1.

```
Algorithm duplicatesBT(node):
    if node is None:
        return False
    nodeStack = []
    previous = Null
    current = node
    while nodeStack is not empty or current is not Null:
        while current is not Null:
            nodeStack.append (current)
            current = current.left
        current = nodeStack.pop ()
        if previous is not None and current.value = previous:
            return True
        previous = current.value
        current = current.right
    return False
```

My algorithm runs in O(n) time complexity since each node is visited once only, the outer loop runs as long as there are nodes in the stack, the inner loop pushes nodes to the stack, and finally each node is evaluated by popping it from the stack to be compared with previous node.

2.

i) The advantage of this technique is that searching for an entry becomes faster since when a negative value is found, the search is terminating. The space complexity remains O(n) since no additional space is required beyond the original size. The disadvantages are that when we find a key that has the same value as one that was removed, it will result in incorrect results. This technique impacts the logic since it creates confusion during insertions and deletions.

Example: Insert 11, 25, 32 in the hash table

Remove 11, the entry for 11 is now -11

Search for 11 terminates when -11 is found.

ii) The advantage of this technique is that searching for keys becomes faster since the table maintains the keys more likely to be found and thus, we have less entries. The disadvantages are that finding a key can be more complicated since the key can be located in a different location than the one it was hashed, the space complexity is bad since more space is needed to store the replaced keys.

Example: Insert 11, 25, 32 in the hash table

Remove 11

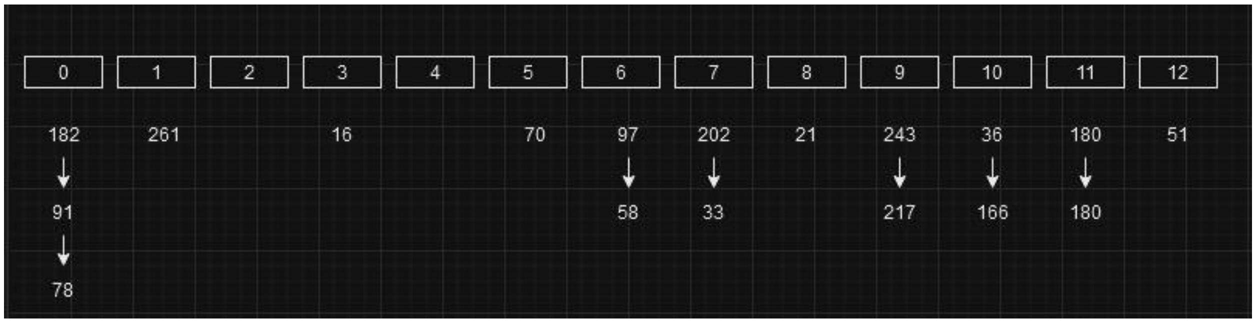
Make a chain by relocating 25 in the slot of 11 and 32 in the slot of 25

Search for 25 finds it very fast in the original hash location.

3.

4.

i)



ii)

The maximum number of collisions caused by the insertions is 3, found at index 0.