

Solutions: Mock Exam for Final Discrete Mathematical Structures CS2233

May, 2009

The final exam is cumulative. This mock exam covers material starting with that presented in lecture 4/09/2009. This includes material in sections 4.1, 4.2, 4.3, 4.4, 7.3, 8.1, 8.4, and 8.5. Note that we ran out of time before covering closure operations and graphs in class, and they are not on the exam.

Students should refer to midterms 1 and 2, and to the corresponding mock exams and homeworks, for assistance in studying material previously tested.

Students are strongly encouraged to attempt the problems here without referring to the solutions, which are included in a separate handout. Then compare your solutions to those presented here.

1. Recall that the Fibonacci sequence, $\{f_i\}_i \in \mathbb{N}$, can be recursively defined as follows:

$$\begin{aligned}f_0 &= 0 \\f_1 &= 1 \\f_n &= f_{n-1} + f_{n-2}, \text{ for } n > 1\end{aligned}$$

Use this definition and mathematical induction to show that $f_1 + f_3 + \cdots + f_{n-1} = f_n$ for every positive, even natural number n .

Solution: Every positive, even natural number n can be represented as $n = 2m$ for some $m \in \mathbb{Z}^+$. We use mathematical induction on m to show $\forall m \in \mathbb{Z}^+. P(m)$, in which

$$P(m) \equiv \sum_{i=1}^m f_{2i-1} = f_{2m}$$

Base case: We prove $P(m)$ holds for $m = 1$ as follows:

$$\begin{aligned}\sum_{i=1}^m f_{2i-1} &= \sum_{i=1}^1 f_{2i-1} = f_{2-1} \\&= f_1 = f_1 + 0 = f_1 + f_0 \\&= f_2, \text{ by definition of } f_n \\&= f_{2m}\end{aligned}$$

Step: Given any $m \in \mathbb{Z}^+$, we assume $P(m)$ and show $P(m+1)$, i.e., that $\sum_{i=1}^{m+1} f_{2i-1} = f_{2(m+1)}$:

$$\begin{aligned}\sum_{i=1}^{m+1} f_{2i-1} &= \sum_{i=1}^m f_{2i-1} + f_{2(m+1)-1} \\&= f_{2m} + f_{2m+1}, \text{ by the induction assumption} \\&= f_{2(m+1)}, \text{ by definition of } f_n\end{aligned}$$

2. Recall from lecture that the set of positive propositional formulas Pos be recursively defined as follows:

Basis: $p \in Pos$ for all propositional variables p .

Recursive step: if $\varphi \in Pos$ and $\psi \in Pos$, then $\varphi \wedge \psi \in Pos$, $\varphi \vee \psi \in Pos$, and $(\varphi) \in Pos$.

Given truth assignments σ_1 and σ_2 , we write $\sigma_1 \preceq \sigma_2$ if σ_2 makes true every variable that σ_1 makes true (and possibly some others)

Prove the following by structural induction: Consider any $\varphi \in Pos$ and any truth assignments σ_1 and σ_2 such that $\sigma_1 \preceq \sigma_2$. If σ_1 satisfies φ , then so does σ_2 .

Solution:

Basis: If φ is a propositional variable and σ_1 satisfies φ , then so does σ_2 by definition of $\sigma_1 \preceq \sigma_2$.

Inductive step: Suppose for the induction hypothesis that if σ_1 satisfies φ then σ_2 satisfies φ and that if σ_1 satisfies ψ then σ_2 satisfies ψ . We show that if σ_1 satisfies $\varphi \vee \psi$ then σ_2 satisfies $\varphi \vee \psi$. A complete answer to this problem also requires one to prove that if σ_1 satisfies $\varphi \wedge \psi$ then σ_2 satisfies $\varphi \wedge \psi$ and that if σ_1 satisfies (φ) then σ_2 satisfies (φ) . However, we omit these cases here, as they are similar.

We assume the antecedent and show the consequent. Since σ_1 satisfies $\varphi \vee \psi$, it must be that σ_1 satisfies φ or σ_1 satisfies ψ . In the former case it follows by using the induction assumption that σ_2 satisfies φ , from which it follows that σ_2 satisfies $\varphi \vee \psi$, as desired. We obtain the desired conclusion similarly in the case that σ_1 satisfies ψ .

Discussion: In class, I showed how to solve a problem related to this one by using mathematical induction over a sequence of sets Pos_i for $i \in \mathbb{N}$ and setting $Pos = \bigcup_{i \in \mathbb{N}} Pos_i$. Here, I have used the shorter form, which uses structural induction. Both proofs are equally valid. I presented the longer form in class in an effort to illustrate the point that structural induction can be framed as a form of mathematical induction.

- Write a recursive program that computes n^n for $n \in \mathbb{Z}^+$ in time $\mathcal{O}(\log n)$. You should ignore the fact that this value may not fit in a single machine word.

Solution:

n-to-the-n (n)

```
% Takes  $n \in \mathbb{Z}^+$ 
% Returns  $n^n$ 
return power ( $n, n$ )
```

power (m, n)

```
% Takes  $m, n \in \mathbb{Z}^+$ .
% Returns  $n^m$ .
if ( $m = n$ ) return  $m$ 
 $x :=$  power ( $\lfloor \frac{m}{2} \rfloor, n$ )
if  $\lfloor \frac{m}{2} \rfloor = \frac{m}{2}$ 
    return  $x \cdot x$ 
else
    return  $x \cdot x \cdot n$ 
```

- Write a recurrence relation defining $T(n)$ to be the number of multiplications performed by the algorithm you wrote for the previous problem.

Solution: We let $T(n)$ define the number of multiplications used by `power` (n_1, n_2) when $n = (n_2 - n_1) + 1$.

$$T(1) = 1$$

$$T(n) = T\left(\frac{n}{2}\right) + 1, \text{ when } n \text{ is even}$$

$$T(n) = T\left(\lfloor \frac{n}{2} \rfloor\right) + 2, \text{ when } n \text{ is odd}$$

5. Use strong induction to prove that $T(n) = \mathcal{O}(\log n)$, in which $T(n)$ is defined by your answer to the last problem.

Solution: We show that $T(n) \leq 2 \log n$ for all $n \geq 2$. This proves the result because $2 \log n = \mathcal{O}(\log n)$. (As usual, I am using log base 2.)

Basis: $n = 2$

$$\begin{aligned} T(n) &= T(2) \\ &= T(1) + 1, \text{ by definition of } T(n) \\ &= 2 \\ &= 2 \cdot 1 \\ &= 2 \log 2 \\ &= 2 \log n \end{aligned}$$

Step: There are two cases: n is odd or n is even. When n is even, we have the following:

$$\begin{aligned} T(n) &= T\left(\frac{n}{2}\right) + 1, \text{ by definition of } T(n) \\ &\leq 2 \log \frac{n}{2} + 1, \text{ by the induction assumption} \\ &= 2(\log n - \log 2) + 1, \text{ by property of log applied to fractions} \\ &= 2(\log n - 1) + 1 \\ &= 2 \log n - 2 + 1 \\ &\leq 2 \log n \end{aligned}$$

When n is odd, we have:

$$\begin{aligned} T(n) &= T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 2, \text{ by definition of } T(n) \\ &\leq 2 \log \left\lfloor \frac{n}{2} \right\rfloor + 2, \text{ by the induction assumption} \\ &\leq 2 \log \frac{n}{2} + 2, \text{ by property of } \lfloor \cdot \rfloor \text{ (and the fact that log is monotonic)} \\ &= 2(\log n - \log 2) + 2 \\ &= 2(\log n - 1) + 2 \\ &= 2 \log n - 2 + 2 \\ &= 2 \log n \end{aligned}$$

6. Show that applying Divide and Conquer to calculating the n^{th} Fibonacci number leads to a complexity of $\mathcal{O}(2^n)$.

Solution: Algorithm:

Fibonacci (n)

```
% Uses Divide and Conquer to calculate the nth Fibonacci number
if n = 0 return 0
if n = 1 return 1
return Fibonacci (n-1) + Fibonacci (n-2)
```

Recurrence: $T(n)$ returns the number of additions performed by Fibonacci (n)

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

We use strong induction to show that $T(n) \leq 2^n$ for all $n \in \mathbb{N}$.

Basis: $T(n) = 0 \leq 2^n$ holds when $n = 0$ and when $n = 1$.

Step: $n \geq 2$

$$\begin{aligned}
T(n) &= T(n-1) + T(n-2) + 1, \text{ by definition of } T(n) \\
&\leq 2^{n-1} + 2^{n-2} + 1, \text{ by induction assumption} \\
&\leq 2 \cdot 2^{n-1}, \text{ because } 2^{n-1} \geq 2^{n-2} + 1 \text{ for all } n \geq 2 \\
&= 2^n
\end{aligned}$$

7. Define *relation, reflexive, symmetric, anti-symmetric, transitive, equivalence relation, partition*. Explain why functions are relations.

Solution: Refer to class notes and book.

8. Show that given any set A and any partition of A , being in the same set in the partition defines an equivalence relation over A .

Solution:

Let $\{A_1, \dots, A_n\}$ be any partition of A . Let the relation r be given by $(a_1, a_2) \in r$ if $\exists i \in \{1..n\}. a_1 \in A_i \wedge a_2 \in A_i$. We must show that r is reflexive, symmetric, and transitive.

Reflexive: We must show that for any $a \in A$, $(a, a) \in r$. It suffices to show that $a \in A_i$ for some $i \in \{1..n\}$. This follows from the property of partitions that $A_1 \cup \dots \cup A_n = A$.

Symmetric: We must show that for any $a_1, a_2 \in A$, if $(a_1, a_2) \in r$ then $(a_2, a_1) \in r$. This follows immediately from the definition of r : if both are in the same A_i , then both are in the same A_i .

Transitive: We must show that for any $a_1, a_2, a_3 \in A$, if $(a_1, a_2) \in r$ and $(a_2, a_3) \in r$ then $(a_1, a_3) \in r$. The key point here is that a_2 can be in only one set in the partition because the sets in the partition are mutually disjoint: no two sets in the partition intersect. So if a_1 and a_2 are in the same A_i , and a_2 and a_3 are in the same set A_j , then $A_i = A_j$, so a_1 and a_3 are also in the same set.