CS 3000: Algorithms & Data — Spring 2024

Sample Solutions to Homework 1
Due Monday January 22 at 11:59pm via Gradescope

Name: Alexander Rosenthal
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This homework is due Monday January 22 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typed. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. We recommend that you use LaTeX, in which case it would be best to use the source file for this assignment to get started. Your submitted file must be a PDF file.
- We encourage you to work with your classmates on the homework problems, but also urge you to attempt all of the problems by yourself first. If you do collaborate, you must write all solutions by yourself, in your own words. Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

Problem 1. *(4 + 8 = 12 points) What does this code do?*

You encounter the following mysterious piece of code.

Algorithm 1: Mystery Function

```
Function F(n)):
    If n = 0 :
        Return (2,1)
    Else
        b ← 1
        For i from 1 to n
            b ← 2b
        (u,v) ← F(n−1)
        Return (u + b, v · b)
```

(a) What are the results of $F(1)$, $F(2)$, $F(3)$, and $F(4)$?

Solution:
$F(1) = (4, 2)$
$F(2) = (8, 8)$
$F(3) = (16, 64)$
$F(4) = (32, 1024)$

(b) What does the code do in general, when given input integer $n \geq 0$? Prove your assertion by induction on n.

Solution: The function above returns a value of $F(n) = (2^{n+1}, 2^{\frac{n^2+n}{2}})$
Proof: Let $P(n)$ be the statement that $F(n) = (2^{n+1}, 2^{\frac{n^2+n}{2}})$
Base Case: $F(0) = (2^{0+1}, 2^{\frac{0^2+0}{2}} = (2^1, 2^0) = (2,1)$ which can be seen in the return statement at the beginning of the function. Therefore $P(0)$ is true.
Inductive Hypothesis (*IH*): Suppose $P(n-1)$ is true for some $n \geq 1$
Inductive Step: Consider executing $F(n)$:
The for loop doubles the value of b n times, therefore $b = 2^n$.
$(u, v) = F(n-1)$ so by the *IH*, u = $2^{(n-1)+1} = 2^n$ and v = $2^{\frac{(n-1)^2+(n-1)}{2}} = 2^{\frac{n^2-2n+1+n-1}{2}} = 2^{\frac{n^2-n}{2}}$
The function returns $(u+b, v*b) = (2^n + 2^n, 2^{\frac{n^2-n}{2}} * 2^n) = (2(2^n), 2^{\frac{n^2-n}{2}+n}) = (2^{n+1}, 2^{\frac{n^2-n}{2}+\frac{2n}{2}}) = (2^{n+1}, 2^{\frac{n^2+n}{2}})$
This shows that $P(n-1) \implies P(n)$, thus the claim must be true for all $n \geq 0$

Problem 2. *(12 points) Making exact change*

In the country of Perfect Squares, all coins are in denominations that are perfect squares i.e. i^2 for some integer i. You need to buy a jacket whose price is an integer $n \geq 1$. The country is obsessed with being perfect and you can only buy an item if you pay the exact price. You happen to have exactly one coin of value i^2 for each integer $i \geq 2$. Additionally you have 4 coins of value 1 each. Use induction to show that you can pay the exact cost of the jacket using your coins.

Solution:
Lemma: The statement: $2n+1 < n^2$ is true for all $n > 2$
Base case: $2(3)+1 < (3)^2 \implies 7 < 9$, the statement is true for $n = 3$
Inductive Step: Suppose the statement is true for some $n > 2$, then for $k = n+1$:
$2(k+1)+1 < (k+1)^2, \implies 2k+3 < k^2+2k+1 \implies 2 < k^2$ and since $k > 2$, the inequality is true.
Therefore the statement is true.

Proof:
To make perfect change for a jacket of price n, suppose we start with a coin that is the largest perfect square less than or equal to n. This coin has a price of m^2.
Inductive Hypothesis *IH*: Assume that for all $n < (m+1)^2$, we can make perfect change for n.
Base Case: The smallest possible value of m is 2, which, by the *IH*, gives us $n < (2+1)^2 \implies n < 9$.
We can show that for every value of $1 \leq n < 9$, we can make exact change:
$n = 1$: One 1 value coin
$n = 2$: Two 1 value coins
$n = 3$: Three 1 value coins
$n = 4$: One coin of value $2^2 = 4$
$n = 5$: One coin of value 4 and one 1 value coin
$n = 6$: One coin of value 4 and two 1 value coins
$n = 7$: One coin of value 4 and three 1 value coins
$n = 8$: One coin of value 4 and four 1 value coins

Inductive Step: Now suppose $k = m+1$ for some $m \geq 2$. To make change for a jacket of price n we consider 2 cases:
Case 1: If $n < k^2$, then by the *IH* we can make exact change
Case 2: If $k^2 \leq n < (k+1)^2$, then we will start with a coin of value k^2 (the largest perfect square $\leq n$). This leaves us with $n - k^2$ value to make change for.
$n < (k+1)^2 \implies n < k^2 + 2k + 1 \implies n - k^2 < 2k + 1$
By the lemma proved above, $2k+1 < k^2$, therefore $n - k^2 < k^2$ and by the *IH* we can make exact change for the remaining $n - k^2$ value without using a coin of k^2.

Therefore we can make change for all jackets of price $n \geq 1$.

Problem 3. *(12 points) More induction practice*

Consider the following function f defined on the nonnegative integers.

$$\begin{aligned} f(0) &= 3 \\ f(1) &= 4 \\ f(n) &= 3f(n-2) + 2f(n-1), \text{ for } n \geq 2 \end{aligned}$$

Prove by induction that $f(n) = (5 \cdot (-1)^n + 7 \cdot 3^n)/4$ for all integer $n \geq 0$.

Solution:
Let $P(n)$ be the statement that $f(n) = \frac{5(-1)^n+7(3^n)}{4}$ for all integers $n \geq 0$
Base case: $f(0) = \frac{5(-1)^0+7(3^0)}{4} = \frac{5+7}{4} = \frac{12}{4} = 3$
$f(1) = \frac{5(-1)^1+7(3^1)}{4} = \frac{5(-1)+7(3)}{4} = \frac{-5+21}{4} = \frac{16}{4} = 4$
Inductive Hypothesis *IH*: Suppose $P(k)$ is true for all integers $0 \leq k < n$
Inductive Step: By the recursive definition above, $f(n+1) = 3f(n-1)+2f(n)$. By the IH, we have:
$f(n+1) = \frac{3*(5(-1)^{n-1}+7(3^{n-1}))}{4} + \frac{2*(5(-1)^n+7(3^n))}{4} = \frac{15(-1)^{n-1}+21(3)^{n-1}+10(-1)^n+14(3)^n}{4} = \frac{\frac{15(-1)^n}{-1}+\frac{21(3)^n}{3}+10(-1)^n+14(3)^n}{4} =$
$\frac{-15(-1)^n+10(-1)^n+7(3)^n+14(3)^n}{4} = \frac{-5(-1)^n+21(3)^n}{4} = \frac{5(-1)(-1)^n+7(3)(3)^n}{4} = \frac{5(-1)^{n+1}+7(3)^{n+1}}{4}$
Therefore, $P(n)$ is true for all integers $n >= 0$

Problem 4. *(12 points) Growth of functions*

Arrange the following functions in order from the slowest growing function to the fastest growing function. Note that $\lg n = \log_2 n$.

$$n^{2/3} \qquad n + \lg n \qquad 2^{\sqrt{\lg n}} \qquad (\lg n)^{\lg n}$$

Justify your answers. Specifically, if your order is of the form

$$f_1 \qquad f_2 \qquad f_3 \qquad f_4,$$

you should establish $f_1 = O(f_2)$, $f_2 = O(f_3)$, and $f_3 = O(f_4)$. For each case, your justification can be in the form of a proof from first principles or a proof using limits, and can use any of the facts presented in the lecture or the text. (Hint: It may help to plot the functions and obtain an estimate of their relative growth rates. In some cases, it may also help to express the functions as a power of 2 and then compare.)

Solution: slowest - $\quad 2^{\sqrt{\lg n}} \qquad n^{2/3} \qquad n + \lg n \qquad (\lg n)^{\lg n} \quad$ - fastest

First, we show that $2^{\sqrt{\lg n}} = O(n^{2/3})$:
$n^{2/3}$ can be rewritten as $2^{(\lg n)^{2/3}} = 2^{\frac{2}{3}\lg n}$
$\sqrt{\lg n} \leq \frac{2}{3}\lg n$ for $n \geq 5$, therefore $2^{\sqrt{\lg n}} \leq c \cdot 2^{\frac{2}{3}lgn}$ for $c = 1$ and $n \geq 5$
Thus $2^{\sqrt{\lg n}} \leq c \cdot n^{2/3}$ for $c = 1$ and $n \geq 5$ so $2^{\sqrt{\lg n}} = O(n^{2/3})$

Next, we show that $n^{2/3} = O(n + \lg n)$:
First, since $\lim_{n\to\infty} \frac{n^{2/3}}{n} = \lim_{n\to\infty} \frac{1}{n^{1/3}} = \frac{1}{\infty} = 0$, we know that $n^{2/3} = O(n)$
And since $n \leq c(n + \lg n)$ for all $n \geq 1$ and $c = 1$, we know $n = O(n + \lg n)$
Finally, since $n^{2/3} = O(n) = O(n + \lg n)$, we see that $n^{2/3} = O(n + \lg n)$

Last, we show that $n + \lg n = O((\lg n)^{\lg n})$:
We know that $\lg n < n \implies n + \lg n < n + n \implies n + \lg n < 2n$, therefore $n + \lg n = O(n)$
We know that $2 \leq \lg n$ for all $n \geq 4$, raising both sides to the power $\lg n$ gives us $2^{\lg n} \leq (\lg n)^{\lg n}$ which simplifies to $n \leq (\lg n)^{\lg n}$ for all $n \geq 4$. Therefore, $n = O((\lg n)^{\lg n})$
Finally, since $n + \lg n = O(n) = O((\lg n)^{\lg n})$, we see that $n + \lg n = O((\lg n)^{\lg n})$

Problem 5. *($2 \times 5 = 10$ points) Properties of asymptotic notation*

Let $f(n)$, $g(n)$, and $h(n)$ be asymptotically positive and monotonically increasing functions.

(a) Using the formal definition of the O and Ω notation, prove that if $f(n) = O(h(n))$ and $g(n)^2 = \Omega(h(n)^2)$, then $f(n) = O(g(n))$.

Solution:
If $g(n)^2 = \Omega(h(n)^2)$, then $g(n)^2 \geq c \cdot h(n)^2$ for some $c \geq 0$ and all $n \geq n_o$, following this:
$\sqrt{g(n)^2} \geq \sqrt{c \cdot h(n)^2} \implies g(n) \geq \sqrt{c}h(n) \implies \frac{1}{\sqrt{c}}g(n) \geq h(n)$
And since $f(n) = O(h(n))$, then $f(n) \leq c' \cdot h(n)$, which gives us:
$\frac{1}{c'}f(n) \leq h(n) \leq \frac{1}{\sqrt{c}}g(n) \implies f(n) \leq \frac{c'}{\sqrt{c}}g(n)$
$f(n) \leq c''g(n)$ for some $c'' \geq 0$ and $n \geq n_o$, Therefore $f(n) = O(g(n))$

(b) Give distinct functions f and g satisfying both $f(n) = \Theta(g(n))$ and $3^{f(n)} = \Theta(3^{g(n)})$.

Give distinct functions f and g satisfying $f(n) = O(g(n))$ yet $3^{f(n)} \neq O(27^{g(n)})$.

Solution:
Part 1: $f(n) = n, \quad g(n) = n + 1$
Part 2: $f(n) = 2\log_3 n, \quad g(n) = \log_{27} n$

Problem 6. *(12 points) Determining the largest element in one list that is not present in another list*

We have learned that Mergesort sorts an array of n numbers in $O(n \log n)$ time and Binary Search determines if a given number is present in a sorted array of n numbers in $O(\log n)$ time.

Describe an $O(n \log n)$ time algorithm that takes as input two arrays A and B with n elements each (not necessarily sorted) and determines the largest number in B that is not in A. If all elements of B are in A, then return "All elements of B are in A".

Your algorithm should use mergesort and binary search and should not use hash tables. Give your algorithm in pseudocode. Justify the running time of your algorithm.

Solution:

```
Function F(A, B)):
    A ← MergeSort(A)
    B ← MergeSort(B)
    n ← len(B)
    For i from n down to 1
        If BinarySearch(B[i] in A) is not found :
            Return B[i]
    Return "All elements of B are in A"
```

Both calls to MergeSort take $O(n \lg n)$ time, giving us $2n \lg n$ running time.
The for loop iterates n times, and each iteration makes a call to BinarySearch, which takes $O(\lg n)$ time. So the for loop has a $O(n \lg n)$ runtime.
This gives us a total run time of $3n \lg n = O(n \lg n)$