1 שאלה

```
public class UnionList implements ListInterface {

    private static final int SUB_LISTS=3; //
    private static final int OWN= 2;//
    private ListInterface[] _lists; //

    public UnionList(ListInterface x, ListInterface y) {
        validate(x); validate(y); //
        _lists = new ListInterface[SUB_LISTS];
        _lists[0] = x;
        _lists[1] = y;
        _lists[OWN] = new List(); //
    }

    public void add(Object x) throws IllegalArgumentException {

        _lists[OWN].add(x);
    }
```

throws IllegalArgumentException;

המשך השאלה

```java
public int size() {

    int totalSize=0;

    for (int i=0; i<SUB_LISTS; ++i)
        totalSize += _lists[i].size();

    return totalSize;
}

// בודקת את הפרמטר אינו null
private void validate(Object o) throws IllegalArgumentException
{   if (o== null)
    throw new IllegalArgumentException();
}

public boolean contains(Object x) {
    for (int i=0; i< SUB_LISTS; i++){
        if (_lists[i]. contains(x))
            return true;
    } //else...
    return false;
}
```

```
public IteratorInterface iterator() {
    return new UnionIterator();
}
                    // מחלקה פנימית רשימה
class UnionIterator() ~~extends~~ implements IteratorInterface {

    private int _list; // אינדקס הרשימה הנוכחית
    private IteratorInterface _it; // איטרטור רשימה
                                   // הנוכחית
    ~~public boolean hasNex~~
```

צריך לקבל את _lists כפרמטר [ולשמור אותו בתור שדה]. −2

```
    protected UnionIterator() {
        _list = 0;
        _it = _lists[0].iterator();
    }

    public boolean hasNext() {
        // תוך מעבר על כל האיברים שלה, מתודה
        // זו גם מעדכנת את המצביע _it לרשימה הבאה לא הנוכחית הסופית //
        // אם אנחנו ברשימה שלא האחרונה //
        while (_list < SUB_Lists -1) {
            if (~~_lists[_list]~~._it.hasNext())
                return true;
            // else...
            _list++;
            _it = _lists[_list].iterator();
        }
        return _it.hasNext(); // רשימה אחרונה:
                              // אם היא ריקה, נחזיר
                              // false
    }
```

הקוד הזה אמור להיות של next [ולא hasNext]. −3

המשך ✓

```java
public Object next() throws NoSuchElementException{

        if (!hasNext())
            throw new NoSuchElementException();
            // אחרי hasNext() אפשר להיות בטוחים
         // שהמצביע _it מצביע לiterator הנכון

        return _it.next();

    }

  }//end of UnionIterator
} //end of Unionlist
```

שאלה 2.

```
class PinhasHelper {
    // n - מספר הכדים
    // k - מספר הכדורים

public static int howManyOptions (int n, int k) {

    if (n<0 || k<0)
        throw new IllegalArgumentException();
    if (n==0)                       // אם אין כד = אין אופציות
        return 0;
    if (k==0)                       // אין כדורים - אופציה אחת
        return 1;
    if (n==1)
        return 1;                   // כד 1 - אופציה אחת
    int sum = 0;
                // i גודל כמה כדורים יש בתוך הנוכחי (האופציות)
    for (int i=0; i<=k; ++i)
        sum += howManyOptions(n-1, k-i);
                                          (<=i)
    return sum;

    }
```

מיותר (שגוי)

יפה.

פתרון רקורסיבי. ע"י ~~...~~ סכום האופציות לשים את הכדורים שנותרו בכל אחד מהכדים.

```
} // end of class
```

33/33

```
Class Balancer {

private static int getRightSpan(Node root){
    int length=0;
    while (root!=null){
        root=root.right;
        length++;
    }
    return length;
}

static public Node balance (Node root){
    return do_balance(root, getRightSpan(root));
    if (root==null)
        return;
    return do_balance(root, getRightSpan(root));
```

```
private static Node do_balance(Node lo, int len){
    if (len==0)
        return null;
    if (len ==1) {
        lo.right=null;
        lo.left = null;
        return lo;
    }
    Node middle = lo;
    for (int i =0; i< len/2; ++i)
        middle = middle.right;
    middle.left = do_balance(lo, len/2);
    middle.right= do_balance(middle.right, len/2);
    return middle;
}
} // end of class
```

ע ג.

כשקיבלנו מבצעים סריקה של העץ בצורה בינארית

O(n) (הערה: עושים כאן כמו בשיטה הקודמת).

לאחר מכן, אנו מבצעים יצור כל קודקוד שאינו עומד בדרך הנכון בצורה

i ד-ב הפעולות (כמו ב-rebalance).

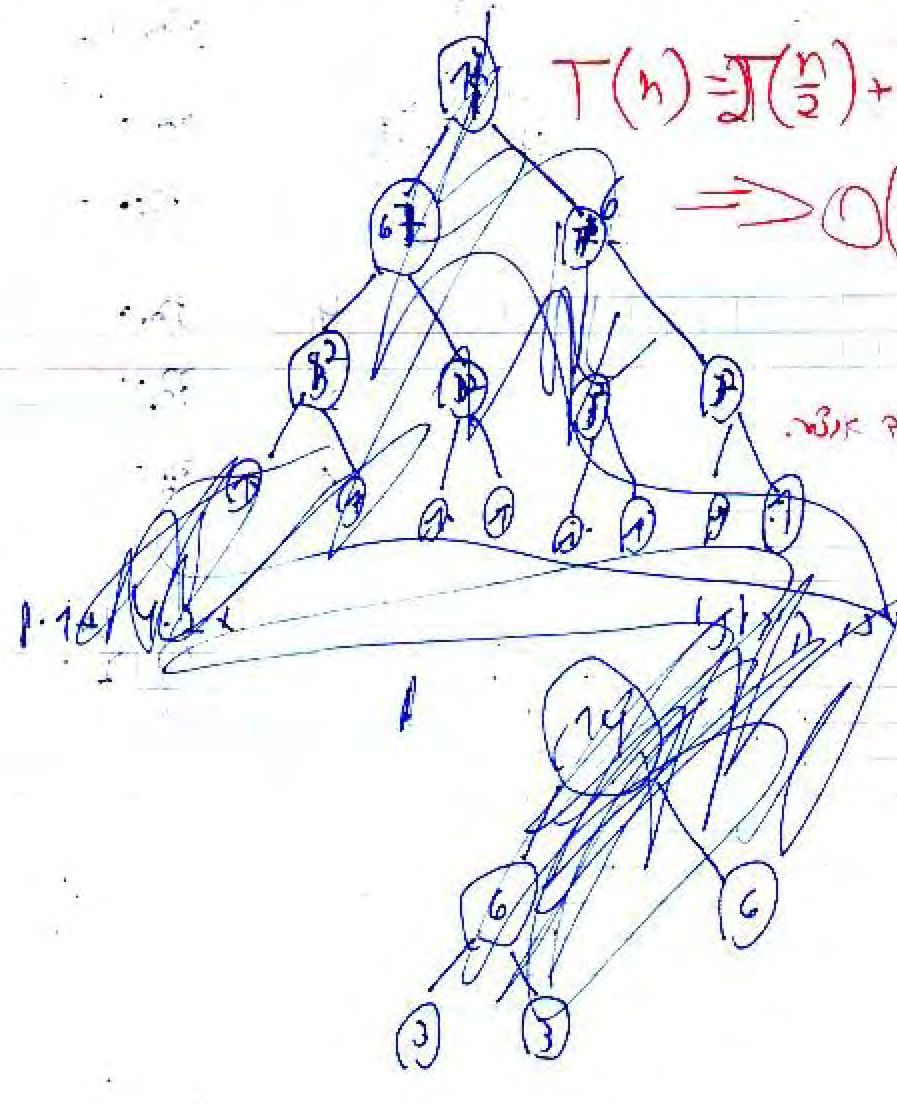
~~כדי ... (כמו ...) ... אידיאלי (כאשר כמו)~~

~~כאשר כמות ... של הפעולות הרבה ... (לא ...)~~

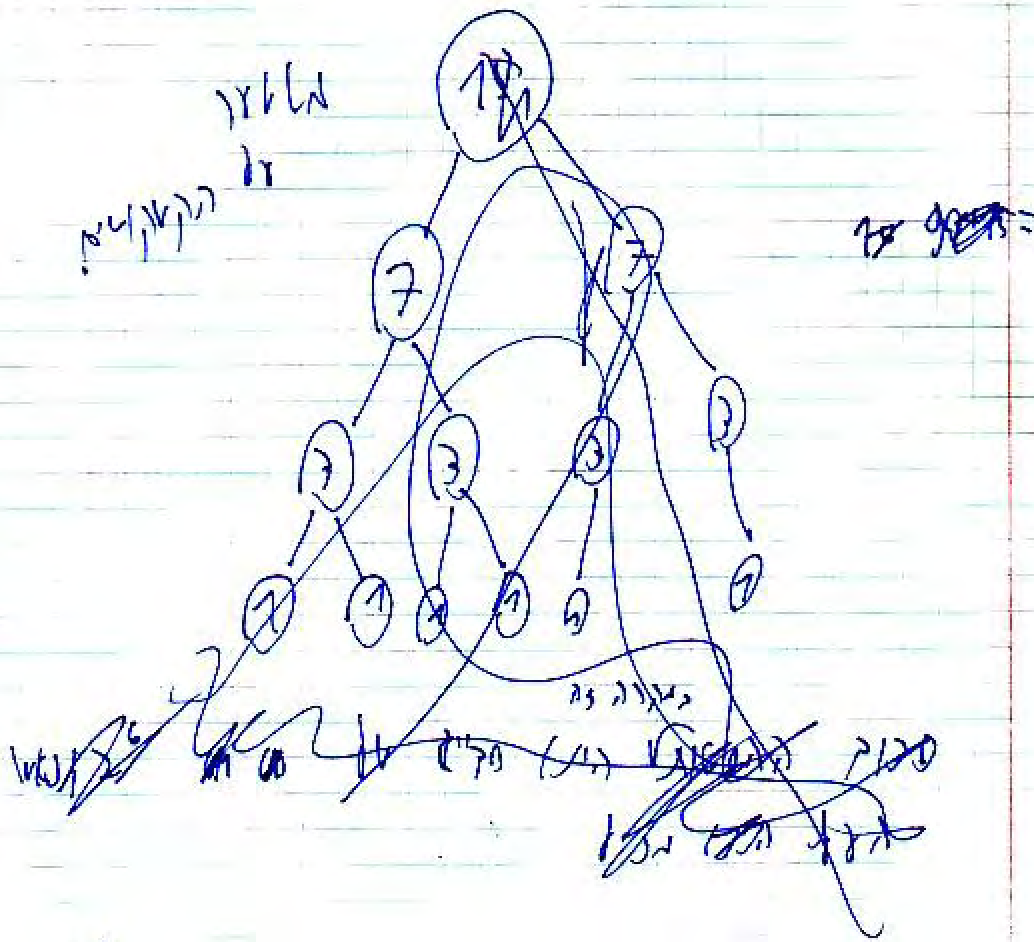
-5

$T(n) = 2T(\frac{n}{2}) + O(n)$

$\Rightarrow O(n \log n)$



חיפוש קודם אנטר.



ניתן לראות שבמקרה הגרוע איננו מבצעים בהכללות
יותר מ- $n\log_2 n$ פעולות השוואה והעתקה,
~~[illegible]~~ שהסדר גודל הוא $O(n\log_2 n)$. (n הכמות
של המספרים, שהפעולות שבין שני מספרים נחשב ליחידה)