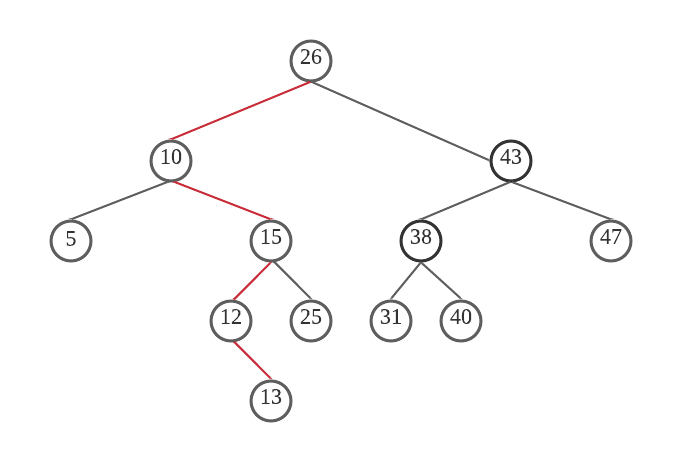
# Assignment 5

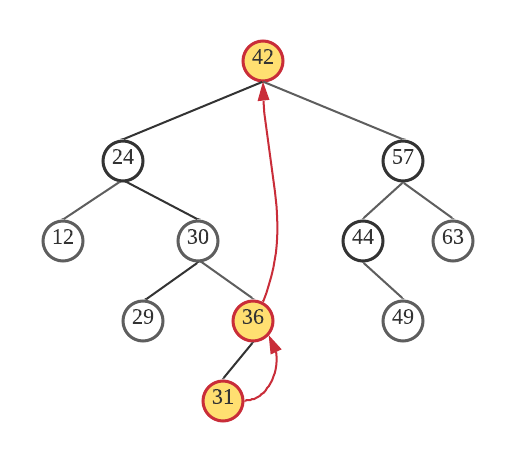
## Question 1

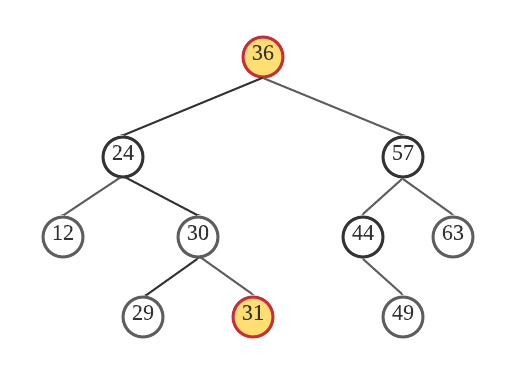
1. We need to add the entry with key = 13. So, we start checking the value from root
2. 13 < 26, so go to left child tree
3. 13 > 10, so go to right child tree
4. 13 < 15, so go to left child tree
5. 13 > 12, so go to right child tree, finish insert 13 at here



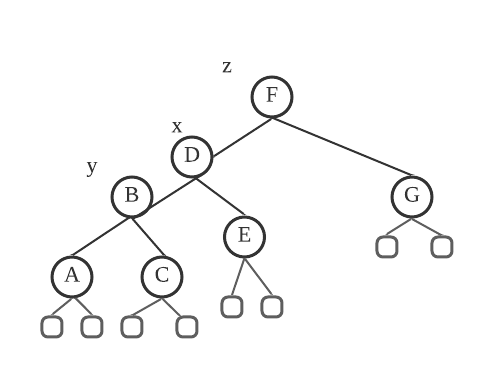
## Question 2

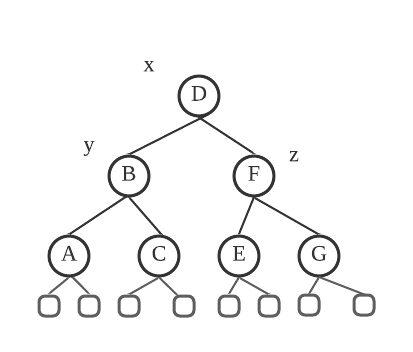
1. We need to delete the entry with key = 42, so we find the node r that has the largest key that is strictly less than p’s key.
2. So, the predecessor is 36, and let 36 replace 42
3. And node 36 is removed and the subtree rooted at 36’s left child is promoted to 36’s position





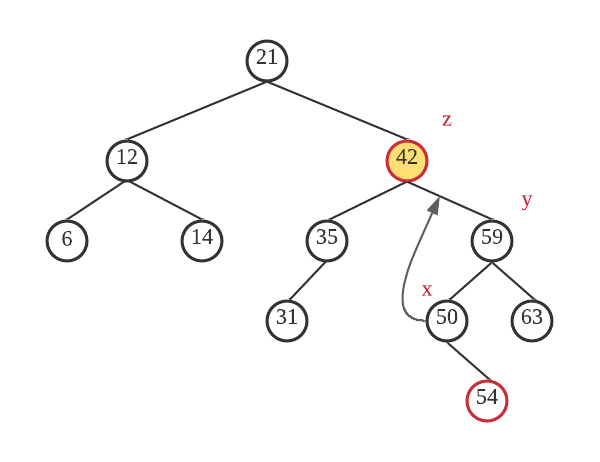
## Question 3

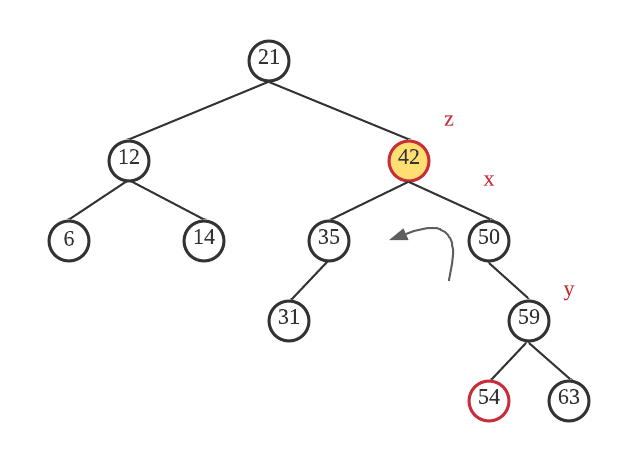


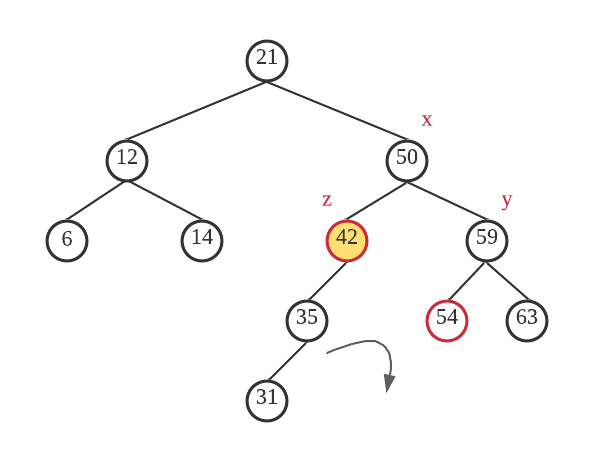


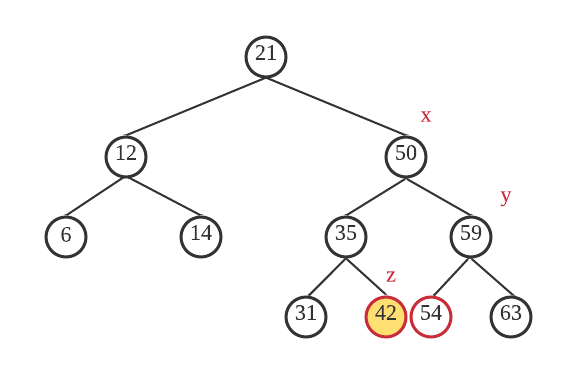
## Question 4

After inserting 54, searching a node z that the lowest ancestor of p that is unbalanced. That is 42 (z), y is z’s child with the greater height (59, y), x is y’s child with the greater height (50, x)









## Question 5

1. Encode follow the path from the root to the leaf

B: 1111, D: 011, E: 010, G: 10, C: 00

BDEGC 11110110101000

1. Decode 010111010011

010: E, 1110: A, 10: G, 011: D

EAGD

## Question 6

P (4, 1) = (P (4, 0) + P (3, 1)) / 2 = (0 + 1/8) / 2 = 1/16

P (2, 4) = (P (2, 3) + P (1, 4)) / 2 = (11/16 + 15/16) / 2 = 13/16

## Question 7

Both quick sort and heapsort algorithms are got from https://www.geeksforgeeks.org/quick-sort/

https://www.geeksforgeeks.org/heap-sort/

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n/algorithm | 10000 | 20000 | 30000 | 40000 | 50000 |
| Insertion | 18 | 94 | 77 | 135 | 218 |
| Merge | 3 | 2 | 3 | 3 | 5 |
| Quick | 2 | 1 | 2 | 3 | 4 |
| Heapsort | 3 | 3 | 3 | 4 | 6 |
| n/algorithm | 60000 | 70000 | 80000 | 90000 | 100000 |
| Insertion | 306 | 409 | 545 | 708 | 852 |
| Merge | 6 | 3 | 4 | 8 | 12 |
| Quick | 4 | 6 | 6 | 7 | 8 |
| Heapsort | 7 | 7 | 8 | 10 | 11 |

Insertion sort has the biggest sorting time of these four sort algorithms.

Other three algorithms’ sorting time has not a big difference. As the amount of data increases, the speed of heap sort is slightly higher than the other two algorithms. Quick sort is slightly faster than merge sort when the data volume is less than 50,000, but it is slightly slower than merge sort when the data volume rises above 70,000. At the same time, the time of the algorithm is different each time the program is executed. Generally speaking, the execution time of the three sorting algorithms of merge/quick/heap is not much different