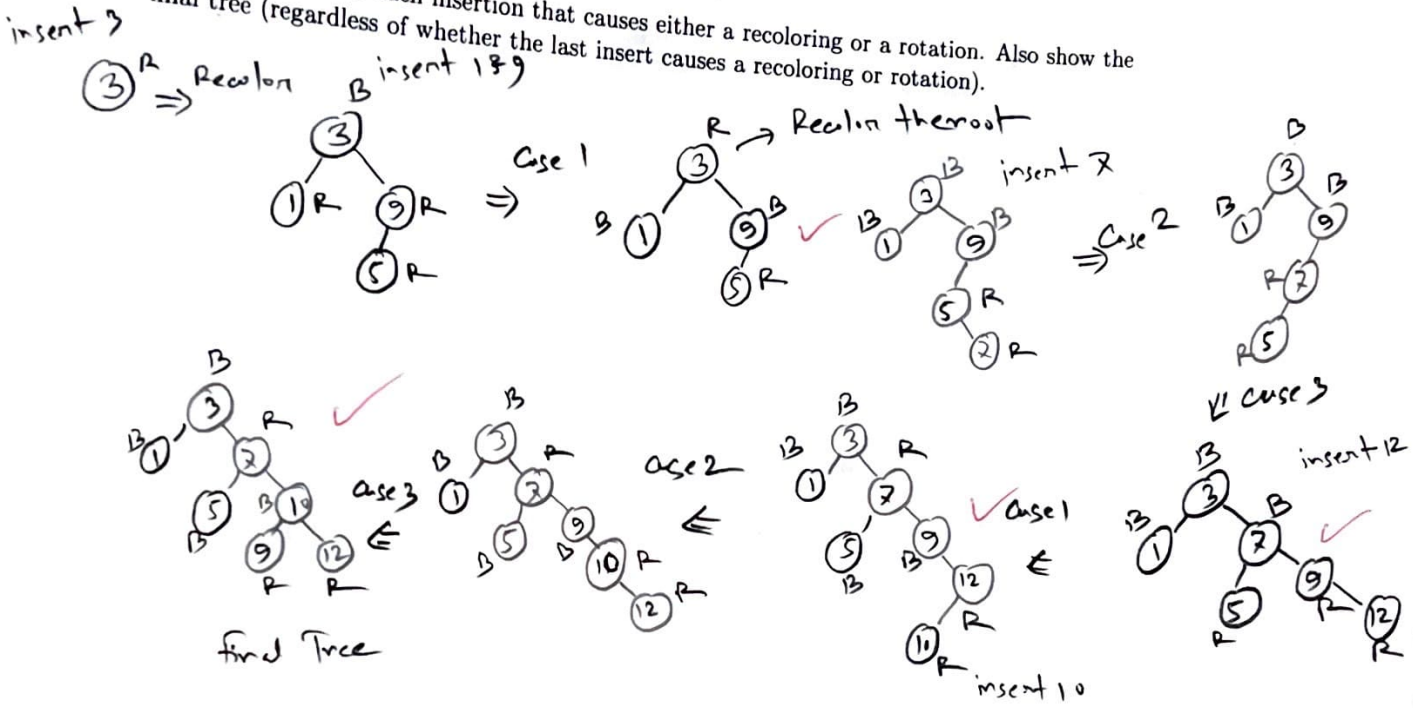
19

1 Red-Black Trees (19 Points)

Consider inserting the following values into an initially-empty red-black tree:

$$3, 1, 9, 5, 7, 12, 10$$

Show the tree after each insertion that causes either a recoloring or a rotation. Also show the final tree (regardless of whether the last insert causes a recoloring or rotation).



18

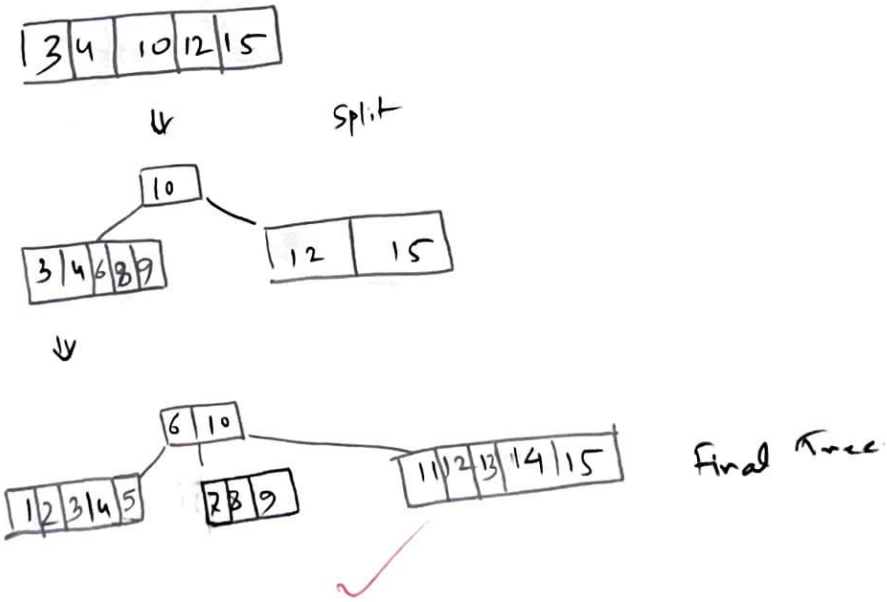
2 B-Trees (18 Points)

Consider making the following sequence of inserts into an initially-empty B-tree with $t = 3$.

4, 10, 15, 12, 3, 9, 6, 8, 5, 11, 7, 13, 1, 2, 14 — 15 elements.

Show the tree before and after each split (i.e. you do not need to draw it after each insertion if the insertion does not force a split), but do show the final tree regardless of whether the last insert caues a split.

$t = 3$ so, $2 \cdot 3 - 1 = 5$ elements in each node



Final Tree

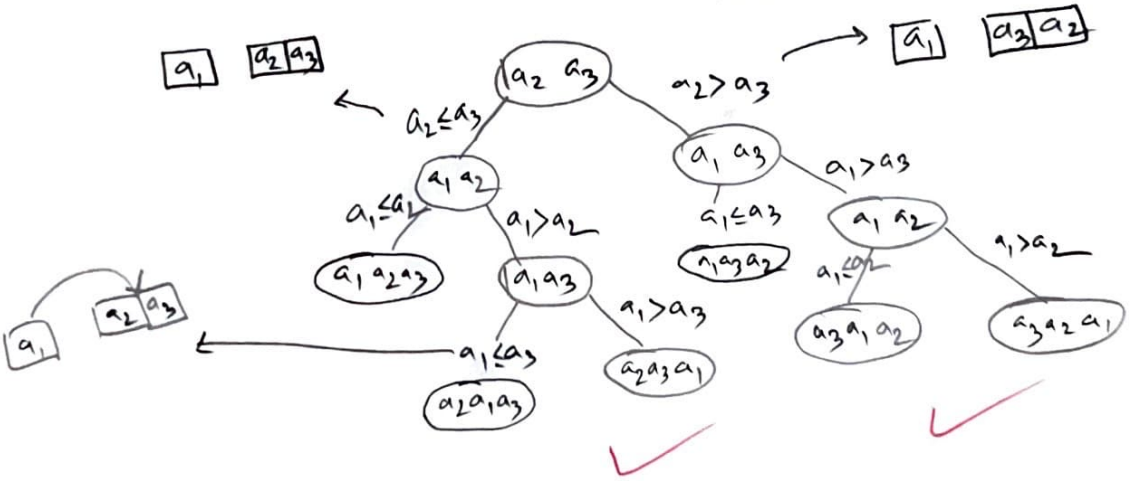
18

3 Decision Trees (18 Points)

Show the decision tree for the Merge Sort algorithm on an input $[a_1, a_2, a_3]$ of size 3 with the following assumptions about the implementation of the algorithm.

- If the array has an odd number of elements, then when we cut the array in "half", the extra element goes to the right subproblem (so on an array of size 3, the left subproblem would have 1 element, and the right subproblem would have 2 elements).
- The base case is size 1 (so a subproblem of size 2 would get split into two subproblems of size 1).

The base case is subproblem of size 1
so, the array after dividing until the base case is $[a_1]$ $[a_2]$ $[a_3]$ let's merge the right two first



15

4 Sorting Runtimes (15 Points)

Suppose we want to choose a sorting algorithm with the best worst-case running time possible for the assumed input. What sorting algorithm would you use? What would the running time of the algorithm be? Justify your answer.

1. A list of n sports teams according to their ranking.

   Ranking $k = n$ so, radix sort or counting sort with a run time of $O(n)$ ✓

2. An array of n rational numbers (which are numbers that can be represented as $\frac{a}{b}$ for integers a and b where $b \neq 0$).

   There is no bound for the rational numbers. so, merge sort with a runtime of $\theta(n \log n)$ ✓

3. A phone book consisting of n telephone numbers (e.g., XXX-XXX-XXXX).

   only 10 digits per phone number. so, radix sort with a runtime of $O(n)$ ✓

23

5 Dynamic Programming (30 Points)

Recall from class that a string A is a subsequence of a string B if A can be obtained by deleting 0 or more characters from B (without rearranging any of the letters of B. So for example, if $A = abc$ and $B = adbdcd$, then A is a subsequence of B (by deleting the d's from B). If we have $A = abc$ and $B = axcxbx$, then A is not a subsequence of B. Also, a string is a *palindrome* if it is spelled the same way forwards as it is backwards. Examples of palindromes: mom, dad, radar, racecar. Note that any string consisting of a single character is a palindrome. In this problem, we are given a string S of n characters and we want to find the *length* of a longest subsequence of S such that the subsequence is a palindrome.

Examples: If $S_1 = abcd$, then the longest palindrome subsequence has length 1 (any of the characters is a palindrome of length 1 and there is no subsequence with two or more characters that is a palindrome). If $S_2 = abca$, then the longest palindrome subsequence has length 3 (there are two optimal solutions: aba and aca).

1. Consider a brute force algorithm that considers every possible subsequence and checks to see if it is a palindrome. What is the running time of this algorithm?

making all possible subsets with n characters gives us $(2^n - 1)$ subsets. [considering the $\emptyset$ out of the subset] we can check whether a subsequence is a palindrome or not in $O(n)$ time. That gives us a runtime of $O(n \cdot 2^n)$

2. This problem can be solved in a similar way to how we solved the matrix chain problem in class, in that we can consider "subchains" of length 1, followed by "subchains" of length 2, etc. To that end, let $c[i, j]$ denote the length of an optimal solution considering only the substring of S starting from character i until character j. So for example, if $S = abcb$, then $c[2, 3]$ should be the length of the optimal solution for the substring bc. For the example $S = bcb$, what are the values $c[1, 1], c[2, 2], c[3, 3], c[1, 2], c[2, 3]$, and $c[1, 3]$?

$c[1,1] = 1$

$c[2,2] = 1$

$c[3,3] = 1$

$c[1,2] = 1$

$c[2,3] = 1$

$c[1,3] = 3$ [max(2,3)]

✓

bcb if adding makes it a palindrome then

max (including the next characters, not including)

3. Give a recursive definition for $c[i, j]$. That is, explain how $c[i, j]$ can be computed as a function of its subproblems. You do not need to give an implementation of the algorithm, only define how $c[i, j]$ can be computed recursively.

Base case: $c[i,j] = 1$ if $i = j$ [for a character of length one]

$$c[i,j] = \begin{cases} c[i,j-1] & \text{if taking character } j \text{ doesnot make the sequence a palindrome} \\ c[i,j-1]+1 & \text{if it makes the sequence a palindrome.} \end{cases}$$

✗

$$c[i,j] = \max_{\text{for all } k:\ i \le k < j} \left(c[i,k] + c[k+1,j] + 1 \right)$$

* when we add a character to the subchain it can either be a palindrome or not. if it's a palindrome then $c[i,j]$ will be j otherwise it will be the value of the previous subchain