1-

A –

Selection sort is not a stable sorting

2a, 2b, 1

After the first iteration we have:

1, 2b, 2a

Meaning the list is sorted but didn’t kept the initial order of elements with same value.

B-

Class Node:

Def \_\_init\_\_(self,data):

Self.data = data

Self.next = none # -> pointer to the next node

Self.prev = none # -> pointer to the previous node

Def change\_data(self, new\_data)

Def get\_data(self)

Def get\_prev(self, node)

Def get\_next(self, node)

Class LinkedList:

Def \_\_ini\_\_(self):

Self.head = none # -> pointer to the first element of the LinkedList

Self.tail = none # -> pointer to the last element of the LinkedList

Def add\_node(self, node) # -> add Node to the end of the list

Def remove\_node(self,node) # -> remove Node

Def get\_node(self,node) -> return None

Def get\_head(self)

Def get\_tail(self)

C –

Yes it’s possible. If n is 1 or 2. Meaning there’s only one root without children, therefore no leafs, or with one children, therefore one leaf.

Log1 = 0 – one element, no leafs

Log2 = 1 – two elements, one leaf

For any other input this affirmation is not right, since we have that a Heap is a complete binary tree and therefore we have that half of the elements are leafs.

D –

To insert an element in an open addressing hashing with linear probing, it takes O(1) in the best case and O(n) in the worst case. The best case is when we call the hash function for a key and find an empty slot at first try, and the worst case is when we have a linear clustering in the array and have to go through all the slots.

Θ (1/1- α) for α = load factor

E –

10000 = Teta(logn)

n^5 + n^2 = O(0,1n^5)

2^n = O(n)

2-

A –

20

15 25

10 18 x 30

x x 16 19 x x

B –

18

15 20

10 16 19 25

3 –

We can return the arr in a reverse order, meaning the first element will always be the biggest from the array, and the first element is always the root of the MaxHeap and the element of preference in the heap.

This will have a complexity time of O(n).