
***READ THIS NOW!***

- Print your name in the space provided below.
- Unless a question involves determining whether given C++ code is syntactically correct, assume that it is. Unless a question specifically deals with compiler #include directives, you should assume the necessary header files have been included.
- There are 8 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 such that the following are true:

$$\forall n \geq 7, f(n) \geq 3g(n)$$

$$\forall 0 \leq n \leq 10000, h(n) \leq f(n) \leq g(n)$$

a) [10 points] What does the first fact given above imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

**By definition, f(n) is Ω(g(n)). If we divide through the inequality by 3, the resulting inequality implies that g(n) is O(f(n)).**

b) [10 points] What does the second fact given above imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

**Since the second fact only implies that certain bounds apply for a finite range of values of n, no conclusions about O, Ω, or Θ follow from it.**

---

2. [10 points] A BST implementation may use a hierarchy of node types to save memory. Write C++ declarations for the necessary node types, making good use of inheritance. Do not show member function implementations.

**Here is a minimal implementation:**

```
template <typename T> class nodeWOPtrs {     // leaf node type
public:
   T   Elem;
   virtual ~nodeWOPtrs() {}
};

class nodeWithPtrs : public nodeWO Ptrs {    // internal node type
public:
   nodeWOPtrs<T>* Left;
   nodeWOPtrs<T>* Right;

   nodeWithPtrs();                           // needs default constructor to
                                             //   guarantee pointers are set
};
```

3. Consider the following algorithm for computing the sum of two matrices:

```
for ( i = 1; i <= N; i = 2*i ) {          // 1 before, 3 each pass, 1 on exit
   if ( (someConst + N) % anotherConst ) // 3 per pass
      accum = accum + N;                 // 2 per pass, if done
}
```

a) [15 points] Using the rules given in class for counting operations, find a simplified function T(N) that counts the number of operations the algorithm would perform.

**As was the case for some examples in the notes, the counter for the loop will go from to 1 to log N.**

$$T(N) = 1 + \sum_{i=1}^{\log N} (3+3+2) + 1$$
$$= 2 + 8 \log N$$

b) [5 points] What big-Θ complexity class does your answer to the previous part belong to? (No proof is necessary.)

**Trivially, it's Θ(log N).**

4. [8 points] In a B-tree of order 101, assume that the deletion of a particular value from a leaf node causes that leaf node to underflow, and that the underflow can be alleviated by "borrowing". Is it necessary, or possible, or impossible that this "borrowing" will cause the parent of the leaf to underflow? Explain clearly why or why not.

   **It is impossible that this situation will cause the parent of the leaf to underflow. If the leaf underflows, and "borrowing" is possible, then the relevant sibling must contain at 51 values (so it won't underflow after losing one). A value is moved from that sibling to the parent, where it replaces the value in the parent that falls "between" the two leaf nodes, and that value is moved into the leaf that underflowed.**

   **This does not change the number of values stored in the parent, so there is no possibility the parent can underflow.**

5. [7 points] A B+ tree differs from a B-tree in two respects, one being that each leaf node has pointer(s) to the leaves to its immediate left and right. What is the specific advantage in adding those pointers to the leaves in a B+ tree?

   **By linking the leaves into a linear list, range searches will be more efficient since it will now be necessary to just search for the smallest value that's larger than or equal to the first value in the range and then perform a linear walk across the leaf level until the first value that's too big is located.**

   **In a B-tree, this would require returning to levels above the leaves in order to continue the search.**

6. [9 points] Briefly explain whether using a buffer pool be expected to improve performance in the following situations:

   a) application that uses a vector to store a collection of records that are searched repeatedly

   **There would be no significant advantage; the buffer pool can only significantly decrease the cost of retrieving records from disk. There might be a slight gain however; if the searches exhibit some temporal locality, using a cache to store recently-accessed records could make repeated searches for those records cheaper.**

   b) application that reads sequentially through a data file to build a vector of records

   **There would be no advantage; no record would be accessed more than once. Since a sequential pass through the file would exhibit spatial locality (in a very simple way), there would be some merit in retrieving a block of records at once from disk. However, this doesn't need, or benefit from the use of a buffer pool.**

   c) application that uses a file to store a collection of records that are searched repeatedly

   **In this case, there is the potential that the buffer pool will lead to improved performance, depending on whether the record searches exhibit enough locality.**

---

7. A buffer pool implementation uses a fixed number of slots, organized by some simple linear data structure.

   a) [8 points] Describe a specific physical data structure that would be a good choice if the replacement policy is FIFO.

   **The records are removed in the order they are inserted, so this is just a queue. Either an array or a singly-linked list would be appropriate. Since the number of slots is fixed, the linked list would use slightly more memory, but that may be offset by the fact that the array would require extra arithmetic for the circular indexing that would be needed.**

   b) [8 points] Repeat part a if the replacement policy is least frequently used (LFU).

   **The record to be replaced is the one with the smallest number of hits. If the records were stored in a poor order, this could require a linear search. So, it would be better to store the records in order of their frequency counts. Since that would require swapping records when frequency counts change, it would be best to do this in a way that didn't require too many copy operations. Therefore, a linked list would be a better choice.**

8. [10 points] In a PR quadtree with a bucket size of 1, is the height of the tree determined (or bounded) by a function that depends only on the number of data points stored in it? If yes, conjecture what the bound would be and give an intuitive justification. If not, explain what factors would be involved in determining the height of the tree.

**The height of a PR quadtree doesn't really depend on the number of elements, at least not if the number is greater than two. Adding a second record that is very close to a record that's already in the tree will cause a branch of the tree to extend by many levels. So, the height of the tree depends on the distance between the two closest points.**

**A more subtle point is that the number of levels also depends on the size of the "world" since that determines how many times it must be subdivided to separate the two closest points.**

**So, the height of a simple PR quadtree is bounded by some formula that depends on the minimum distance between data points and the "diameter" of the world.**

**The interested reader is encouraged to consult the survey paper by Samet on the website.**