

**CISS350: Data Structures and Advanced Algorithms**  
**Quiz q10201**

Name: YOUR EMAILScore: 

Note that **answercode** is for writing code/pseudocode/simple answers and does not require mathematical notation. For **answerlong**, you can enter  $\text{\LaTeX}$  code for mathematical notation. Some incomplete/wrong answers are included in the **answerlong** – you will need make modifications.

Here are some pointers on writing math  $\text{\LaTeX}$  code:

1. For “inline math mode”, use  $\$ \dots \$$ . Example:  $\$x = 42 + y\$$  gives you  $x = 42 + y$ . (Mathematical expressions have their own spacing, special symbols, and are in italics.)
2. For “display math mode”, use  $\backslash[ \dots \backslash]$ . Example:  $\backslash[ x = 42 \backslash]$  gives you

$$x = 42$$

(Display math mode is used for emphasis.)

3. Here’s how you do fractions:  $\$ \backslash \text{frac}\{1\}\{2\} \$$  gives you  $\frac{1}{2}$ .
4. Here’s how you do subscript:  $\$ \text{t}_{\{123\}} \$$  gives you  $t_{123}$ .
5. Here’s how you do superscript:  $\$ \text{n}^{\{123\}} \$$  gives you  $n^{123}$ .
6. Here’s how you do log:  $\$ \backslash \text{lg } n \$$  gives you  $\lg n$ .
7. Example:  $\$ T(n) = \backslash \text{frac}\{1\}\{2\} \text{t}_{\{42\}} \text{n}^3 \backslash \text{lg } n = A \text{n}^3 \backslash \text{lg } n = O(\text{n}^3 \backslash \text{lg } n) \$$  gives you  $T(n) = \frac{1}{2}t_{42}n^3 \lg n = An^3 \lg n = O(n^3 \lg n)$ .

The above information should be enough for this quiz. For more information on  $\text{\LaTeX}$  you can go to [my website](#), scroll down to the Tutorials section and click on latex.pdf.

(TURN PAGE)

# RUNTIME FOR ARRAY ASSIGNMENT

The goal is to compute the big-O runtime of the basic array assignment algorithm (i.e., copy values from one array to another).

For the next few questions, the goal is to analyze the runtime of this algorithm:

```
for i = 0, ..., n - 1:
    x[i] = y[i]
```

Q1. Rewrite the above algorithm with goto statements, assign constant times to each statement and compute the time taken as a function of  $n$ ,  $t_1$ ,  $t_2$ ,  $t_3$ , ... where  $t_1, t_2, t_3, \dots$  are runtimes for each statement in your algorithm. Make sure you number these times starting with  $t_1$ , then  $t_2$ , etc. Include the number of times each statement is executed.

ANSWER:

	i = 0	t1	1
LOOP:	if i >= n:	t2	?
	goto ENLOOP	t3	?

Q2. Let  $T(n)$  be the runtime function of the above algorithm. Write  $T(n)$  as a polynomial in descending power of  $n$ , i.e., the *highest degree term appearing first*.

ANSWER:

$$T(n) = t_{200}n^{100} + t_{1000}n^{20}$$

(Correct the above.)

Q3. Simplify the runtime function  $T(n)$  by giving names  $A$ ,  $B$ ,  $C$ ,  $D$ , ... to the constants of the function from (a). (The  $t_1, t_2, \dots$  should disappear.)

ANSWER:

$$T(n) = An^{100} + Bn^{99}$$

(Correct the above.)

Q4. Fudge away the constants and write down the simplest  $g(n)$  such that the runtime is a big-O of your  $g(n)$ . Your  $g(n)$  should be either  $n$  or  $n^2$  or  $n^3$  or  $n^{102.25}$  ...

ANSWER:

$$T(n) = O(n^{3.1415} \lg n + 5000)$$

(Correct the above.)

□

## INSTRUCTIONS

In the file `thispreamble.tex` look for

```
\renewcommand\AUTHOR{}
```

and enter your email address:

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

(This is not really necessary since alex will change that for you when you execute `make`.) In your bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

Enter your answers in `main.tex`. In the bash shell, execute “`make`” to recompile `main.pdf`. Execute “`make v`” to view `main.pdf`.

For each question, you’ll see boxes for you to fill. For small boxes, if you see

```
1 + 1 = \answerbox{}
```

you do this:

```
1 + 1 = \answerbox{2}
```

`answerbox` will also appear in “true/false” and “multiple-choice” questions.

For longer answers that need typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.  
\begin{answercode}  
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.  
\begin{answercode}  
int x;  
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?  
\begin{answerlong}  
\end{answerlong}
```

you can write

```
What is the color of the sky?  
\begin{answerlong}  
The color of the sky is blue.  
\end{answerlong}
```

A question that begins with “T or F or M” requires you to identify whether it is true or false, or meaningless. “Meaningless” means something’s wrong with the question and it is not well-defined. Something like “ $1 + 2 = 4$ ” is either true or false (of course it’s false). Something like “ $1+2 = 4?$ ” does not make sense.

When writing results of computations, make sure it’s simplified. For instance write 2 instead of  $1 + 1$ .

#### HIGHER LEVEL CLASSES.

For students beyond 245: You can put L<sup>A</sup>T<sub>E</sub>X commands in `answerlong`.

More examples of meaningless statements: Questions such as “Is  $42 = 1+2$  true or false?” or “Is  $42 = \{2\}^{\{3\}}$  true or false?” does not make sense. “Is  $P(42) = \{42\}$  true or false?” is meaningless because  $P(X)$  is only defined if  $X$  is a set. For “Is  $1 + 2 + 3$  true or false?”, “ $1 + 2 + 3$ ” is well-defined but as a “numerical expression”, not as a “proposition”, i.e., it cannot be true or false. Therefore “Is  $1 + 2 + 3$  true or false?” is also not a well-defined question.

More examples of simplification: When you write down sets, if the answer is  $\{1\}$ , do not write  $\{1, 1\}$ . And when the values can be ordered, write the elements of the set in ascending order. When writing polynomials, begin with the highest degree term.

When writing a counterexample, always write the simplest.