Final Exam Review

Final Exam Information

- Date: 5/2nd Tuesday
- Time: 6:30 – 9:00 PM (Online)
- Exam Time Required: 2.5 hours
  - The exam will be ready in Canvas @ 6:30 PM and close @ 9:10 PM.
  - Students must submit by 9 PM.
  - Late submission penalty (5 pts for every 5 minutes – a maximum of 10 points lost).
  - Any exams not submitted to Canvas or submitted after 9:10 PM will not be graded.

Final Exam Information

Format and Rules:

1. Rules
   - It must be an individual work. Any evidence of copy or cheating is subject to be reported.
   - Closed notes and books.
   - If asked, all work must be shown.

2. Question Types:
   - Multiple choice, T/F, and multi-selection
   - Short conceptual open-ended questions.
   - Theoretical application open-ended questions.

Final Exam Information

Topics:

- Algorithm Analysis
  - Function growth
  - Time complexities (recursion case)
  - Loop Invariants

- Sorting Algorithms
  - Insertion, Merge, Heap, Quick, Counting, Radix, and Bucket

- Trees
  - Binary Search Tree, Red-Black Tree

Final Exam Information

Topics:

- Data Structure
  - Dynamic Programming, Greedy Algorithm

- Graph Algorithms
  - Elementary Graph Algorithms: Breath First Search, Depth First Search, Topological sort, Strongly Connected Component
  - Minimum Spanning Tree
  - Shortest Paths

Final Exam Practice Questions

Conceptual Question:

Suppose an array A has elements that are short integers, as shown [1,1,2,3,2,1,4,2,1,5,2,1]. While the user wants to sort the elements, the user also wants to print out the monotonically increasing index order. Demonstrate the algorithm that you would suggest and explain.

Final Exam Practice Questions

Conceptual Question:
Suppose an array A has elements that are short integers, as shown [1,1,2,3,2,1,4,2,1,5,2,1]. While the user wants to sort the elements, the user also wants to print out the monotonically increasing index order. Demonstrate the algorithm that you would suggest and explain.

1. The stable sorting algorithm, e.g., insertion, merge, or counting.
2. Insertion sort may not be the best considering the worst case.
3. Counting sort vs. merge sort.
    1. Counting sort wins: running time
    2. Easy to track the index of A.

Final Exam Practice Questions

Conceptual Question:

If $T(n) = 2T\left(\frac{n}{2}\right) + n^3$, show that $T(n) = \Theta(n^3)$ using the Master’s theorem. Justify your answer.

Final Exam Practice Questions

Conceptual Question

If $T(n) = 2T\left(\frac{n}{2}\right) + n^3$, show that $T(n) = \Theta(n^3)$ using the Master's theorem.

Master's Theorem :

$$n^{1+\epsilon} =?\, n^3$$
$$2\left(\frac{n}{2}\right)^3 <?\, cn^3$$

Justification:

- Constant conditions are justified.
- Suppose T(n) characterizes a divide-conquer algorithm. The recursion requires half split to have two subproblems. The dividing and combining times dominate the running time.

Final Exam Practice Questions

Application Question

Suppose you have an array with random integers. You are asked to implement an algorithm that finds the maximum sum of four sequences. How would you modify the following divide-conquer algorithm? What would be the running time?

```
Find-Max-Crossing-Subarray(A,low,mid,high)
$leftsum = -\infty$
$sum = 0$
For $i = mid$ downto $low$
    $sum = sum + A[i]$
    If $sum > leftsum$
        $leftsum = sum$
        $maxleft = i$
$rightsum = -\infty$
$sum = 0$
For $j = mid + 1$ to $high$
    $sum = sum + A[j]$
    If $sum > rightsum$
        $rightsum = sum$
        $maxright = j$
return $maxleft, maxright, leftsum + rightsum$
```

```
if $high == low$
    return $(low, high, A[low])$   //base case: only one element
else $mid = \lfloor (low + high)/2 \rfloor$
    $(leftlow, lefthigh, leftsum)$ =FIND-MAXIMUM-SUBARRAY(A,low,mid)
    $(rightlow, righthigh, rightsum)$ =FIND-MAXIMUM-SUBARRAY(A,mid+1,high)
    $(crosslow, crosshigh, crosssum)$ =FIND-MAX-CROSSING-SUBARRAY(A,low,mid,high)
    if $leftsum \geq rightsum$ & $leftsum \geq crosssum$
        return$(leftlow, lefthgh, leftsum)$
    elseif $rightsum \geq leftsum$ & $rightsum \geq crosssum$
        return$(rightlow, righthigh, rightsum)$
    else return$(crosslow, crosshigh, crosssum)$
```