שאלה 1:

Given a stack with a limit of k elements, where each k operations that is a copy operation that take O(I) where I is the number of elements in the stack.

Operation running time:

Here we wrote the running time for each operation.

Let’s use amortize analysis to solve this.

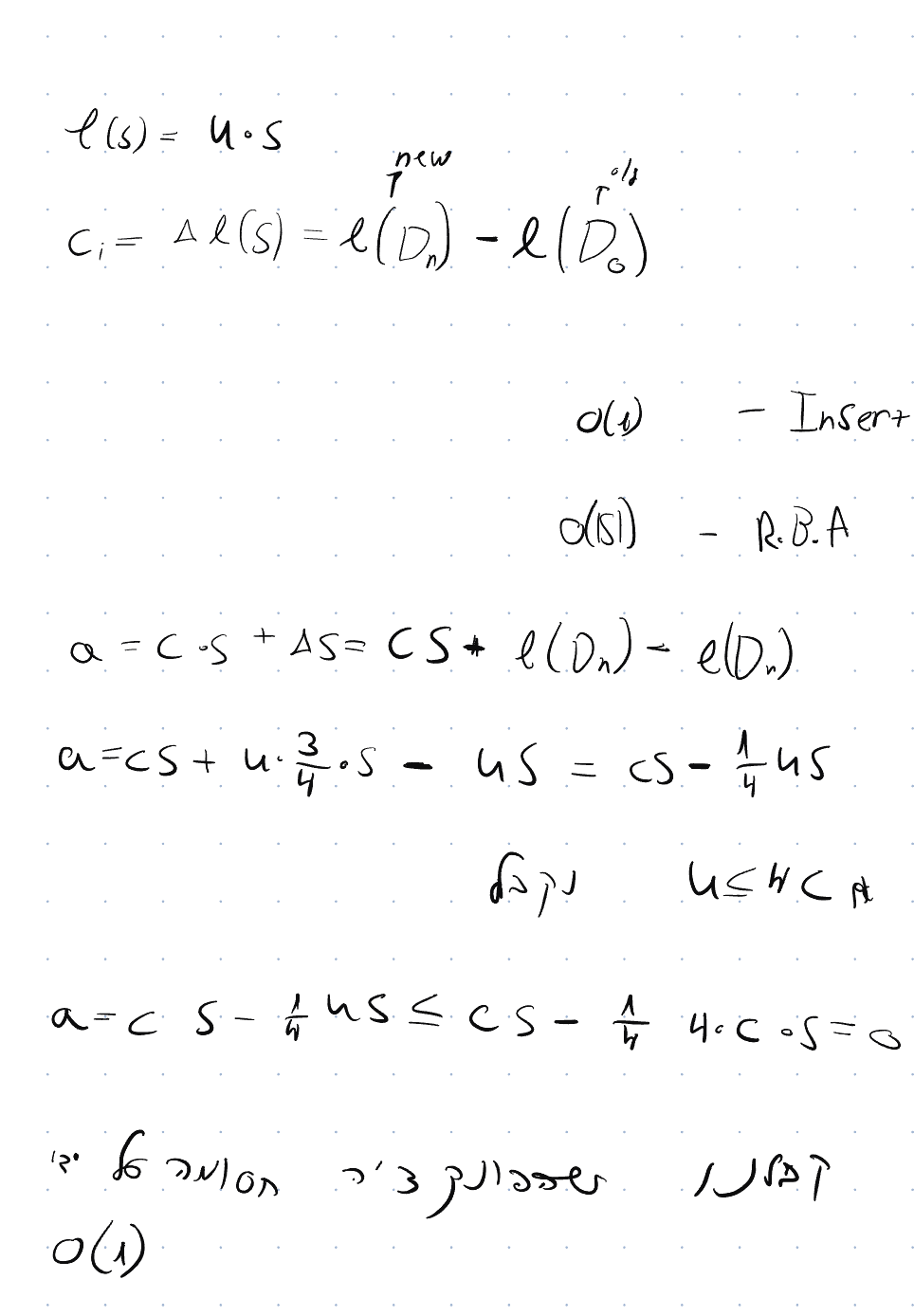
Firstly, we need to understand that the copy running time is bounded by

Because we are given a stack that can’t exceed k elements.

Copy operations

Running time analysis

שאלה 2:



שאלה 3:

כאשר יש לנו עץ 2-3 הגובה יכול לנוע בין

ב:

להבנתי מדובר על פעולה של search, אפשר לבצע פעולה זו בצורה יחסית מהירה

על ידי האלגוריתם הבא

A screenshot of a computer code

Description automatically generated

ג:



שאלה 4:

We will describe the following algorithm:

First, we will assert W.L.O.G that h1 > h2 and T1 >T2 the algorithm will be very similar for T2<T1.

We will move h2 steps on T1, so we get loc = T1.moveleft(h2) == O(h2)

We will append T2 root to loc == O(1)

If loc.count == 4: O(1)

CreateNewParent(loc) == O(h1)

Else if loc.count == 3: O(1)

FixErrorFrom(loc) == O(h1)

DONE

Function CreateNewParent(loc): O(H)

P = Take the loc[2] and split the items

P.parent.push(P)

If P.silblings.len > MAX:

FixErrorFrom(P)

Return P

Function FixErrorFrom(loc): O(H)

P = Take the middle child and push it to the parent

If P.siblings > MAX:

FixErrorFrom(Loc)