Problem 3

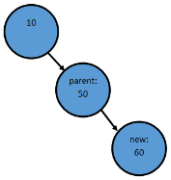
a) Merge sort, as it is stable (will maintain order of duplicates) and will be fast with a worst case of Θ(N*log(N)) where N is the number of elements (books) in the array. Further efficiencies can be made by if we know where in the array the already sorted chunks (boxes) are as we could simply perform merging of the smallest chunks up until we are left with the sorted array.

b) Selection sort as it doesn't need o be too fast but performs very little 'record swaps' (At most Θ (N)) where N is the number of records and thus will be energy efficient for this case.

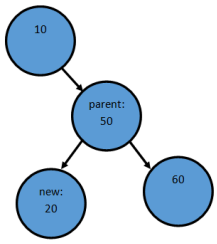
c) Counting sort, as it is very fast Θ(N + K), where n is the number of stars (50) and k is the range (10-0 = 10). Thus only K auxiliary memory is used and as K<N (10<50) we satisfy the condition of using no more memory other than an additional array of less than N.

Problem 4

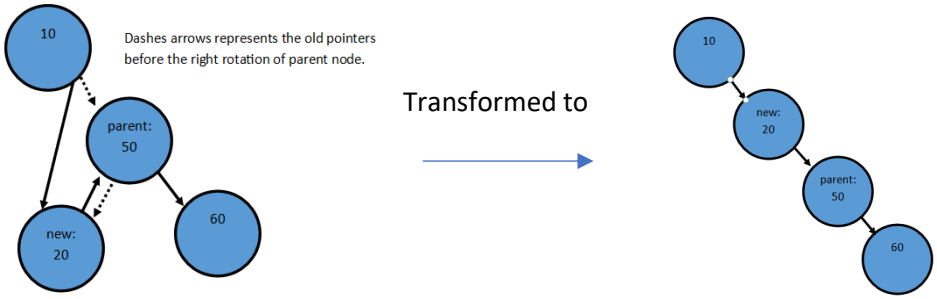
i) The Stickify function will work by if a left child is added rotate the root right to make a straight tree. If a right node is added to the BST before the Stickify call then no action needs to take place as it will maintain its stick structure.



However, if a left node is added this will break the stick structure of the nodes.



Thus we must rotate right the parent node. Now new points to parent, the parent no longer points to new and the parent of the parent node points to new instead of the parent like so. After re-arranging the diagram the stick structure is revealed.



ii) The pseudocode would be as follows:

function Stickify(root, new):

    if new.value < root.value **then**

        RotateRight(root)