CS325: Analysis of Algorithms, Fall 2016

Practice Assignment 1 Solution

Problem 1. For each of the following, indicate whether $f = O(g)$, $f = \Omega(g)$ or $f = \Theta(g)$.

(a) $f(n) = \Theta(g)$.

(b) $f(n) = \Omega(g)$.

(c) $f(n) = O(g)$.

(d) $f(n) = \Omega(g)$.

(e) $f(n) = O(g)$.

Problem 2. Let $h(n) = \max(f(n), g(n))$, and let $s(n) = f(n) + g(n)$. We want to show that $h(n) = \Theta(s(n))$, that is there are constants $c_1, c_2, n_0 > 0$ such that $c_1 s(n) \leq h(n) \leq c_2 s(n)$. We show that $c_1 = 1/2$, $c_2 = 1$ and $n_0 = 1$ ensure the desired inequalities.

By the definition of $h(n)$, we have $h(n) \geq f(n)$ and $h(n) \geq g(n)$, therefore:

$$\frac{s(n)}{2} \leq \frac{f(n) + g(n)}{2} \leq \frac{h(n) + h(n)}{2} = h(n), \tag{1}$$

for all $n \geq n_0 = 1$.

On the other hand, since both $f(n)$ and $g(n)$ are nonnegative, we have:

$$h(n) \leq f(n) + g(n) = s(n), \tag{2}$$

for all $n \geq n_0 = 1$.

Problem 3.

(a) The idea is to:

(i) flip the first k top pancakes so that the largest pancake is on the top

(ii) flip the whole stack so that the largest pancake is in the bottom

(iii) recursively sort the top $n - 1$ pancakes

Below is the pseudocode ($A[1 \cdot\cdot n]$ contains the sizes of the input pancake stack. $A[1]$ and $A[n]$ are the topmost and bottommost pancakes, respectively.):

```
PancakeFlipper($A[1 \cdots n]$)
    if $n > 1$
        Let $k$ be the index if the largest pancake
        Flip($A[1 \cdots k]$)
        Flip($A[1 \cdots n]$)
        PancakeFlipper($1 \cdots n - 1$)
```

proof of correctness. We use induction to show that the algorithm above correctly sorts the pancake stack.

Base case: PancakeFlipper correctly sorts a stack of zero pancakes by doing nothing.

Induction Hypothesis: For any $N < n$, PancakeFlipper correctly sorts a stack of N pancakes.

Induction Step: PancakeFlipper($A[1 \cdots n]$) correctly sorts $A[1 \cdots n]$. Suppose $n \geq 1$, otherwise the base case implies the statement. Since $A[k]$ is the largest pancake, we have that the largest pancake is the topmost pancake after the first flip. Consequently, the largest pancake is the bottommost pancake after the second flip. Finally, the recursive call PancakeFlipper($A[1 \cdots n-1]$) sorts the rest of the pancakes and does not change the position of the largest pancake by the induction hypothesis. Thus, everything is sorted after this recursive call. □

Number of flips. Let $F(n)$ be the number of flips that the PancakeFlipper($A[1 \cdots n]$) performs. We have:

$$\begin{aligned} F(1) &= 0 \\ F(n) &= 2 + F(n-1) \end{aligned} \tag{3}$$

We solve the recurrence relation to find the closed formula for $F(n)$.

$F(n) = F(n-1) + 2 = (F(n-2) + 2) + 2 = F(n-2) + 2 * 2 = F(1) + (n-1) * 2 = 2n - 2.$

(A more careful algorithm would chose $F(2) = 1$ as the base case to obtain $F(n) = 2n - 3$.)

(b) The idea is similar to (a).

(i) flip the first k top pancakes so that the largest pancake is on the top
(ii) if the burned side of the largest pancake faces down, flip the largest pancake
(iii) flip the whole stack so that the largest pancake is in the bottom
(iv) recursively sort the top $n - 1$ pancakes

Below is the pseudocode ($A[1 \cdots n]$ contains the sizes of the input pancake stack. $A[1]$ and $A[n]$ are the topmost and bottommost pancakes, respectively.):

```
BurnedPancakeFlipper($A[1 \cdots n]$)
    if $n > 0$
        Let $k$ be the index if the largest pancake
        Flip($A[1 \cdots k]$)
        if the burned side of $A[1]$ faces down
        Flip($A[1 \cdot\cdot 1]$)
    Flip($A[1 \cdot\cdot n]$)
    BurnedPancakeFlipper($1 \cdot\cdot n-1$)
```

proof of correctness. We use induction to show that the algorithm above correctly sorts the pancake stack.

Base case: BurnedPancakeFlipper correctly sorts a stack of zero pancakes by doing nothing.

Induction Hypothesis: For any $N < n$, PancakeFlipper correctly sorts a stack of N pancakes.

Induction Step: PancakeFlipper($A[1 \cdot\cdot n]$) correctly sorts $A[1 \cdot\cdot n]$. Suppose $n \geq 1$, otherwise the base case implies the statement. Since $A[k]$ is the largest pancake, we have that the largest pancake is the topmost pancake after the first flip. After the "if statement" we have that the largest pancake is the topmost pancake and its burned side faces up. Consequently, the largest pancake is the bottommost pancake with its burned side facing down after the last flip. Finally, the recursive call BurnedPancakeFlipper($A[1 \cdot\cdot n-1]$) sorts the rest of the pancakes, ensures their burned sides face down, and does not alter the position of the largest pancake by the induction hypothesis. Thus, everything is sorted with burned side facing down after this recursive call. □

Number of flips. Let $H(n)$ be the number of flips performed by BurnedPancakeFlipper($A[1 \cdot \cdot n]$). Then, we have:

$$\begin{aligned} H(0) &= 0 \\ H(n) &\leq 3 + H(n-1) \end{aligned} \tag{4}$$

We solve the recurrence relation to find the closed formula for $H(n)$.

$$H(n) \leq H(n-1) + 3 \leq (H(n-2) + 3) + 3 \leq H(n-2) + 2 * 3 \leq H(0) + (n) * 3 = 3n.$$

(Can you ensure a smaller number of flips by choosing a different base case?)