csce350 — *Data Structures and Algorithms*
Fall 2020 — Test B

Evan

first/given name

5:10

time downloaded

Owre

last/family name

Problem	Value	Your Score
1	32	
2	8	
3	10	
4	8	
5	6	
6	10	
7	8	
8	8	
9	10	
Total	100	

- ✓ Read the questions carefully and make sure to give the answers asked for. Pay particular attention to **boldface** words. Don't give a beautiful answer to the wrong question.
- ✓ You have 120 minutes to complete this test.
- ✓ A single 8.5″ × 11″ sheet of notes in your own handwriting is required.
- ✓ No calculators nor other reference material are allowed.
- ✓ Show enough work to convince your instructor that you know what you're doing. Mark your answers clearly.
- ✓ Make sure you have all 15 pages, including the three reference sheets at the end.
- ✓ Partial credit will be awarded for incorrect answers that demonstrate partial understanding of the relevant concepts. Therefore, it is to your advantage to explain your reasoning and show your work. However, meaningless or irrelevant writing will not earn partial credit.
- ✓ Be sure to review the submission checklist at the end before uploading.

I understand that it is the responsibility of every member of the Carolina community to uphold and maintain the University of South Carolina's Honor Code. I certify that I have neither given nor received unauthorized aid on this exam.

signature

Problem 1 *(32 points)*
Multiple choice. **Choose** the best answer for each question. (2 points each)

A B C D

1. When does a 2-3 tree increase in height?

   A. every time a leaf node overflows and must be split
   B. every time an element is inserted
   C. every time the root node overflows and must be split
   D. every time an insert results in a balance factor of +2 or -2

A B C D

2. Which sorting algorithm's behavior can be described as recursively sorting the first $n-1$ elements, then locating the correct "home" for element n?

   A. quicksort
   B. selection sort
   C. insertion sort
   D. mergesort

A B C D

3. Which of the following data structures is the best choice for implementing a priority queue?

   A. heap
   B. 2-3 tree
   C. AVL tree
   D. linked list

A B C D

4. Which algorithm design strategy best describes the algorithm below?

```
ARRAYHASDUPLICATEELEMENTS(A[0, ..., n-1])
  MERGESORT(A)
  for i ← 1, ..., n-1 do
    if A[i] = A[i-1] then
      return true
    end if
  end for
  return false
```

   A. decrease and conquer
   B. divide and conquer
   C. brute force
   D. transform and conquer

A B C D

5. The equation

$$a^n = \begin{cases} \left(a^{\frac{n}{2}}\right)^2 & \text{if } n \text{ is even} \\ \left(a^{\lfloor \frac{n}{2} \rfloor}\right)^2 \cdot a & \text{if } n \text{ is odd} \end{cases}$$

is the basis for a _______ algorithm for computing integer powers.

A. decrease and conquer
B. divide and conquer
C. brute force
D. transform and conquer

A B C D

6. Binary search is applicable only when _______.

A. the search key is not in the array
B. the array is already sorted
C. the search key is in the array
D. the array is not already sorted

A B C D

7. The worst-case run time of mergesort is _______.

A. $\Theta(n \log n)$
B. $\Theta(n)$
C. $\Theta(n^2)$
D. $\Theta(\log n)$

A B C D

8. The worst-case run time of insertion sort is _______.

A. $\Theta(n \log n)$
B. $\Theta(\log n)$
C. $\Theta(n^2)$
D. $\Theta(n)$

A B C D

9. The typical application of the divide-and-conquer approach is to divide a problem into several smaller subproblems, solve those subproblems recursively, then _______.

A. discard all but the first of those subproblem solutions
B. discard all but the last of those subproblem solutions
C. randomly select one of the subproblem solutions
D. combine the solutions to those subproblems to form a solution to the original problem

A B C D 10. The basic idea of mergesort is to ______.

A. merge each element of the array with the element to its left
B. partition the array around a pivot element, then recursively sort the left and right sides
C. merge each element of the array with the element to its right
D. sort the first and second halves of the array separately, then merge the results

A B C D 11. What is the primary difference between decrease-and-conquer and divide-and-conquer?

A. Decrease-and-conquer algorithms are generally inefficient, divide-and-conquer algorithms are generally very efficient.
B. Decrease-and-conquer algorithms generally solve only one smaller subproblem recursively, but divide-and-conquer algorithms generally solve two or more subproblems recursively.
C. Decrease-and-conquer algorithms are generally recursive, but divide-and-conquer algorithms are generally iterative.
D. There is no meaningful difference. These are two different names for exactly the same algorithm design strategy.

A B C D 12. The basic idea of quicksort is ______.

A. partition the array around a pivot element, then recursively sort the left and right sides
B. merge each element of the array with the element to its left
C. sort the first and second halves of the array separately, then merge the results
D. merge sorted elements into the array one-by-one

A B C D 13. Which algorithm uses the same partitioning idea that quicksort uses?

A. insertion sort
B. quickselect
C. binary search
D. travelling salesperson

A B C D 14. When a node in a 2-3 overflows, the correct response is to split that node and to promote the ______ key to the next level up.

A. smallest
B. median
C. largest
D. newest

A B C D 15. The worst-case performance for quicksort occurs when ______.

A. the recursion reaches a base case immediately
B. the pivot element is the median element in the array
C. the pivot is the largest or the smallest element in the array
D. the partition algorithm fails to select any pivot at all

A B C D 16. The worst-case run time of quicksort is ______.

A. $\Theta(n \log n)$
B. $\Theta(\log n)$
C. $\Theta(n^2)$
D. $\Theta(n)$

Problem 2 *(8 points)*
The array below has just been partitioned.

3	18	9	15	6	21	12	24

Is it possible to determine, simply by inspecting this particular array, which element was the pivot? **Circle** one:

Yes No

If you answered "Yes," **circle** the pivot element and **explain**, in one sentence, how you know.

If you answered "No," **circle** all of elements that might have been the pivot and **explain**, in one sentence, how you know. both 3 & 24 are in their final positions, & no other values are both larger than everything to the left & smaller than everything to the right

Problem 3 *(10 points)*

Two numbers a and b are called a *septemic pair* if $a - b = 7$ or $b - a = 7$. Suppose you want determine whether a given array contains any septemic pairs.

- **Input**: An array of $A[0, \ldots, n-1]$ of n integers.
- **Output**: **True** if A contains a septemic pair, or **False** otherwise.

Write pseudocode for a $\Theta(n \log n)$ time algorithm for this problem. (You may assume that all of the algorithms we've covered in this class are available as subroutines; you can call them directly without writing down pseudocode for them.)

```
Septemicpair(A[0,...,n-1])
  QuickSort(A[0,...,n-1]) <---- Θ(n log n)
  for(i ← 0,...,n-2
      ~~if A[i]-A[i+1]=7 or A[~~
      if A[i+1]-A[i]=7
         return true
     end if
  end for                        } Θ(n) < Θ(n log n)
  return false
```

Problem 4 *(8 points)*

List the recursive calls made by the decrease-and-conquer version of INTEGERPOWER(23, 21).

a n

If n is even t ← intRgerPower(2

n is odd IntegerPower(23, ⌊21/2⌋) -> IntegerPower(23, 10)

n is even IntegerPower(23, 10/2) -> IntegerPower(23, 5)

n is odd IntegerPower(23, ⌊5/2⌋) -> IntegerPower(23, 2)

n is even IntegerPower(23, 2/2) -> IntegerPower(23, 1)

n is 1 ← base case

Problem 5 *(6 points)*

Jenny uses the Karatsuba algorithm to compute this product:

$$\underset{a}{5532} \times \underset{b}{5737}$$

$n = 4 \therefore m = 4/2 = 2$

$a = 10^2 a_1 + a_0$

$b = 10^2 b_1 + b_0$

She calls KARATSUBA(5532, 5737), being careful to represent the inputs as arrays of digits. She knows, of course, that this algorithm makes *three recursive calls* at the top level, followed by some additional computation to combine those results.

Find the parameters passed to the three top-level recursive calls made when Jenny uses this algorithm. (You only need to fill in the 6 blanks below; you do not need to consider any lower levels of recursion, nor compute the final answer.) **Show** your work.

KARATSUBA(55 , 57)

KARATSUBA(32 , 37)

KARATSUBA(87 , 44)

$a_1 = 55$

$a_0 = 32$

$b_1 = 57$

$b_0 = 37$

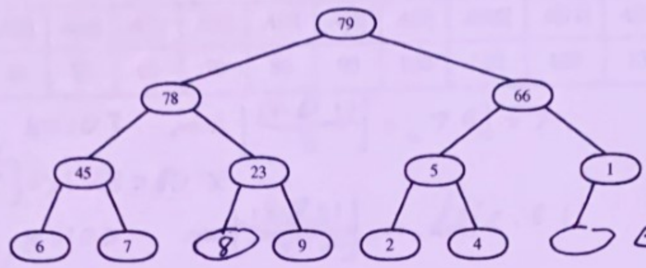
$ab = 10^{2m} a_1 b_1 + 10^m((a_1+a_0)(b_1+b_0) - a_1 b_1 - a_0 b_0) + a_0 b_0$

$c_2 \leftarrow$ Karatsuba(55, 57)

$c_0 \leftarrow$ Karatsuba(32, 37

$c_1 \leftarrow$ karatsuba$(\underbrace{a_1+a_0}_{\underbrace{55+32}_{87}}, \underbrace{b_1+b_0}_{\underbrace{57+37}_{44}}) - c_0 - c_2$

Problem 6 *(10 total points)*
[a] The following tree is *not* a heap. **Draw** an additional node to make the tree a heap. This node may contain any key you like, as long as the resulting tree is a heap. *(2 points)*



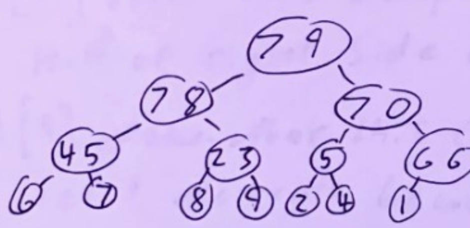
70 insert here

[b] **Fill in** the table below with the array representation of the corrected heap you created in part (a). *(2 points)*

A[0]	*A*[1]	*A*[2]	*A*[3]	*A*[4]	*A*[5]	*A*[6]	*A*[7]	*A*[8]	*A*[9]	*A*[10]	*A*[11]	*A*[12]	A[13]
79	78	66	45	23	5	1	6	7	8	9	2	4	70

[c] Suppose we insert the key 70 into your corrected heap. **List**, in order, the **key comparisons** and **swaps** that occur. **Draw** the final heap in tree form. *(6 points)*

$A[13] > A[6] \iff 70 > 1$ true
Swap $A[13]$ & $A[6]$
$A[6] > A[2] \iff 70 > 66$ true
Swap $A[6]$ & $A[2]$
$A[2] > A[0] \iff 70 > 79$ false, final comparison



final form of heap after insert & bubble up

Smaller Sub problems

Problem 7 *(8 points)*
Timothy uses binary search search for **103** in the following array. **List**, in order, the indices of the array elements to which the search key 103 is compared. **Explain** your answer.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]	A[11]	A[12]	A[13]	A[14]
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

Start $l=0,\ u=14,\ k=103$ $\quad m = \left\lfloor \frac{14-0+1}{2} \right\rfloor = \lfloor 7.5 \rfloor = 7$ $\quad 7$

$103 = A[7] \Rightarrow 103 = 80$ X

$l=8,\ u=14,\ k=103$ $\quad \cancel{m = \left\lfloor \frac{14-8+1}{2} \right\rfloor}$ middle is 11

$103 = A[11]$ X $\quad 103 < 120$

$l=8 \quad u=10 \qquad m=9$

$103 = A[9]$ X $\quad 103 > 100$

~~$l=8\ u=9\ m=$~~

Compare to last element A[10]

$103 \neq A[10] = 110$

A[7] after this comparison we eliminate left half of array because 103 > A[7]

A[11] after this comparison we eliminate right half of right side because 103 < 120

A[9] ~~of the~~ after this comparison we eliminate remaining left elements because 103 > 100

A[10] This is the final comparison & $103 \neq 110$ upon recursion the base case of $l > u$ is reached

Problem 8 *(8 total points)*
Solve the following recurrences using the Master Theorem.
If the Master Theorem does not apply, **write** "does not apply".

[a] $T(n) = 16T(n/4) + n^3$ *(2 points)*

$T(n) \in \Theta(n^3)$

$a = 16$
$b = 4$
$d = 3$

$16 < 4^3$

[b] $T(n) = 16T(n-4) + n^2$ *(2 points)*

~~es~~ does not apply
no constant "b"

[c] $T(n) = 16T(n/4) + n^2$ *(2 points)*

$a = 16$
$b = 4$
$d = 2$

$16 = 4^2$ $T(n) \in \Theta(n^2 \log n)$

[d] $T(n) = 17T(n/4) + n^2$ *(2 points)*

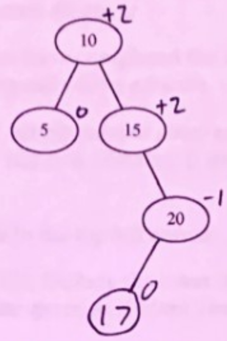
$a = 17$
$b = 4$
$d = 2$

$17 > 4^2$

$T(n) \in \Theta(n^{\log_4 17}) \simeq \Theta(n^2)$

Problem 9 *(10 total points)*

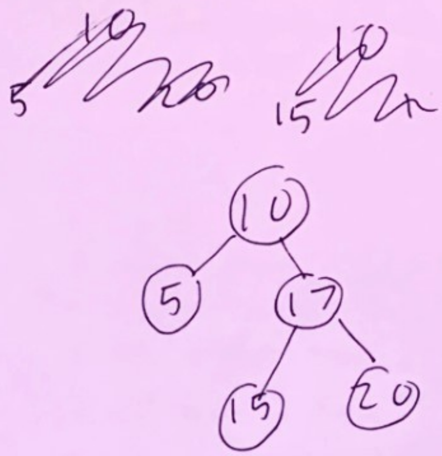
[a] **Draw** the new node created by inserting a 17 into the following AVL tree. Do not rebalance yet. *(2 points)*



[b] **List** the rotation or rotations needed to restore the tree to balance, if any. For each one, include the direction of rotation (left or right) and the node at the rotation should be done. *(2 points)*

Rebalance(15), RotateLeft(20), RotateRight(15)

[c] **Draw** the final, rebalanced tree. *(6 points)*



Submission checklist Be sure to:

- ☐ **Download** the exam within the allotted 72-hour window.
- ☐ **Write** the time you downloaded the exam on the cover page. *This is primarily for your own reference; we will verify the actual access time for your exam document.*
- ☐ **Sign** the cover page to confirm that you have completed the test using only the allowed resources. *Submissions for which this signature is missing will receive a 0 score.*
- ☐ **Include** the required sheet (8.5″ by 11″, single side) of notes as the final page of your submission. *You are required to use exactly one sheet of notes. Not zero. Not two. If you do not feel the need for notes, you may use a blank sheet.*
- ☐ **Label** your notes page with your name in the top left corner.
- ☐ **Scan** the exam back into a single PDF file. Include the cover sheet but omit the formula sheets. *This may be done, for example, using the Google Drive app on an Android Device, or the Notes app on an iOS device, or even an old school flatbed scanner.*
- ☐ **Upload** the completed exam to

  http://dropbox.cse.sc.edu/

  at most two hours after first accessing the download link. Submissions after the expiration of this two-hour window will receive a 0 score.

Decrease & Conquer: Smaller Sub problems
L by const $n \Rightarrow n-1$, by con. fact. $n \Rightarrow n/2$, variable size $n \Rightarrow < n$
Insertion Sort $\Theta(n^2)$ worst $\Theta(n)$ best, sorted array
Binary Search $\Theta(\log n)$

QuickSelect uses Partition to select element
$C(n) = \Theta(n) = BC = AC$
$WC = \Theta(n^2)$

Array Partition $i \leftarrow 1,\dots,n-1$ or $i \leftarrow LB,\dots,RB$

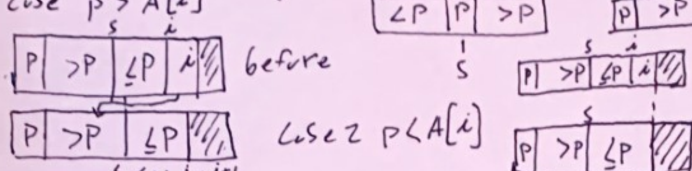
case $p > A[i]$ before

case 2 $p < A[i]$

$s_1 = s+1$, $i_1 = i+1$; $i = i+1$

[<P | P | >P]

at end Swap $A[0]$ & $A[s]$



$C(n) = \sum_{i=1}^{n-1} 1 = n-1-1+1 = n-1 \in \Theta(n)$

Divide & conquer
Solve several smaller probs recursively, combine results

The Master Theorem, constants a b d
$T(n) = aT(n/b) + f(n)$ & $f(n) \in \Theta(n^d)$
$a < b^d$, $T(n) \in \Theta(n^d)$
$a = b^d$, $T(n) \in \Theta(n^d \log n)$
$a > b^d$, $T(n) \in \Theta(n^{\log_b a})$
note: in polynomial like n^3+n-1, 3 takes precedence

Problem size n
SProb - size n/2 - SProb
recursion
Solution Solution
final sol.

MergeSort, sort first half then second, then combine
$T(n) \to \Theta(n \log n)$
A [2|1|4|8]
B [2|1] C [4|8] → B [1|2] C [4|8]
A [1|2|4|8]

QuickSort, Partition array putting pivot in final index, recursively sort L & R $\Theta(n \log n)$, use l & r, boundaries
BC, WC $\in \Theta(n^2)$, Avg C $\in \Theta(1.39 n \log n) \approx \Theta(n \log n)$

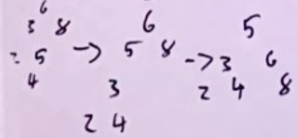
Karatsuba: $m = \lfloor n/2 \rfloor$
$a = 10^m a_1 + a_0$
$b = 10^m b_1 + b_0$
$T(n) = 3T(n/2) + \Theta(n) = \Theta(n^{1.58}) < \Theta(n^2)$ "gradeschool" efficiency

multiply n nums $A = [1|2|3]$
$\Theta(n)$ $A \times 10^3 = [1|2|3|0|0|0]$
$ab = 10^{2m} a_1 b_1 + 10^m((a_1+a_0)(b_1+b_0) - a_1 b_1 - a_0 b_0) + a_0 b_0$
$\Rightarrow c_2 = a_1 b_1$; $c_0 = a_0 b_0$ $\to ab = 10^{2m} c_2 + 10^m c_1 + c_0$
$c_1 = (a_1+a_0)(b_1+b_0) - c_2 - c_0$

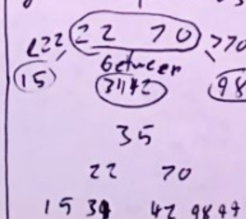
Transform & Conquer

Transform the problem into something easier to solve
- simpler instance of same prob
- different representation of same prob
- instance of dif. prob.

BST, one key each node K>L K<R
Insert & Search $\Theta(\log n)$
AVL BST, guarantee height of children differ by 1 or less
BF = LST - RST
Balance takes at most 2 rotations

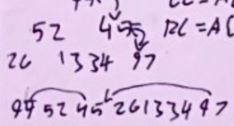


2-3 Tree → B-Tree
2 node 1 key 2 kids
3 node 2 keys 3 kids
Insert 35
Leaves same level, grows upwards
Search for correct leaf, add the new leaf to that leaf



if more than 2 keys split into 2 nodes & push one up

Heap, BST one key each node essentially complete, key > children
$\Theta(\log n)$
root = A[0] parent = $A[\lfloor (i-1)/2 \rfloor]$
LC = A[2i+1]
RC = A[2i+2]



Heap Sort, store
Extract $\Theta(n \log n)$
$\Theta \log n$
insert: new leaf @ right, "bubble up"
Extract: pop off move last leaf to front bubble down → largest child