CSIT 5500 Advanced Algorithms

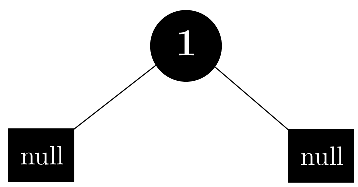
2020 Spring Semester

Written Assignment 1 suggestd solution

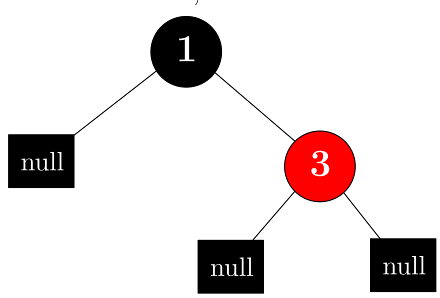
1. (10 points)

   (a) (7 points)

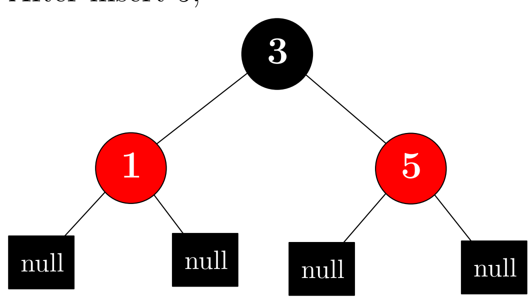
   After inserting 1,



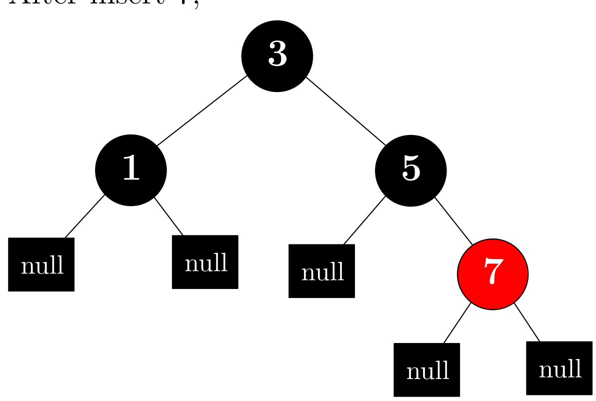
   After insert 3,



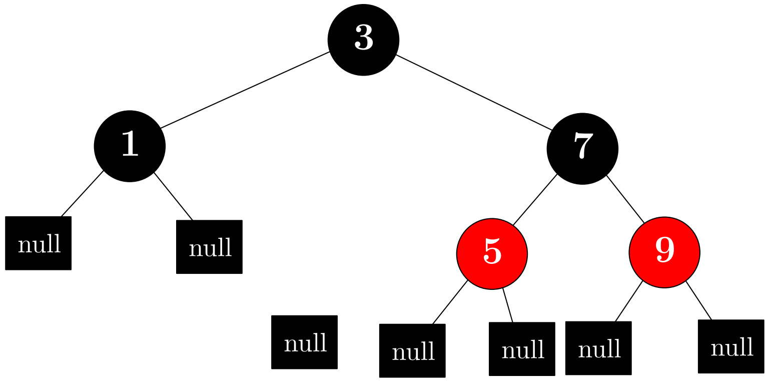
   After insert 5,



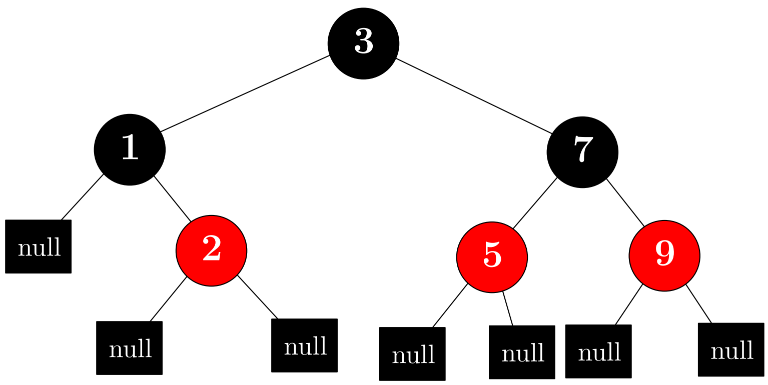
   After insert 7,



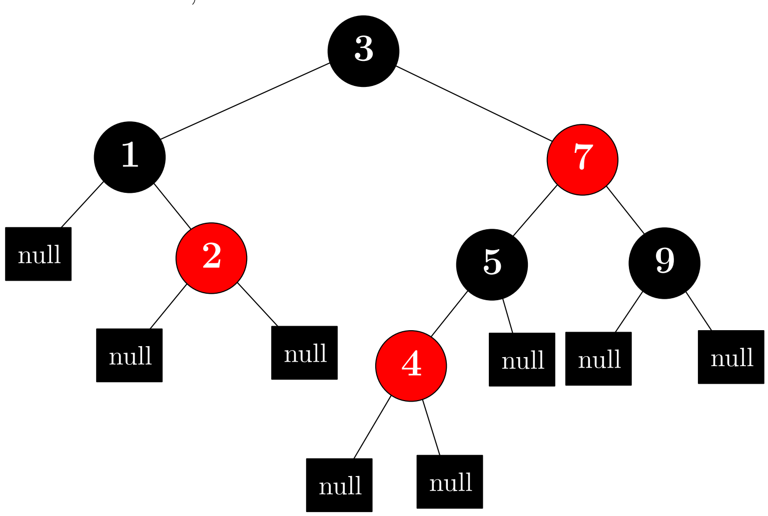
After insert 9,



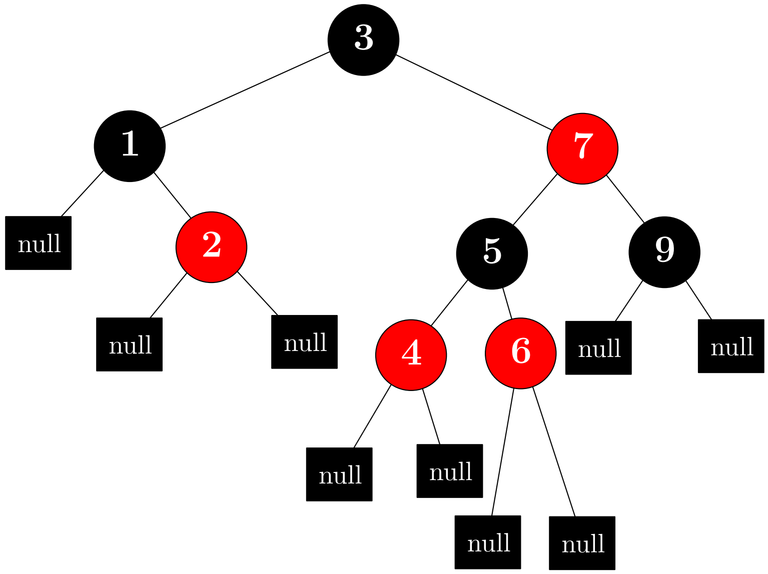
After insert 2,



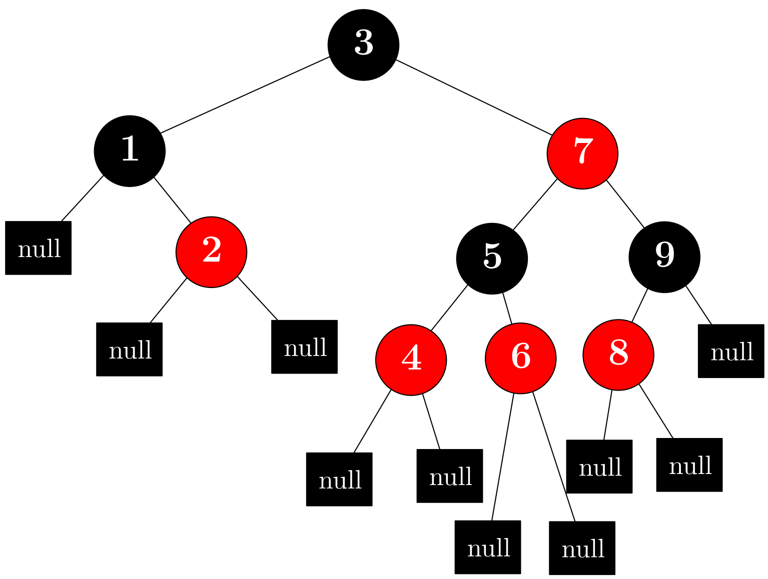
After insert 4,



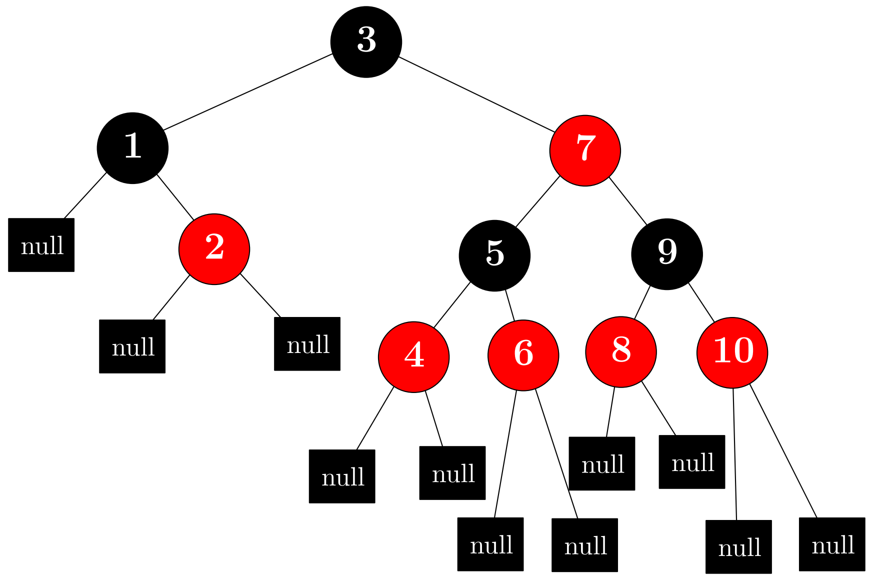
After insert 6,



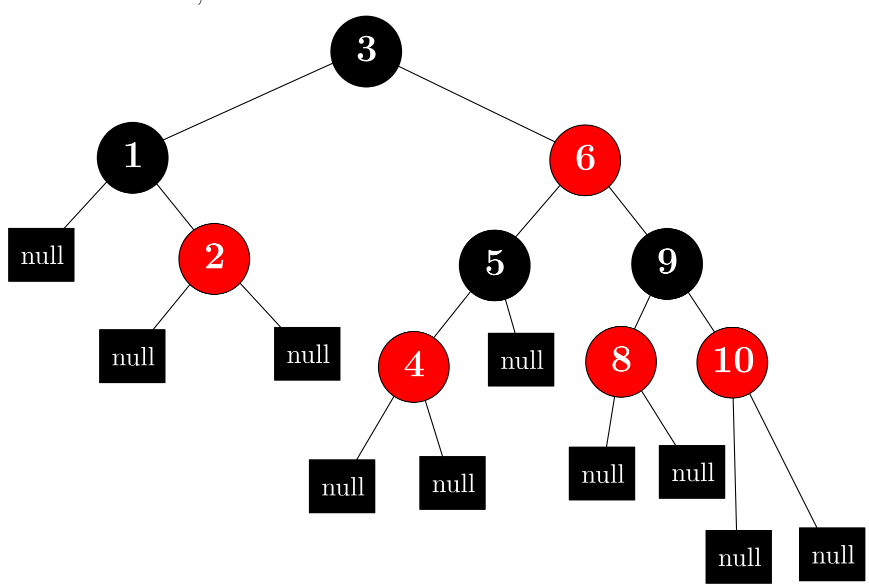
After insert 8,



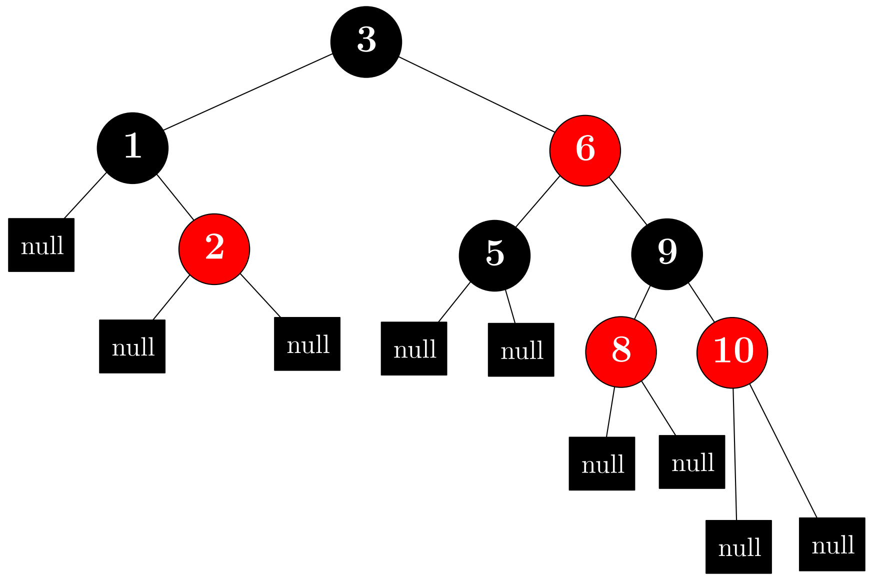
After insert 10,



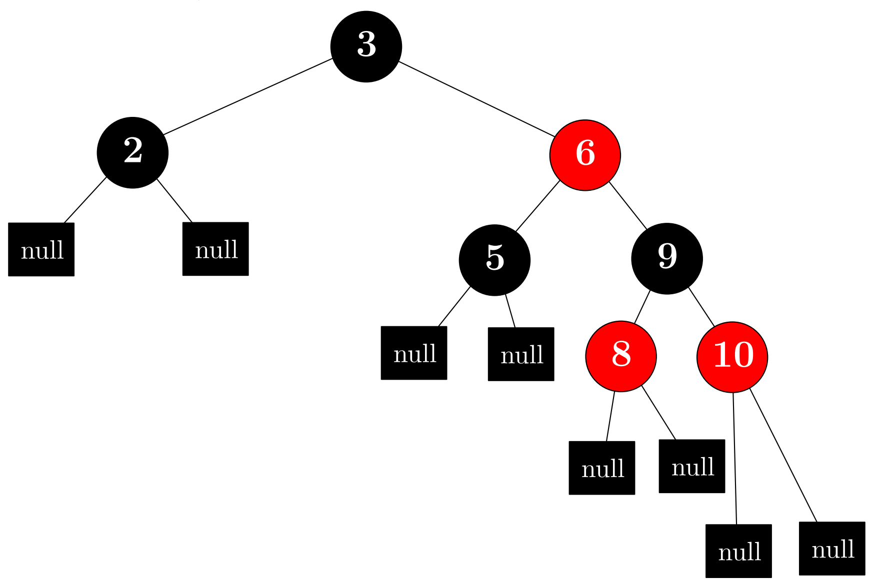
(b) (3 points)
After delete 7,



After delete 4,



After delete 1,



2. (10 points) Set a counter c for counting the number of inversion. The initial value of c is 0. We perform merge sort for sorting the n integers in ascending order. During the merge operation for merging a left sublist L and a right sublist R, there are two indices k and l for comparing $L[k]$ and $R[l]$. If $L[k] > R[l]$, then we increase c by $|L| - k + 1$. This is because $L[j] > R[l]$ for $k \leq j \leq |L|$, so each such $L[j]$ forms an inversion with $R[l]$. There are exactly $|L| - k + 1$ such $L[j]$'s. Morever, since $R[l]$ will be moved into the merged list, $R[l]$ will not be counted in any inversion in the future. So there is no danger of overcounting. The running time for each update of c is $O(1)$, which is absorted by the comparison of x_i and x_j. So, the total running time is $O(n \log n)$.

3. (10 points) Recall that every level of a complete binary tree is full except possibly the bottommost level. So, for a complete binary tree of n nodes with h height, the number of nodes from level 0 to level $h-1$ is $2^0 + 2^1 + \cdots + 2^{h-1} = 2^h - 1$ which is less than n. So, we have $2^h \leq n+1$, and thus $h \leq \log_2(n+1)$. Therefore, the height is $O(\log n)$.