + k + 1), \cdots, A(max)$ 逐一对比，若相同的数目 t_1' 超过了 $\lceil \frac{n+1}{2} \rceil - t_1$，则 **return** 账户 A,$t_1 + t_1'$
  end if
  if $B \neq None$ **then** 将 B 与集合 $A(min), \cdots, A(min + k)$ 逐一对比，若相同的数目 t_2' 超过了 $\lceil \frac{n+1}{2} \rceil - t_2$，则 **return** 账户 B,$t_2 + t_2'$
  end if
  return $None, None$
end function

Algorithm 1: Find(min,max)

5.4 Help them out by designing an algorithm that computes all the forces F_j in $O(n \log n)$ time.

令 $a = (q_1, q_2, \cdots, q_n), b = ((n-1)^{-2}, (n-2)^{-2}, \cdots, \frac{1}{4}, 1, 0, -1, -\frac{1}{4}, \cdots, -(n-1)^{-2})$，则计算可知，$(a * b)_{n+j}$ 位为 $\sum_{i<j} \frac{q_i}{(j-i)^2} - \sum_{i>j} \frac{q_i}{(j-i)^2}$，其中 $j = 1 : n$，于是通过一次 FFT 计算即可得到 $F_j = \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2} = Cq_j(a * b)_{n+j}$，计算量为 $O(n \log n)$。 ■