Algorithm: Homework #1

Due on Oct. 8, 2020 at 3:10pm

Professor HaiSheng Tan
Ch 1

李昱祁　　　　PB18071496

Problem 1

考虑以下查找问题:
输入: n 个数的一个序列 A = a1,a2,...,an 和一个值 v.
输出: 下标 i 使得 v=A[i] 或者当 v 不在 A 中出现时,v 为特殊值 NIL.
(a). 写出线性查找的伪代码, 它扫描整个序列来查找 v. 使用一个 Loop Invariant (循环不变式) 来证明你的算法是正确的.
(b). 假定 v 等可能的为数组中的任意元素，平均需要检查序列的多少元素? 最坏情况又如何呢? 用 Θ 记号给出线性查找的平均情况和最坏运行时间.

Part (a)

Linear-Search(A, n) 的伪代码：

```
function LINEAR-SEARCH(A, n)
    for j ∈ [1, n] do
        if A[j] = v then
            i = j
            return i
        end if
    end for
    return NIV
end function
```

Algorithm 1: Linear-Search

使用的循环不变式：对于每层循环的 j 值，不存在正整数 k 满足 $k < j$ 且 $A[k] = v$
满足以下三点：
* 初始化：第一次迭代前，$j = 1$，小于 j 的正整数不存在

* 保持：若某次迭代前为真，则对于 $k < j$ 的 k，都有 $A[k] \neq v$; 且若还有下层循环，说明当前循环的 j 值也满足 $A[j] \neq v$; 那么对于下层循环，当然不存在 $k < j + 1$ 且 $A[k] = v$ 的 k

* 终止：导致 for 循环执行到 $endfor$ 的的条件是 $j > n$，在循环不变式中将 j 用 $n + 1$ 代替，那么不存在正整数 k 满足 $k < n + 1$ 且 $A[k] = v$，即对所有正整数 $k \in [1, n]$ 都有 $A[k] \neq v$，此时输出 NIV

以上即证明了该算法的正确性

Part (b)

平均检查元素数 $= \frac{1}{n} * \sum_1^n n = \frac{1}{n} * \frac{n(n+1)}{2} = \frac{n+1}{2}$，而最坏情况检查元素数为 n

则平均时间复杂度 $t_{average} = \Theta(n)$

最坏情况下 $t_{worst} = \Theta(n)$

Problem 2

假定 f (n) 与 g(n) 都是渐进非负函数.
判断下列等式或陈述是否一定是正确的，并简要解释你的答案
a f(n)=O($f(n)^2$).
b f(n)+g(n)=Θ(max(f(n),g(n))).
c f(n)+O(f(n)) = Θ(f(n)).
d if f (n) = Ω(g(n)), then f (n) = o(g(n)). (注意是小 o)

Part (a)
不一定正确。比如：若 $f(n) = \frac{1}{n}$ 时，有 $f(n) \neq O(\frac{1}{n^2})$

Part (b)
正确。存在正整数 n_0，使得对所有 $n \geq n_0$，
都有 $f(n) + g(n) \leq max(f(n), g(n)) + max(f(n), g(n)) = 2max(f(n), g(n))$
所以 $f(n) + g(n) = O(max(f(n), g(n))$

同理，存在正整数 n_1，使得对所有 $n \geq n_1$，
都有 $f(n) + g(n) = max(f(n), g(n)) + min(f(n), g(n)) \geq max(f(n), g(n))$
所以 $f(n) + g(n) = \Omega(max(f(n), g(n))$

综上，由课本定理 3.1，$f(n) + g(n) = \Theta(max(f(n), g(n))$

Part (c)
正确。存在正整数 n_0，使得对所有 $n \geq n_0$，
都有 $0 \leq O(f(n)) \leq cf(n)$，其中 $c \geq 1$ 为一常数

所以 $f(n) + O(f(n)) \leq (c + 1)f(n)$，则 $f(n) + O(f(n)) = O(f(n))$，$(n \geq n_0)$
并且 $0 \leq f(n) \leq f(n) + O(f(n))$，则有 $f(n) + O(f(n)) = \Omega(f(n))$

综上，由课本定理 3.1，$f(n) + O(f(n)) = \Theta(f(n))$

Part (d)
不一定正确。举反例如下：
设 $f(n) = g(n) = n^2$

则显然有 $f(n) = O(g(n))$，但 $f(n) \neq o(g(n))$
因为 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1 \neq 0$

Problem 3

证明 lg(n!) $=\Theta$(nlg(n))(课本等式 3.19), 并证明 n! $=\omega(2^n)$ 且 n! $=$ o(n^n).

证明. 由题意得

$lg(n!)=\sum_1^n lg(i)\leq nlg(n)$
所以 $lg(n!)=O(nlg(n))$

由 Stirling 公式，在 $n\to\infty$ 时，有：
$lg(n!)=lg(\sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n})))=\frac{1}{2}lg(2\pi n)+nlg(n)-nlg(e)+lg(1+\Theta(1/n))$

当 $n\to\infty$ 时，$\frac{1}{2}lg(2\pi n)$、$nlg(e)$ 都是 $nlg(n)$ 的低阶无穷大量，
所以存在常数 c(例如 $c=\frac{1}{2}$) 和正整数 n_0，使得：

对所有 $n\geq n_0$(且 n 很大)，都有
$0\leq c\cdot nlg(n)\leq lg(\sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n})))=\frac{1}{2}lg(2\pi n)+nlg(n)-nlg(e)+lg(1+\Theta(1/n))=lg(n!)$
所以 $lg(n!)=\Omega(nlg(n))$

综上，由课本定理 3.1，$lg(n!)=\Theta(nlg(n))$

下面继续运用 Stirling 公式证明：

$\lim_{n\to\infty}\frac{n!}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}(\frac{n}{e})^n}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\cdot n^n}{(2e)^n}$
其中，$\lim_{n\to\infty}\sqrt{2\pi n}=\infty$，并且 $\lim_{n\to\infty}\frac{n^n}{(2e)^n}=\infty$，所以：$\lim_{n\to\infty}\frac{n!}{2^n}=\infty$

即 n! $=\omega(2^n)$

$\lim_{n\to\infty}\frac{n!}{n^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}(\frac{n}{e})^n}{n^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\cdot n^n}{e^n\cdot n^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}}{e^n}=0$

即 n! $=$ o(2^n)

□

Problem 4

使用代入法证明 T (n) = T (n/2) + 1 的解为 O(lgn).

Solution

当 $n = 1$ 时，$T(1) = \Theta(1)$，此时条件成立
则当 $n = k$ 时，由归纳假设知对于所有 $j < k$，有 $T(j) = O(lg(j))$，即 $T(j) \leq clg(j)$
则对于 $T(k)$，有

$T(k) = T([k/2]) + 1$
$\leq c_1 \cdot lg([k/2]) + 1$
$= c_1 \cdot lg(k) - c_1 \cdot lg(2) + 1$
$\leq c_2 \cdot lg(k)$

所以 $T(k) = O(lg(k))$，归纳成立，题干中所给式子的解为 O(lgn)

Problem 5

对递归式 T(n) = T(n−a)+T(a)+cn, 利用递归树给出一个渐进紧确解, 其中 $a \geq 1$ 和 $c > 0$ 为常数.

证明. 由题意得

画出递归树如下:

$$\begin{array}{c} T(a)+cn \\ | \\ T(n-a) \end{array}$$

进一步递归为:

$$\begin{array}{c} T(a)+cn \\ | \\ T(a)+c(n-a) \\ | \\ T(n-2a) \end{array}$$

逐步递归...

最终得到递归树如下:

$$\begin{array}{c} T(a)+cn \\ | \\ T(a)+c(n-a) \\ | \\ T(a)+c(n-2a) \\ | \\ \vdots \\ | \\ T(a)+c(n-(\frac{n}{a}-2)a) \\ | \\ T(a) \end{array}$$

树上一共有 n 个结点，将将他们相加的到：

$T(n) = \frac{n}{a}T(a) + \sum_{i=0}^{\frac{n}{a}-2} c(n-ia) = \frac{n}{a}T(a) + (\frac{n}{a}-1)\frac{cn+2ca}{2} = \frac{c}{2a}n^2 + (\frac{c}{2} + \frac{T(a)}{a})n - ac$

可以看出，其中最高阶的无穷大量为 n^2，可以得出 $T(n) = \Theta(n^2)$

□

Problem 6

对下列递归式, 使用主方法求出渐进紧确解:
(a). T(n) = 2T(n/4)+$\sqrt{n}$.
(b). T(n) = 2T(n/4)+n^2 .

Solution

通式为 $T(n) = aT(\frac{n}{b}) + f(n)$

Part (a)

对于此递归式，$a = 2$，$b = 4$，$log_b a = log_4 2 = \frac{1}{2}$

而 $f(n) = \sqrt{n}$，即 $f(n)$ 与 $n^{log_a b}$ 增长速率相同，且 $f(n) = \Theta(\sqrt{n}) = \Theta(\sqrt{n} lg^0 n)$

所以 $T(n) = \Theta(\sqrt{n} \cdot lg^{0+1} n) = \Theta(\sqrt{n} \cdot lgn)$

Part (b)

对于此递归式，$a = 2$，$b = 4$，$log_b a = log_4 2 = \frac{1}{2}$

而 $f(n) = n^2$，即 $f(n)$ 比 $n^{log_a b}$ 增长速率快，且 $af(n/b) = a \cdot (n/b)^2 = \frac{a}{b^2} \cdot n^2 = \frac{1}{8} \cdot n^2$

则 $\exists c < 1$，使得 $af(n/b) = \frac{1}{8} \cdot n^2 < c \cdot n^2 = c \cdot f(n)$

所以 $T(n) = \Theta(f(n)) = \Theta(n^2)$

Problem 7

主方法能应用于递归式 $T(n) = 4T(n/2) + n^2lgn$ 吗? 请说明为什么可以或者为什么不可以. 给出这个递归式的一个渐进上界.

不可以，理由如下:

此递归式中，$a = 4$，$b = 2$，$log_ab = log_24 = 2$，$f(n) = n^2lg(n)$

* 对于任何 $\varepsilon > 0$，$f(n) = n^2lg(n)$ 的渐进上界都不可能是 $O(n^{2-\varepsilon})$

情形 1 不成立

* $f(n) = n^2lg(n) \neq \Theta(n^2)$

情形 2 不成立

* 由微积分知识可知，对 $\forall\varepsilon > 0$，n^ε 都是 $lg(n)$ 的高阶无穷大量，即 $f(n) = \Omega(n^{2+\varepsilon})$ 不可能成立、、
情形 3 不成立

综上，不能用主方法求解该递归式

给出一个渐近上界为 $O(n^3)$，简单证明如下:

当 $n = 1$ 时，$T(1) = \Theta(1)$，成立

当 $n = k$ 时，若对于 $\forall j < k$，都有 $T(j) = O(j^3)$

则
$T(k) = 4T(k/2) + k^2lg(k)$
$\leq 4c_1 \cdot (k/2)^3 + k^2$
$= \frac{1}{2}c_1 \cdot k^3 + k^2$
$\leq c_2k^3$

所以 $T(k) = O(k^3)$ 也成立

由归纳知，$O(n^3)$ 为 $f(n)$ 一个渐近上界