

# A10: DIVIDE AND CONQUER

Nothing is particularly hard if you divide it into small jobs. —Henry Ford

Course: CS 5002

Fall 2018

Due: 28 Nov 2018, Midnight

## OBJECTIVES

---

After you complete this assignment, you will be comfortable with:

- Bounds of recurrence relations
- Recurrence trees
- Master Theorem
- Divide and Conquer algorithm analysis

## RELEVANT READING

---

Rosen:

- Chapter 8.3: Divide-and-Conquer Algorithms and Recurrence Relations

Cormen:

- Algorithms Unlocked, pp 40-59 (mergesort, quicksort)

## NEXT WEEK'S READING

---

- Chapter 5: Induction and Recursion

## EXERCISES

---

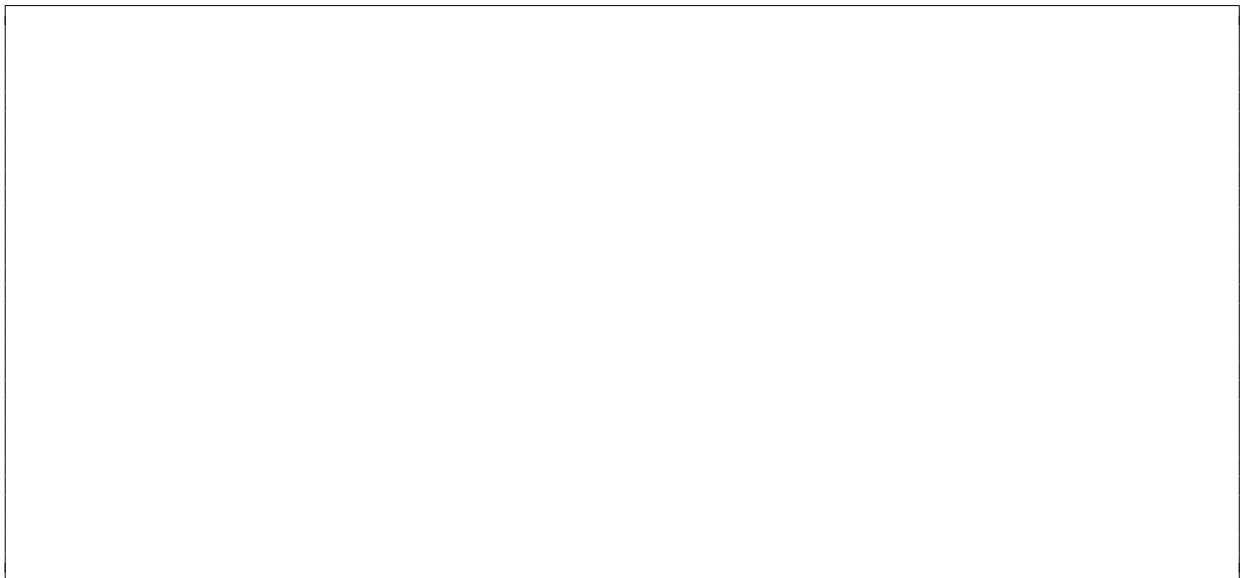
### Question 1

Draw a recurrence tree (to at least 3 levels, including the root) for the following recurrence relation, and answer the following questions.

$$T(n) = 3T(n/2) + T(n/4) + n/2$$

- (a) Draw the recurrence tree (to at least 3 levels including the root)

(5)



(b) What is the total amount of work done on the 3<sup>rd</sup> level of the tree (if the root is the first)? (2)

(c) How deep is the tree? (2)

(d) How many leaves are in the tree? (2)

(e) What is the amount of work done in the tree, EXCEPT for the leaf/bottom level? Show your work. (2)

(f) What is a bound for this recurrence? (2)

**Question 2**

Use the Master Theorem to find a bound for these recurrences. For each, indicate  $a$ ,  $b$ ,  $f(n)$ ,  $n^{\log_b a}$  and which case applies.

(a)  $T(n) = 4T(n/4) + 5n$

(3)

(b)  $T(n) = 4T(n/5) + 5n$

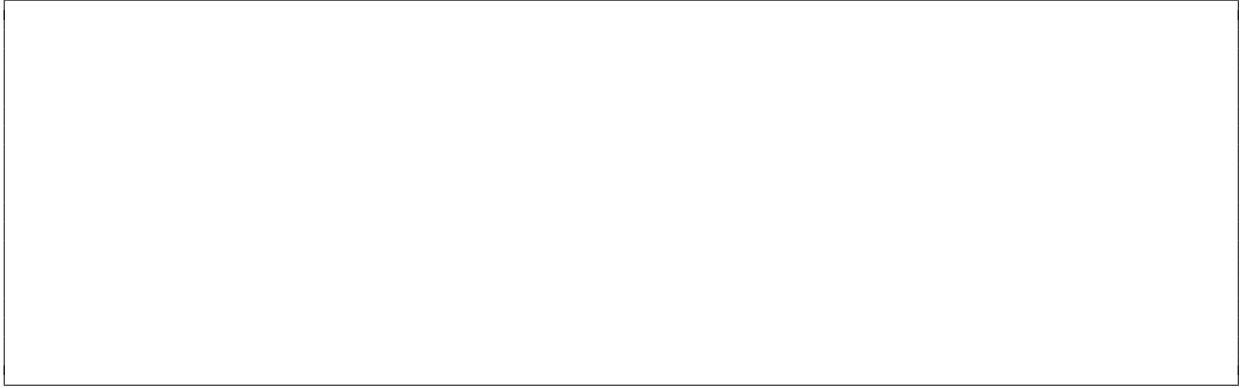
(3)

(c)  $T(n) = 5T(n/4) + 4n$

(3)

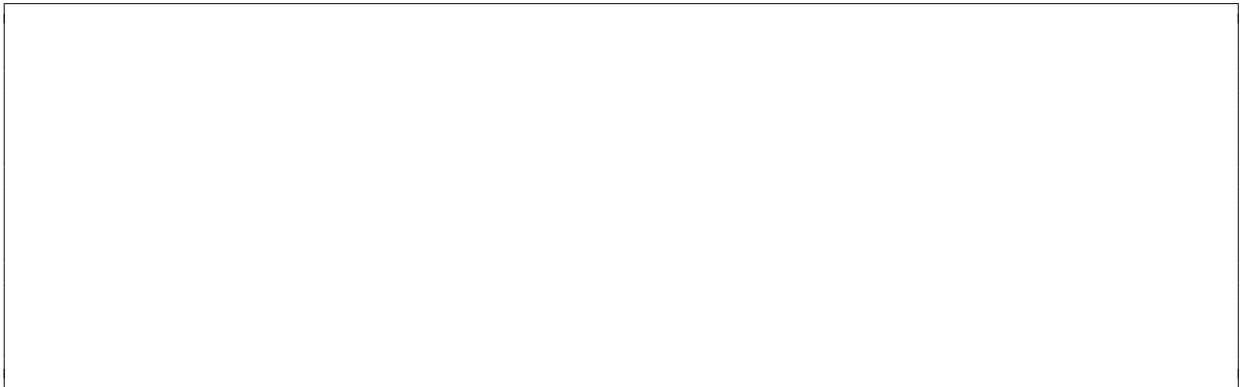
(d)  $T(n) = 25T(n/5) + n^2$

(3)



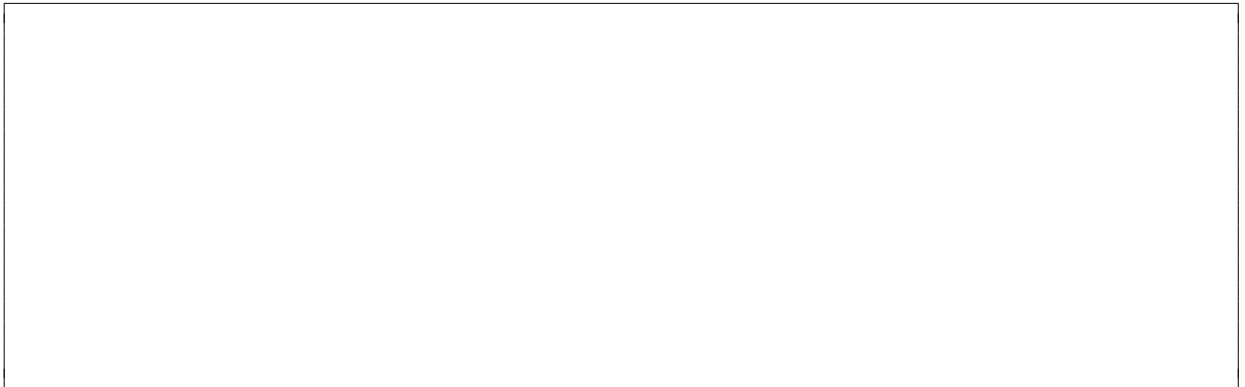
(e)  $T(n) = 4T(n/5) + \lg n$

(3)



(f)  $T(n) = 4T(n/5) + \lg^5 n$

(3)



(g)  $T(n) = T(\sqrt{n}) + 2T(n/5) + T(n/10) + 4n$

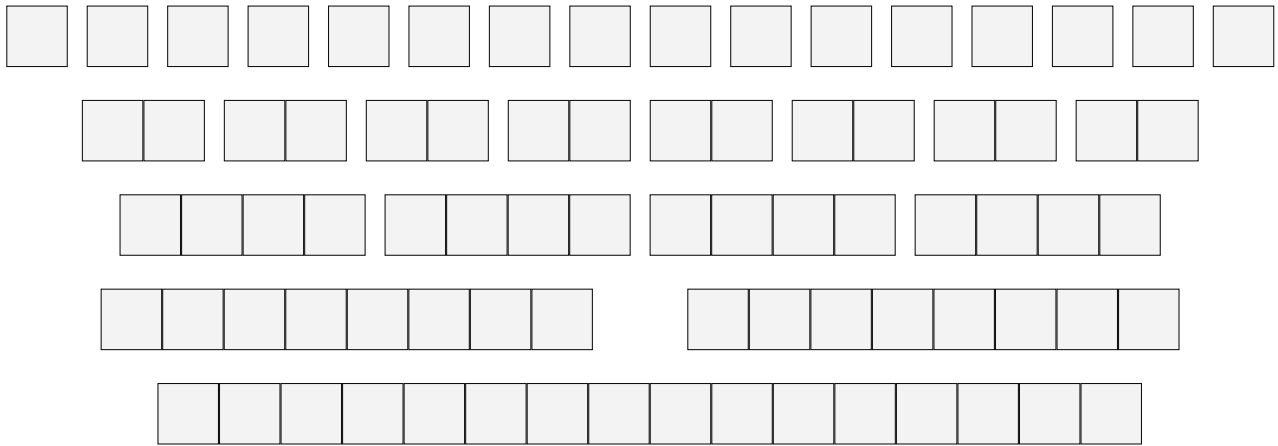
(3)

**Question 3**

For the given array, show the effect of MERGE on each step. Assume the divide has already been done.

(5)

[53, 24, 86, 92, 5, 13, 52, 68, 18, 21, 14, 38, 75, 56, 9, 44]



#### Question 4

This problem pertains to finding the closest pair of points in a 2-dimensional plane. The closest pair of points is defined as the pair that has the smallest Euclidean distance between them.

Recall, the Euclidean distance between two points  $(x_i, y_i)$  and  $(x_j, y_j)$  is  $\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$

(Note: More on closest pairs can be found in the book (pg 532); also some slides here (<https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/closepoints.pdf>) builds a nice intuition, starting on slide 61).

**Input:** An array of 2-tuples  $(x, y)$  such that  $x$  is the x-coordinate of the point, and  $y$  is the y-coordinate of the point.

**Output:** The distance between the closest two points.

CLOSESTPAIR( $P$ ):

```
1  if P.length == 2:
2      return DIST(P[1],P[2])
3  splitX = COMPUTEVERTICALSPLIT(P)
4  dist1 = CLOSESTPAIR(LeftHalf(P, splitX))    ▷ Divide into subproblems
5  dist2 = ClosestPair(RightHalf(P, splitX))
6  d = MIN(dist1, dist2)
7  S = FINDPOINTS(P, splitX, d)                ▷ Merge the solved subproblems
8  SORTBYY(S)
9  for i = [1:S.length]:
10     for j = [1:15]:                          ▷ Seems random; see refs for rationale.
11         d = min(DISTANCE(S[i], S[j]), d)
12  return d
```

- (a) The algorithm provided for this is pretty vague. Write pseudocode to implement the step COMPUTEVERTICALSPLIT that finds an  $x$  value such that half the points are to the left of  $x$  and the other half are to the right of  $x$ . Indicate the runtime.

**Input:** An array of 2-tuples  $(x, y)$  such that  $x$  is the x-coordinate of the point, and  $y$  is the y-coordinate of the point.

**Output:** A value  $n$  such that half of the input points have an  $x$ -coordinate smaller than  $n$ , and half the input points have an  $x$ -coordinate larger than  $n$ .

(10)

- (b) The CLOSESTPAIR algorithm provided above finds the smallest distance between two pairs of points. How could this algorithm be modified to return the pair of points that are closest together?

(5)

- (c) Apply the CLOSESTPAIR algorithm provided above, with your specified modification from the previous part, to find the closest pair of points from this list of points:  $[(4, 3), (9, 7), (10, 1), (7, 2), (6, 8), (3, 5), (4, 7), (7, 5)]$ . Show your work, including the value of  $splitX$ ,  $dist1$ , and  $dist2$  for the first loop through the algorithm.

(5)

### Question 5

Believe it or not, it's almost snowboard season. And every year, there's always at least one good story that comes out of the season (ask Ian!!). Last year, the event I remember the most is when my kid came back from the mountain one night after spending the day up there with his class from school.

A parent who had gone up with the kids told me about a big mess with the gear when they got up to the mountain. In the morning, they had put all the gear in a big truck. The kids stacked the snowboards up nicely, but then just threw their boots in the truck in a big pile. By the time they got up there, they had no idea which boots belonged to who. There were no names, and all the boots looked the same, and all the kids had feet about the same size (no one could distinguish the boots just by looking at them). The only way to know if a pair of boots belonged to a kid was for that kid to try them on. The boots either fit perfectly, or didn't fit at all (the kid could report whether the boots were too big or too small). Every kid had one and only one pair of boots.

(a) When the parent told me what they did to get each kid their boots, I said "Dude, that's an  $O(n^2)$  process!".

Provide an algorithm they likely used to match each kid to their boots. Describe in 1-2 sentences the approach, and then provide pseudocode. Indicate the input and output of your algorithm.

(5)



(b) Another parent standing nearby said “Well, it would take  $2n - 2$  comparisons to find the boots that fit the largest kid...even though we don’t know which kid is the largest one.” Show that this is a valid claim by providing an algorithm to find the largest boots and the largest kid. Describe in 1-2 sentences the approach, and then provide pseudocode. Indicate the input and output of your algorithm.

(5)

(c) The first parent seemed to be pretty peeved that I wasn’t impressed. He said to me “Well, do you think you can do any better??. I said “Of course I can! I can do the same thing in  $O(n \log n)$ !” He said “Fine— prove it! There’s a mess of boots in the truck right now.”. Being old and grumpy, I wanted to get home and go to bed. So, I took him up on the challenge, and I got each kid their boots in  $O(n \log n)$  time. Can you come up with an  $O(n \log n)$  algorithm to match each kid to their boots? Describe in 1-2 sentences the approach, and then provide pseudocode. Indicate the input and output of your algorithm.

(10)

### Question 6

Suppose that each person in a group of  $n$  people votes for exactly two people from a slate of candidates to fill two positions on a committee. The top two finishers both win positions as long as each receives more than  $n/2$  votes.

- (a) Devise a divide-and-conquer algorithm that determines whether the two candidates who received the most votes each received at least  $n/2$  votes and, if so, determine who these two candidates are.

(10)

- (b) Indicate the runtime of the algorithm you provided above. If you use the Master Theorem, show your work and indicate the case. If you use a recurrence tree, show 3 levels of the tree.

(5)

## SUBMISSION DETAILS

---

Things to submit:

- Submit the following on Blackboard for Assignment 10:
  - The written parts of this assignment as a .pdf named “CS5002-[lastname]\_A10.pdf”. For example, Ben Bitdiddle’s file would be named “CS5002\_Bitdiddle\_A10.pdf”. (There should be no brackets around your name).
  - Make sure your name is in the document as well (e.g., written on the top of the first page).