CISS350: Data Structures and Advanced Algorithms
Quiz q10205

Name: YOUR EMAIL Score:

Note that `answercode` is for writing code/pseudocode/simple answers and does not require mathematical notation. For `answerlong`, you can enter 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 LaTeX code:

1. For "inline math mode", use `$...$`. 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 `\[...\]`. Example: `\[ x = 42 \]` gives you
$$x = 42$$
(Display math mode is used for emphasis.)
3. Here's how you do fractions: `$\frac{1}{2}$` gives you $\frac{1}{2}$.
4. Here's how you do subscript: `$t_{123}$` gives you t_{123}.
5. Here's how you do superscript: `$n^{123}$` gives you n^{123}.
6. Here's how you do log: `$\lg n$` gives you $\lg n$.
7. Example: `$T(n) = \frac{1}{2} t_{42} n^3 \lg n = An^3 \lg n = O(n^3 \lg n)$` 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 LaTeX you can go to my website, scroll down to the Tutorials section and click on latex.pdf.

For the following algorithms, state the big-O of the runtime complexity. For each problem the size of the problem is n. You should state your runtimes as big-O of n^k or $\lg n$ or $n^k \lg n$ where k is as small as possible (You do not need to do the math. Just state the big-O.) Unless otherwise stated, recall that "runtime" means "worst runtime". Also, recall that for recursion runtimes:

1. If the recursive runtime has the form
$$T(n) = T(n/2) + A$$

where A is a constant, then

$$T(n) = O(\lg n)$$

2. And if

$$T(n) = 2T(n/2) + An + B$$

where A, B are constants, then

$$T(n) = O(n \lg n)$$

(Turn Page)

Q1.

```
a = x[0]
for i = 0, 1, 2, ..., n - 1:
    x[i] = a + x[i]
    a = a + 1
```

Answer:

$T(n) = O(??)$

Q2.

```
for i = n, n - 1, ..., 1:
    a = rand() % n
    b = rand() % n
    if x[a] > x[b]:
        t = x[a]
        x[a] = x[b]
        x[b] = t
```

Answer:

$T(n) = O(??)$

Q3.

```
p = 1
for i = 1, 2, ..., (n - 1)/2:
    p = p * 2
```

Answer:

$T(n) = O(??)$

Q4.

```
a = x[0]
for i = 0, 1, 2, ..., n - 1:
    x[i] = a + x[i]
    for j = 0, 1, 2, ..., i:
        x[0] = x[0] + x[i]
    a = a + 1
```

Answer:

$T(n) = O(??)$

Q5. State the best runtime:

```
for i = 0, 1, 2, ..., n - 1:
    if x[i] > 0:
        for j = 0, 1, 2, ..., i:
            x[j] = 2 * x[j]
```

Answer:

$T(n) = O(??)$

Q6. State the worst runtime:

```
p = 1
for i = 1, 2, 3, ..., n - 1:
    if x[i] == 0:
        for j = 0, 1, 2, ..., i - 1:
            x[i] = 1
    x[i] = x[i] + 2
```

Answer:

$T(n) = O(??)$

Q7.

```
int f(int n)
{
    if n <= 3:
        return 42
    else:
        return f(n/2) + 43
}
```

Answer:

$T(n) = O(??)$

Q8. The size of the problem is `end - start`.

```
int f(int start, int end)
{
    n = end - start
    if n <= 1:
        return 0
    else:
        mid = n / 2
        if mid is even:
            return f(start, mid) + 42
        else:
            return f(mid, end) + 43
}
```

Answer:

$T(n) = O(??)$

Q9. The size of the problem is `end - start`.

```
int f(int start, int end)
{
    n = end - start
    if n <= 1:
        return start % 2
    else:
        mid = n / 2
        left = f(start, mid)
        right = f(mid, end)
        s = 0
        for i = 1, ..., n:
            s += mid
        return left + right + s
}
```

Answer:

$T(n) = O(??)$

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 LaTeX 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.