

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:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}, \text{ for } n > 1$$

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

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 .

3. 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.
4. Write a recurrence relation defining $T(n)$ to be the number of multiplications performed by the algorithm you wrote for the previous problem.
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.
6. Show that applying Divide and Conquer to calculating the n^{th} Fibonacci number leads to a complexity of $\mathcal{O}(2^n)$.
7. Define *relation*, *reflexive*, *symmetric*, *anti-symmetric*, *transitive*, *equivalence relation*, *partition*. Explain why functions are relations.
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 .