CISS350: Data Structures and Advanced Algorithms
Quiz q10902

Name: YOUR EMAIL Score:

Here's how you describe (in text) for a binary tree with a heap tree structure. For this tree

you write

```
{10, 6, 7, 5, 2, 0, 3, 1}
```

i.e., list the values using breadth-first traversal (like what we did in class).

Q1. Execute `build_heap` on the array $\{7, 5, 3, 1, 2, 4, 6, 8\}$ and describe the resulting maxheap as an array. (Note: `build_heap` builds the maxheap *bottom-up*.)
Answer:

```
{?}
```

Q2. The following is already a maxheap:

During heapsort, what is the resulting array after *four* extract-root operations. Recall that for heapsort the roots extracted are *not* thrown away!
Answer:

```
{?}
```

Q3. Suppose your `std::vector< int >` is sorted (in ascending order). What is the runtime of the *fastest* algorithm

```
void ascending_to_maxheap(std::vector< int > & v);
```

that converts it a maxheap?
Answer:

$T(n) = O(?)$

(You should only have $O(1)$ or $O(\lg n)$ or $O(n)$ or $O(n \lg n)$ or $O(n^2)$ or ..., i.e., it should be of the form $O(n^a \lg^b n)$.)

Q4. Write a function that checks if a `std::vector< int >` object is a maxheap, returning the *first* index (scanning *from index 0* to the end of the array) where the maxheap property is violated. If the `std::vector< int >` object is a maxheap, `-1` is returned.

```
int maxheap_violation_index(const std::vector< int > & v);
```

Answer:

```
int LEFT(int i)
{
}

int RIGHT(int i)
{
}

int maxheap_violation_index(const std::vector< int > & v)
{
}
```

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.