Team Members: Danny Zou, Aaron Verkleeren

CSCI 2200 — Foundations of Computer Science (FoCS)
Homework 3 (document version 1.0)

Overview

- This homework is due by 11:59PM on Thursday, October 19
- You may work on this homework in a group of up to four students; unlike recitation problem sets, **your teammates may be in any section**
- You may use at most **two** late days on this assignment
- Please start this homework early and ask questions during office hours and at your October 18 recitation section; also ask questions on the Discussion Forum
- Please be concise in your written answers; even if your solution is correct, if it is not well-presented, you may still lose points
- You can type or hand-write (or both) your solutions to the required graded problems below; **all work must be organized in one PDF that lists all teammate names**
- You are strongly encouraged to use LaTeX, in particular for mathematical symbols; see the corresponding `hw1.tex` file as a starting point and example

Grading

- For each assigned problem, a grade of 0, 1, or 2 is assigned as follows: 0 indicates no credit; 1 indicates half credit; and 2 indicates full credit
- No credit is assigned if a problem is not attempted or minimal work/progress is shown
- Half credit is assigned if a strong attempt was made toward a solution and/or only part of the problem was attempted or solved
- Full credit is assigned for a perfect or nearly perfect solution, i.e., only one or two minor typos/mistakes at most

Warm-up exercises

The problems below are good practice problems to work on. Do not submit these as part of your homework submission. **These are ungraded problems.**

- **Problem 7.9.**
- **Problem 7.11.**
- **Problem 7.12(a-b).**
  (See Problem 7.28 for hints.)
- **Problem 7.21.**
- **Problem 7.41.**
- **Problem 7.44.**
- **Problem 7.45(a-b,d-f).**
- **Problem 7.46.**
- **Problem 7.47.**
- **Problem 7.49.**
- **Problem 8.12(a-c).**
- **Problem 8.13.**
- **Problem 8.18.**

Graded problems

The problems below are required and will be graded.

- ***Problem 7.12(c).** (See Problem 7.28 for hints.)
- ***Problem 7.13(a).**
- ***Problem 7.19(d).**
- ***Problem 7.42.**
- ***Problem 7.45(c).**
- ***Problem 8.12(d).**
- ***Problem 8.14.**

As you might not have the required textbook yet, all of the above problems (both graded and ungraded) are transcribed in the pages that follow.

Graded problems are noted with an asterisk (*).

If any typos exist below, please use the textbook description.

- **Problem 7.9.** $G_0 = 0$, $G_1 = 1$, and $G_n = 7G_{n-1} - 12G_{n-2}$ for $n > 1$. Compute G_5.
  Show $G_n = 4^n - 3^n$ for $n \geq 0$.

- **Problem 7.11.** In each case tinker. Then, guess a formula that solves the recurrence, and prove it.

  (a) $P_0 = 0$, $P_1 = a$, and $P_n = 2P_{n-1} - P_{n-2}$, for $n > 1$.
  (b) $G_1 = 1$; $G_n = (1 - 1/n) \cdot G_{n-1}$, for $n > 1$.

- **Problem 7.12(a-b).** (See Problem 7.28 for hints.) Tinker to guess a formula for each recurrence and prove it. In each case, $A_1 = 1$ and for $n > 1$:

  (a) $A_n = 10A_{n-1} + 1$.
  (b) $A_n = nA_{n-1}/(n-1) + n$.

- ***Problem 7.12(c).** (See Problem 7.28 for hints.) Tinker to guess a formula for each recurrence and prove it. In each case, $A_1 = 1$ and for $n > 1$:

  (c) $A_n = 10nA_{n-1}/(n-1) + n$.

  $A_n = \begin{cases} 1 & \text{if } n = 1 \\ \frac{10nA_{n-1}}{(n-1)} + n & \text{if } n > 1 \end{cases}$

  $A_2 = \frac{20(1)}{1} + 2 = 22 = 2(10^1 + 1)$

  $A_3 = \frac{30(22)}{2} + 3 = 333 = 3(10^2 + 10^1 + 1)$

  $A_4 = \frac{40(333)}{3} + 4 = 4444 = 4(10^3 + 10^2 + 10^1 + 1)$

  $A_5 = \frac{50(4444)}{4} + 5 = 55555 = 5(10^4 + 10^3 + 10^2 + 10^1 + 1)$

  And so on...

  Observe that A(n) = $n(10^{n-1} + 10^{n-2} + ... + 10^0)$

  Define claim P(n) : A(n) = $n(10^{n-1} + 10^{n-2} + ... + 10^0)$ for all n > 1

  We prove by strong induction that P(n) is true for n > 1

  [Base Case] A(1) = $1(10^{1-1}) = 1(1) = 1$ $A_1 = 1$ — 1=1 True

  [Induction Step] We prove $P(1) \wedge\ P(2) \wedge\ ... \wedge\ P(n) \rightarrow P(n+1)$ for n > 1

  We must prove that $A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^0)$

  LHS : $A_{n+1} = \frac{10(n+1)}{n}(A_n) + (n+1)$

  $A_{n+1} = \frac{(10(n+1))(n(10^{n-1}+10^{n-2}+...+10^0))}{n} + n + 1$

  $A_{n+1} = (10(n+1))(10^{n-1} + 10^{n-2} + ... + 10^0) + n + 1$

  $A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^0) + n + 1$

  $A_{n+1} = (n+1)[(10^n + 10^{n-1} + ... + 10^1) + 1]$

  $A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^1 + 1)$

  $A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^1 + 10^0)$

$A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^0)$

Thus, $A_{n+1} = (n+1)(10^n + 10^{n-1} + ... + 10^0)$, as was to be shown

By induction, P(n) is true for all n > 1.

- ***Problem 7.13(a).** Analyze these very fast-growing recursions. [Hint: Take logarithms.]

  (a) $M_1 = 2$ and $M_n = aM_{n-1}^2$ for $n > 1$. Guess and prove a formula for M_n. Tinker, tinker.

$$M_n = \begin{cases} 2 & \text{if } n = 1 \\ aM_{n-1}^2 & \text{if } n > 1 \end{cases}$$

$M_2 = a(M_1^2) = a(2^2) = a(2^{2^1})$

$M_3 = a(M_2^2) = a((4a)^2) = 16a^3 = 2^4(a^3) = 2^{2^2}(a^3)$

$M_4 = a(M_3^2) = a((16a^3)^2) = 256a^7 = 2^8(a^7) = 2^{2^3}(a^7)$

$M_5 = a(M_4^2) = a((256a^7))^2 = 65536a^15 = 2^{16}(a^{15}) = 2^{2^4}(a^{15})$

And so on ...

We observe that M(n) = $2^{2^{n-1}} * a^{2^{n-1}-1}$

Define claim P(n) : M(n) = $2^{2^{n-1}} * a^{2^{n-1}-1}$ for all n > 1

We prove by strong induction that P(n) is true for n > 1

[Base Case] M(1) = $2^1 * a^{1-1} = 2$ $M_1 = 2$ — 2=2 True

[Induction Step] We prove $P(1) \wedge P(2) \wedge ... \wedge P(n) \rightarrow P(n+1) for n > 1$

We must prove that $M_{n+1} = 2^{2^n} * a^{2^n-1}$

LHS: $M_{n+1} = aM_n^2$

$M_{n+1} = a(2^{2^{n-1}} * a^{2^{n-1}-1})^2$

$M_{n+1} = a(2^{2*2^{n-1}} * a^{2*(2^{n-1}-1)})$

$M_{n+1} = a(2^{2^n} * a^{2^n-2})$

$M_{n+1} = 2^{2^n} * a^{2^n-1}$

Thus, $M_{n+1} = 2^{2^n} * a^{2^n-1}$, as was to be shown

By induction, P(n) is true for all n > 1.

- ***Problem 7.19(d).** Recall the Fibonacci numbers: $F_1, F_2 = 1$; and, $F_n = F_{n-1} + F_{n-2}$ for $n > 2$.

  (d) Prove that every third Fibonacci number, F_{3n}, is even.

$$F_n = \begin{cases} 1 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ F_{n-1} + F_{n-2} & \text{if } n > 2 \end{cases}$$

Define claim P(n), where F_{3n} is even

P(n) : F_{3n} is even

Now we want to prove P(n) to be true for all n ≥ 1

Proof by Induction

[Base Case] For n = 1, $F_1 = 1$ $F_2 = 1$ $F_3 = F_1 + F_2$ $F_3 = 1 + 1 = 2$ which is even, True

[Induction Step] Assume that P(n) is true, then we must prove that P(n) $\rightarrow$ P(n+1) for all n > 1

P(n+1) = $F_{3(n+1)} = F_{3n+3}$

Plug into $F_n = F_{n-1} + F_{n-2}$

$F_{3n+3} = F_{3n+2} + F_{3n+1}$

$F_{3n+2} = F_{3n+1} + F3n$

Subsitute, we get $F_{3n+3} = F_{3n+1} + F_{3n+1} + F_{3n}$

$F_{3n+3} = 2 * F_{3n+1} + F_{3n}$

We know from our induction hypothesis that F_{3n} is even.

$2 * F_{3n+1}$ is also even because it is a multiple of 2.

The sum of two even numbers will always come out to be even, thus proving

$F_{3n+3} = 2 * F_{3n+1} + F_{3n}$ to be even

Thus, we have proven our claim P(n) to be true for all n > 1, proving that every third Fibonacci number F_{3n} is even.

- **Problem 7.21.** Show that every $n \geq 1$ is a sum of distinct Fibonacci numbers, e.g., $11 = F_4+F_6$; $20 = F_3 = F_5+F_7$. (There can be many ways to do it, e.g., $6 = F_1+F_5 = F_2+F_3+F_4$.) [Hints: Greedy algorithm; strong induction.]

- **Problem 7.41.** Refer to the pseudocode on the right.

```
out=S([arr],i,j)
 if(j<i) out=0;
 else
  out=arr[j]+S([arr],i,j-1);
```

  (a) What is the function being implemented?
  (b) Prove that the output is correct for every valid input.
  (c) Give a recurrence for the runtime T_n, where $n = j - i$.
  (d) Guess and prove a formula for T_n.

- ***Problem 7.42.** Give pseudocode for a recursive function that computes 3^{2^n} on input n.
  (a) Prove that your function correctly computes 3^{2^n} for every $n \geq 0$.
  (b) Obtain a recurrence for the runtime T_n. Guess and prove a formula for T_n.

$$f(n) = \begin{cases} 3 & \text{if } n = 0 \\ f(n-1)^2 & \text{if } n \geq 1 \end{cases} \tag{1}$$

  (a) Prove that your function correctly computes 3^{2^n} for every $n \geq 0$.

Structural Induction Proof:

Base case: f(0) = 3 = 3^{2^0}

Induction hypothesis: $3^{2^n} = f(n)$, $f(n) = f(n-1)^2$

1. $3^{2^{n+1}} = f(n+1)$; need to prove this
2. $3^{2^{n+1}} = f(n)^2$; substitute the value of f(n+1) based on recursive definition
3. $3^{2^{n+1}} = (3^{2^n})^2$; substitute in induction hypothesis
4. $3^{2^{n+1}} = 3^{2^n 2}$; algebraic manipulation
5. $3^{2^{n+1}} = 3^{2^{n+1}}$; since both side are equal, we can conclude that the statement is true

(b) Obtain a recurrence for the runtime T_n. Guess and prove a formula for T_n.

Formula for runtime: $T_0 = 1$, $T_n = T_{n-1} + 1$

Claim: The runtime of the function T_n can be expressed as $T_n = O(n)$

Base case:

1. $T_0 = O(1)$, which is a constant time.

Induction hypothesis:

2. Assume that $T_n = O(n)$ for some integer $n \geq 0$

Induction steps:

3. $T_{n+1} = T_n + 1$; need to prove this claim
4. $T_{n+1} = O(n) + O(1)$; substitute in induction hypothesis
5. $T_{n+1} = O(n+1)$; claim proved successfully

Since we proved that increasing n by 1 will increase the runtime by 1, this means that the formula is proved true.

- **Problem 7.44.** We give two implementations of `Big(n)` from page 90 (`iseven(n)` tests if n is even).

(a)
```
out=Big(n)
 if(n==0) out=1;
 elseif(iseven(n))
    out=Big(n/2)*Big(n/2);
 else out=2*Big(n-1)
```

(b)
```
out=Big(n)
 if(n==0) out=1;
 elseif(iseven(n))
    tmp=Big(n/2); out=tmp*tmp;
 else out=2*Big(n-1)
```

(i) For each, prove that the output is 2^n and give a recurrence for the runtime T_n. (`iseven(n)` is two operations.)

(ii) For each, compute runtimes T_n for $n = 1, \ldots, 10$. Compare runtimes with Exercise 7.10 on page 90.

- **Problem 7.45(a-b,d-f).** Give recursive definitions for the set $\mathcal{S}$ in each of the following cases.

(a) $\mathcal{S} = \{0, 3, 6, 9, 12, \ldots\}$, the multiples of 3.

(b) $\mathcal{S} = \{1, 2, 3, 4, 6, 7, 8, 9, 11, \ldots\}$, the numbers which are not multiples of 5.

(d) The set of odd multiples of 3.

(e) The set of binary strings with an even number of 0's.

(f) The set of binary strings of even length.

- ***Problem 7.45(c).** Give recursive definitions for the set $\mathcal{S}$ in each of the following cases.

  (c) $\mathcal{S} = \{$ all strings with the same number of 0's and 1's $\}$ (e.g., 0011, 0101, 100101).

  Recursive Definition for $\mathcal{S}$:

  Base Cases:

  1. $\varepsilon \in \mathcal{S}$; (i.e., the empty string is in $\mathcal{S}$)

  Recursive Rules:

  2. $x \in \mathcal{S} \rightarrow x \cdot 01 \in \mathcal{S}$ and $x \cdot 10 \in \mathcal{S}$; In other words, for any string x in $\mathcal{S}$, you can create new strings in $\mathcal{S}$ by appending either "01" or "10" to the end of x.

  3. $x \in \mathcal{S} \rightarrow 01 \cdot x \in \mathcal{S}$ and $10 \cdot x \in \mathcal{S}$; In other words, for any string x in $\mathcal{S}$, you can create new strings in $\mathcal{S}$ by adding "01" or "10" to the beginning of x.

  4. $x \in \mathcal{S} \rightarrow 0 \cdot x \cdot 1 \in \mathcal{S}$ and $1 \cdot x \cdot 0 \in \mathcal{S}$; In other words, for any string x in $\mathcal{S}$, you can create new strings in $\mathcal{S}$ by adding "0" or "1" to the start and then the opposite number to the end of x.

- **Problem 7.46.** What is the set $\mathcal{A}$ defined recursively as shown? (By default, nothing else is in $\mathcal{A}$—minimality.)

  (1) $1 \in \mathcal{A}$

  (2) $x, y \in \mathcal{A} \rightarrow x + y \in \mathcal{A}$
  $x, y \in \mathcal{A} \rightarrow x - y \in \mathcal{A}$

- **Problem 7.47.** What is the set $\mathcal{A}$ defined recursively as shown? (By default, nothing else is in $\mathcal{A}$—minimality.)

  (1) $3 \in \mathcal{A}$

  (2) $x, y \in \mathcal{A} \rightarrow x + y \in \mathcal{A}$
  $x, y \in \mathcal{A} \rightarrow x - y \in \mathcal{A}$

- **Problem 7.49.** There are 5 rooted binary trees (RBTs) with 3 nodes. How many have 4 nodes?

- **Problem 8.12(a-c).** A set $\mathcal{P}$ of parenthesis strings has a recursive definition (right).

  (1) $\varepsilon \in \mathcal{P}$

  (2) $x \in \mathcal{P} \rightarrow [x] \in \mathcal{P}$
  $x, y \in \mathcal{P} \rightarrow xy \in \mathcal{P}$

(a) Determine if each string is in $\mathcal{P}$ and give a derivation if it is in $\mathcal{P}$.
(i) [[[]]]][(ii) [][[]][[]] (iii) [[][]]

(b) Give two derivations of [][][[]] whose steps are not a simple reordering of each other.

(c) Prove by structural induction that every string in $\mathcal{P}$ has even length.

- ***Problem 8.12(d).** A set $\mathcal{P}$ of parenthesis strings has a recursive definition (right).

(1) $\varepsilon \in \mathcal{P}$

(2) $x \in \mathcal{P} \to [x] \in \mathcal{P}$
$x, y \in \mathcal{P} \to xy \in \mathcal{P}$

(d) Prove by structural induction that every string in $\mathcal{P}$ is balanced.

Structural Induction proof for $\mathcal{P}$:

Base case:

1. ε has 0 open parenthesis and 0 close parenthesis, meaning it is balanced

2. [ ] has 1 open parenthesis and 1 close parenthesis, meaning it is balanced

Structural Induction Steps: $x, y \in \mathcal{P}$, x and y are balanced

Case 1: first recursive step

3. $x \in \mathcal{P} \to [x] \in \mathcal{P}$
4. $x = k(open) + k(close)$; rewrite x
5. $[x] = [k(open) + k(close)]$; add [] around both
6. $[x] = (k+1)(open) + (k+1)(close)$; since there are equal amounts of open and close, we know that [x] is balanced

Case 2: second recursive step

7. $x, y \in \mathcal{P} \to xy \in \mathcal{P}$
8. $x = k(open) + k(close), y = n(open) + n(close)$; since we know that x and y are both balanced, we can rewrite x and y
9. $xy = k(open) + k(close) + n(open) + n(close)$; we can rewrite xy as the concatenation of x and y
10. $xy = (k+n)(open) + (k+n)(close)$; since there are equal amounts of open and close, we know that xy is balanced

Since both recursive steps result in balanced strings, we can conclude that every string in $\mathcal{P}$ is balanced.

- **Problem 8.13.** Recursively define the binary strings that contain more 0's than 1's. Prove:

(a) Every string in your set has more 0's than 1's.

(b) Every string which has more 0's than 1's is in your set.

- ***Problem 8.14.** A set $\mathcal{A}$ is defined recursively as shown.

  (1) $3 \in \mathcal{A}$

  (2) $x, y \in \mathcal{A} \rightarrow x + y \in \mathcal{A}$
  $x, y \in \mathcal{A} \rightarrow x - y \in \mathcal{A}$

  (a) Prove that every element of $\mathcal{A}$ is a multiple of 3.

  Base case:

  1. 3 = 3*1; 3 is a multiple of 3

  Induction steps: $x, y \in \mathcal{A}$, x, y are multiples of 3

  Case 1: first recursive step

  2. $x, y \in \mathcal{A} \rightarrow x + y \in \mathcal{A}$;
  3. $x = 3k, y = 3n$; since both are multiples of 3 they can be represented like this with k and n being integers
  4. $x + y = 3k + 3n$
  5. $x + y = 3(k + n)$; here we can clearly see that x+y still results in a number that is a multiple of 3

  Case 2: first recursive step

  6. $x, y \in \mathcal{A} \rightarrow x - y \in \mathcal{A}$;
  7. $x = 3k, y = 3n$; same as above
  8. $x - y = 3k - 3n$
  9. $x - y = 3(k - n)$; here we can clearly see that x-y still results in a number that is a multiple of 3

  Since both recursive steps result in a multiple of 3, we can conclude that every number in A must be a multiple of 3.

  (b) Prove that every multiple of 3 is in $\mathcal{A}$.

  Base Case:

  1. 3 = 3*1; 3 is in $\mathcal{A}$.

  Case 1: all multiples of 3 greater than 3 are in $\mathcal{A}$.

  2. $x, y \in \mathcal{A} \rightarrow x + y \in \mathcal{A}$; recursive statement 1, but we can rewrite this to help easily prove the above claim.
  3. $y = 3, x = 3k$; rewrite x as some unknown multiple of 3, and y as the base case 3
  4. $x + y = 3k + 3$; rewrite x+y to this function
  5. $x + y = 3(k + 1)$; this shows that all multiples of 3, greater than 3 are in $\mathcal{A}$.

  Case 2: all multiples of 3 less than 3 are in $\mathcal{A}$.

  6. $x, y \in \mathcal{A} \rightarrow x - y \in \mathcal{A}$; recursive statement 1, but we can rewrite this to help easily prove the above claim.
  7. $y = 3, x = 3k$; rewrite x as some unknown multiple of 3, and y as the base case 3
  8. $x + y = 3k - 3$; rewrite x+y to this function
  9. $x + y = 3(k - 1)$; this shows that all multiples of 3, less than 3 are in $\mathcal{A}$.

  Since we have proven that all multiples of 3 less than, greater than, and equal to 3 are in $\mathcal{A}$, we have proven that all multiples of 3 are in $\mathcal{A}$

- **Problem 8.18.** Recursively define rooted binary trees (RBTs) and rooted full binary trees (RFBTs).

  (a) Give examples, with derivations, of RBTs and RFBTs with 5, 6, and 7 vertices.

  (b) Prove by structural induction that every RFBT has an odd number of vertices.