ECS 20: Discrete Mathematics for Computer Science — Sections: 1, 2, 3
UC Davis — Vladimir Filkov — February 8, 2005

Midterm

Instructions: Please answer the questions succinctly and thoughtfully. Good luck.

Name:

ID:

On part	you got	out of
1		12
2		20
3		30
4		14
5		24
Σ		100

1 Definitions [12 points]

Finish the definitions as best as possible.

---

1. (4 pts.) For two compound propositions P and Q, $P \Rightarrow Q$ if

2. (4 pts.) We say $f(x)$ is $O(g(x))$ if

3. (4 pts.) Two integers a and b are mutually prime when

2 True or False [20 points]

Put an **X** through the **correct** box. Scoring: correct choice +4 pts., wrong choice −2 pts., no choice 0 pts.

---

1. The logical operators $\{\wedge, \vee\}$ are logically complete.	$\boxed{\text{True}}$	$\boxed{\text{False}}$
2. If $f(x)$ is $\Theta(x^3)$ then $f(x)$ is $O(x^2)$.	$\boxed{\text{True}}$	$\boxed{\text{False}}$
3. "$\sqrt{2}$ is rational" $\rightarrow$ "$1 > 2$".	$\boxed{\text{True}}$	$\boxed{\text{False}}$
4. There exists a bijection from $\mathcal{N} \times \mathcal{N}$ to $\mathcal{N}$.	$\boxed{\text{True}}$	$\boxed{\text{False}}$
5. $\lceil \lfloor x \rfloor \rceil = \lfloor \lceil x \rceil \rfloor$.	$\boxed{\text{True}}$	$\boxed{\text{False}}$

3 Short Problems [30 points]

1. (6 pts.) $P(P(\phi)) = ?$

2. (12 pts.) Show that $15 | 35^{369} - 35^{123}$

3. (12 pts.) Prove that for all positive integers n

$$\frac{1}{1 \cdot 5} + \frac{1}{5 \cdot 9} + \frac{1}{9 \cdot 13} + \cdots + \frac{1}{(4n-3)(4n+1)} = \frac{n}{4n+1}.$$

4 Logical Operator Completeness [15 points]

A set of operators S is *logically complete* if for any compound proposition P one can find a compound proposition Q written with the same variables but only with the operators from S such that $P \leftrightarrow Q$ is a tautology.

The set of operators $\{\vee, \wedge, \neg\}$ is known to be logically complete. Given that, we showed in class that the set of operators $\{\wedge, \neg\}$ is also logically complete, by showing that whenever "$p \vee q$" is seen in a compound proposition it can be substituted by an expression involving the other two operators.

Show that the set of operators $\{\vee, \neg\}$ is also complete.

5 Algorithms for Primality Testing [24 points]

In this part you will be asked to write a few simple algorithms for testing primality and analyze their running time based on two different assumptions for the running time of individual arithmetic operations. Read the questions carefully and answer meaningfully. For full points you must justify your answers whenever possible.

---

1. (6 pts) Write an algorithm that takes as input a positive integer $n > 2$ and determines if n is prime or not by testing its divisibility with all numbers smaller than n.

If you assume that all operations, including integer addition, subtraction, division, and multiplication (i.e. $a + b$, $a - b$, $a \cdot b$, a **div** b, a **mod** b) take a constant time to execute, what is the **worst-case** running time of this algorithms in terms of n? (Use the big-Oh notation, and to keep it simple count only the loops).

2. (6 pts) Keeping all assumptions the same as above, write an algorithm that tests a number for primality and has asymptotically better **worst-case** running time than the one you wrote in **1.** above. What is its **worst-case** complexity in terms of n? Show explicitly that this algorithm has asymptotically better **worst-case** running time than the first one. (Hint: use a result from divisibility we showed in class.)

3. (12 pts) The assumption that arithmetic operations take constant time is of course not sustainable neither in practice nor in theory (e.g. adding two three-digit numbers is faster than adding two twenty-digit numbers).

A better assumption is that executing the basic arithmetic operations takes time **proportional to the number of digits in the larger of the two numbers involved in the operation**. Let us call this number d. Thus, for example, $t_{div}(a, b) = d \cdot C_{div}$, where t_{div} is the time that the **div** operation takes to execute given input a and b, C_{div} is some constant, and d is the number of digits in the max of a and b.

Redo the time complexity analysis for both algorithms above with this new assumption, while still assuming that the time it takes for all other operations is constant. (Effectively, this amounts to making the size of the input be the number of n's digits instead of n itself.)

a) (3 pts) What are the **worst-case** running times of the algorithms as functions of n? (For simplicity count only the time the arithmetic operations take.)

b) (7 pts) What are their **worst-case** running times as functions of d? Are these tight bounds? (Hint: Express d in terms of n, and use big-Oh)

c) (2 pts) In practical algorithm analysis we use the second assumption about the running time of arithmetic operations instead of the first one. Why do you think that is so?