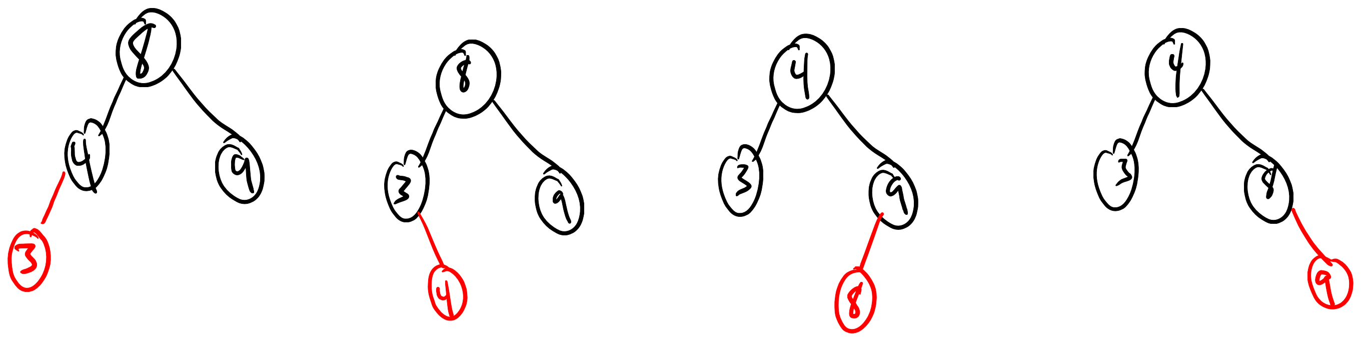
1 Red-Black Trees (18 Points)

In class we gave a set of 5 properties that a red-black tree must satisfy in order to be a valid red-black tree. We then gave an insertion algorithm that performs insertions in a way where we can guarantee that the properties will be maintained. There may be valid configurations of the tree that our insertion algorithm will never realize. For example after 3 insertions with the algorithm discussed in class, we will always end up with a black root with two red children. However, another valid configuration would be if the root had two black children.

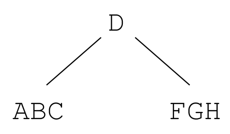
List all of the valid configurations of red-black trees that contain the following four values. Note that the order doesn't matter here, and we don't care about any specific insertion algorithm. We want all valid configurations where these 5 values are the nodes in the tree.

4 9 3 8

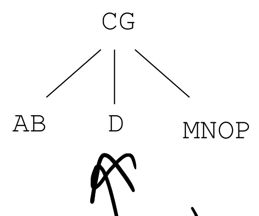


2 B-Trees (19 Points)

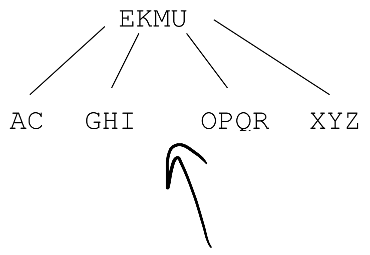
1. Which of the following trees are legal B-trees for $t = 3$? If it is not legal, then give a one or two sentence explanation for why it is not a legal B-tree.



Valid

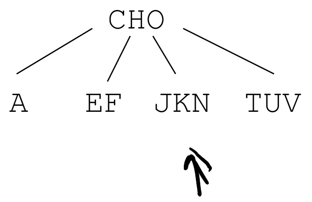


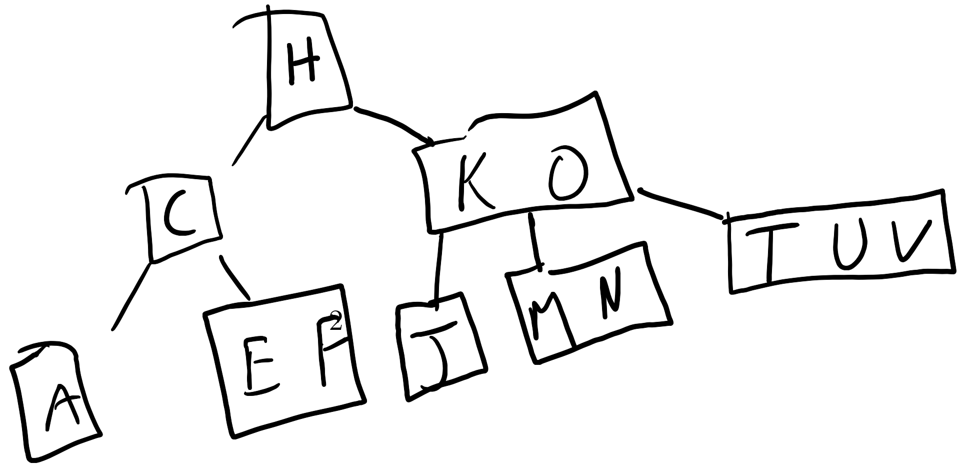
invalid non-roots need ≥ 2 valus



Invalid No Child betwen K&M

2. Consider the following B-tree with $t = 2$. Show the resulting B-tree after inserting M into the tree.

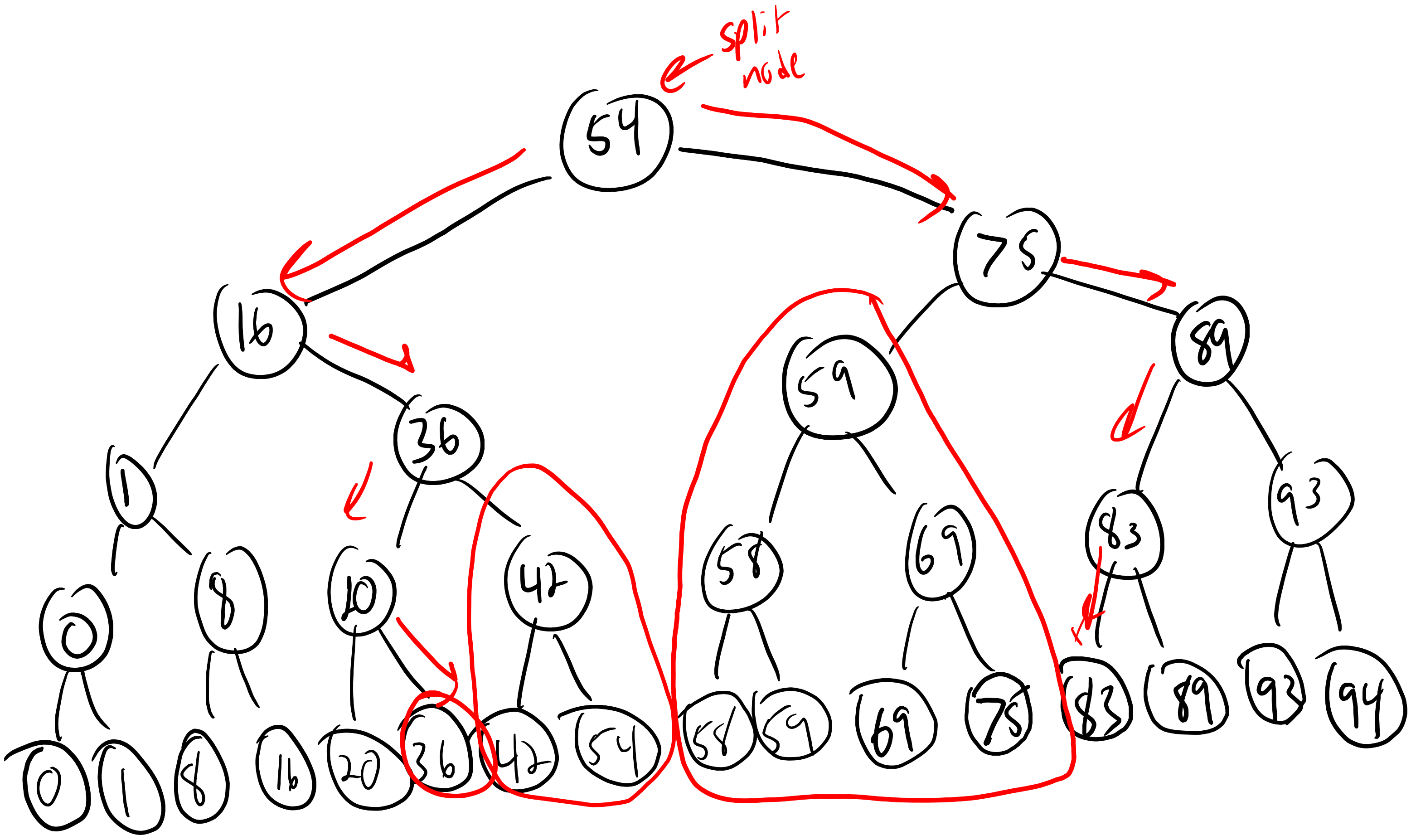




3 Range Trees (18 Points)

Consider a range tree on the 16 following 1D points:
0 1 8 16 20 36 42 54 58 59 69 75 83 89 93 94

1. Draw the 1D range tree.
2. Show how the search works on your tree when searching for the points in the interval [23, 75] (show any split nodes and search paths).



4 Sorting Algorithms (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. An array of n integer variables.

Assuming integer means unbounded, merge sort for $O(n \log n)$.

2. An array of n real numbers in the range [0,1].

Difference between smallest value and largest value is unbounded. $\Rightarrow$ Merge Sort $O(n \log n)$

3. An array of n students by their SAT scores (an integer between 400 and 1600).

Constant range $\Rightarrow$ Counting or Radix Sort $\Rightarrow O(n)$.

5 Dynamic Programming (30 Points)

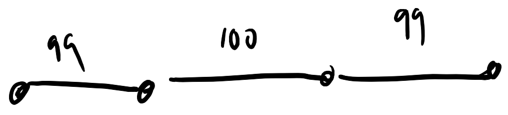
Suppose we have a *path graph* with n vertices, that is, a graph with n vertices $v_1, v_2, \ldots, v_n$ and $n-1$ edges $e_i := \{v_i, v_{i+1}\}$ for each $i \in \{1, 2, \ldots, n-1\}$. That is, v_1 is only connected to v_2, v_2 is only connected to v_1 and v_3, etc. Suppose each edge e_i has a positive weight w_i. We are wanting to pick edges to be in our solution that have large value, but suppose we are not allowed to pick two edges that share a common vertex as an endpoint (e.g., we cannot pick e_3 and e_4 because they both share v_4 as an endpoint). Subject to these constraints, we want to pick a set of edges that maximizes the weights of the chosen edges.

Example: Suppose $n = 5$ and therefore we have 4 edges with weights $[10, 1, 1, 10]$. The optimal solution would be to pick e_1 and e_4 for a value of 20. We cannot pick e_2 or e_3 because they would conflict with e_1 and e_4 respectively.

1. Consider a brute force algorithm that considers every possible subset of edges to pick. What is the running time of this algorithm?

$O(n2^n)$

2. A potential greedy algorithm for this problem would be to greedily pick the heaviest feasible edge until we cannot pick any more edges. Give a counterexample to show that this algorithm will not always produce an optimal solution.



Greedy: 100
OPT: 198

3. Let $a[i]$ denote the optimal solution that considers only the edges e_1 through e_i. Consider the following array of 5 edges. What is $a[1], a[2], a[3]$, $a[4]$, and $a[5]$ for this example?

   $C = [9, 8, 9, 3, 3]$.

$a[1] = 9$

$a[2] = 9$

$a[3] = 18$

$a[4] = 18$

$a[5] = 21$

4. Give a recursive definition for $a[i]$. Do not forget the base case.

$a[1] = W_1$

$a[2] = \max\{W_1, W_2\}$

$a[i] = \max\{a[i-1], a[i-2] + W_i\}$ if $i \geq 3$.