Virginia Tech
1872

READ THIS NOW!

- Print your name in the space provided below.
- Unless a question involves determining whether given Java code is syntactically correct, assume that it is, and that any necessary `imports` have been made.
- There are 7 short-answer questions, priced as marked. The maximum score is 100.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.
- Aside from the allowed one-page fact sheet, this is a closed-book, closed-notes examination.
- No laptops, calculators, cell phones or other electronic devices may be used during this examination.
- You may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.

Name (Last, First) ____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

____
signed

1. Assume that $f(n)$, $g(n)$ and $h(n)$ are positive-valued functions defined on the set of non-negative integers.

a) [6 points] What does the fact given below imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

$$\forall n \geq 10,\ g(n) \geq 7h(n)$$

From the definition of Ω, this directly says that g is $\Omega(\mathrm{h})$**.**

Furthermore, since we can divide through by 7, we also know that $\forall n \geq 10,\ \frac{1}{7}g(n) \geq h(n)$**, which directly says that** h is $\mathrm{O(g)}$

b) [6 points] What does the fact given below imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

$$\forall n \geq 42,\ \frac{1}{2}h(n) \leq f(n) \leq 5h(n)$$

By definition, this says that f is $\Omega(\mathrm{h})$ **and that** f is $\mathrm{O(h)}$**. Therefore, by definition,** f is $\Theta(\mathrm{h})$**.**

Furthermore, since Θis an equivalence relation (symmetric) this also says that h is $\Theta(\mathrm{f})$**.**

---

2. [15 points] For each part, state the simplest function $g(n)$ such that the given function is $\Theta(g)$; you should use theorems from the course notes; no justification is required

a) $a(n) = 14n + 3n\log n$ $n\log n$

b) $b(n) = 3n^2\log n + 5n^2$ $n^2\log n$ **(clearly, this is more than a constant factor larger than** n^2 **)**

c) $c(n) = 10n^2 + 5^n$ 5^n

d) $d(n) = 4n\left(n + \log n + 1\right)$ n^2 **(multiply it out)**

e) $e(n) = 5n^3 + 7n^2 + 2n + 42$ n^3

3. [15 points] The following algorithm, known as Horner's Rule, is used to evaluate a polynomial $f(x) = \sum_{i=o}^{N} a_i x^i$.

Assume that the coefficients of the polynomial are stored in an array `a[]` and the value for x is stored in a variable `x`.

```
Poly = 0;                    // 1, done once
for (i=N; i>=0; i--)         // 1 before loop, 2 each pass, 1 to exit
   Poly = x * Poly + a[i];   // 4 each pass
```

Derive a simplified complexity function T(N) for the given algorithm. Show all supporting work.

$$\begin{aligned} T(N) &= 1 + 1 + \sum_{i=0}^{N} (2+4) + 1 \\ &= \sum_{i=1}^{N+1} 6 + 3 \\ &= 6(N+1) + 3 \\ &= 6N + 9 \end{aligned}$$

4. [14 points] If a buffer pool implementation uses an array to physically organize the individual buffers, would that have any implications when choosing a replacement policy? Explain carefully.

If LRU were used, you would be faced with two choices:

- **after every access that results in a hit, move that element to the front; after every access that results in a miss, place the new element at the front; either way, you'll have to shift lots of array elements each time**
- **leave the elements unordered; then you have to search the entire array every time, in order to find the LRU element, you need to do a replacement**

If LFU were used, the same issue arise; the only difference is that on a hit you may only have to swap that element with a few of its predecessors in the list; but it still requires expensive list upadating.

If FIFO were used, then you simply treat the array as a circular queue; no reordering takes place at any time. For replacement, you just update the tail index. For adding a new element, you just insert it at the front index and update that index.

So, if you're going to use an array, FIFO replacement is the only option that does not entail significant extra costs. (And, therefore, you shouldn't choose an array as your physical structure.)

5. [14 points] Consider the GIS project that you are currently working on for this course. The project specification does not require including a buffer pool. Given the nature of the project, could adding a buffer pool to the system improve performance? Why or why not? If yes, what objects would you store in the buffer pool?

The search commands require retrieving GIS records from a disk file; disk accesses are extremely slow.

If there is any likelihood of temporal locality in the search patterns, then employing a buffer pool to store recently-retrieved GIS records would improve overall performance by reducing the number of times a disk access was required. If there's no significant temporal locality, we'll waste some memory and a tiny amount of time (relative to the cost of the disk accesses).

It makes more sense for the buffer pool to store interpreted GIS records than to store raw strings, since that would entail transforming the raw string into a GIS record only once, no matter how many times it was retrieved from the buffer pool.

6. [14 points] Again, consider the GIS project. The specification requires that the PR quadtree leaf nodes be bucketed — that is, that they be capable of storing several data elements instead of just one. What effect does this requirement have on the time efficiency of operations in the PR quadtree? Why? Does this requirement have any other important effects? If so, what and why?

A PR quadtree leaf does not split until it is full and we try to add another element that falls into the corresponding region. So, it's clear that a bucketed leaf will not split as soon as an unbucketed leaf.

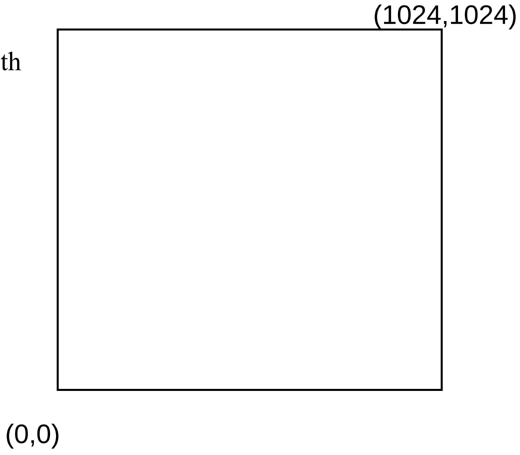
The most important effect is that some branches in the resulting tree will be shorter than if buckets were not used.

That means that searches down those branches will not require as many steps to reach a leaf node (all searches terminate at a leaf); searching the bucket in a leaf is unlikely to be more expensive than it would be to continue traversing the longer branch we would have if buckets were not used. (Remember that we must recomputed region boundaries as we descend from level to level.)

The fact that search is cheaper (down the affected branches) means that insertions and deletions will also be cheaper in those branches. The cost of splitting a bucketed leaf will be somewhat higher since more data values must be pushed down through the new internal node; but, that is more than balanced by the reduced number of times that splits will be necessary at all.

Finally, the memory cost of the tree will be reduced due to the elimination of internal nodes that will not need to be created since fewer splits will occur. Since the size of a leaf node is essentially the same as the size of the element stored in the bucket, the only effect on space cost there is that some leaves will have "wasted" space in unused bucket cells, but in Java a bucket cell is only as large as a pointer, and that "wasted" space is precisely the cost of preventing future splits.

7. Consider using a PR quadtree, with bucket size 1, to organize data points that lie in the region shown below.



a) [8 points] Is it possible that inserting two data points will cause multiple splittings to occur? If yes, explain how and illustrate with a specific example. If no, explain why not.

If a new data point is inserted and it falls into an empty quadrant, then no splitting occurs. If it falls into a quadrant that already contains a data point, then that quadrant must split at least once.

If a single split still leaves the two data points in the same sub-quadrant, then additional splits must be performed.

Consider A at (10, 10) and B at (20, 20)…

b) [8 points] Is it necessarily the case that inserting two data points that are a distance of 10 apart will cause multiple splittings to occur? If no, explain why not and illustrate with a specific example. If yes, explain why.

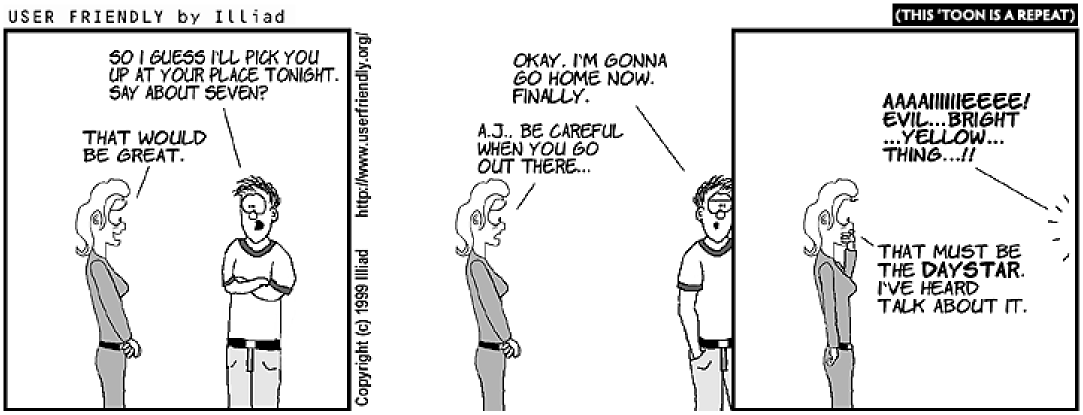
No. The issue is where the two points lie in relation to the "natural" region boundaries defined by the PR quadtree.

When a PR quadtree is created, the specific world boundaries completely determine exactly where future region boundaries can be created.

If two data points lie within a small distance of each other, but a "natural" boundary lies between them, and the PR quadtree has already created that boundary (due to earlier insertions) then no further splitting will be required to separate those two points (although they could trigger splitting due to other data points).

If the "natural" boundary between them does not exist yet, then the PR quadtree only needs to split far enough to create that boundary, and that may require only a single splitting operation.

For example, consider inserting A at (511, 511) and B at (513, 513) into the empty region given above; only a single splitting is needed to separate them.


USER FRIENDLY by Illiad
(THIS 'TOON IS A REPEAT)
SO I GUESS I'LL PICK YOU UP AT YOUR PLACE TONIGHT. SAY ABOUT SEVEN?
THAT WOULD BE GREAT.
http://www.userfriendly.org/
Copyright (c) 1999 Illiad
OKAY. I'M GONNA GO HOME NOW. FINALLY.
A.J., BE CAREFUL WHEN YOU GO OUT THERE...
AAAAIIIIIEEEEE! EVIL...BRIGHT ...YELLOW... THING...!!
THAT MUST BE THE DAYSTAR. I'VE HEARD TALK ABOUT IT.