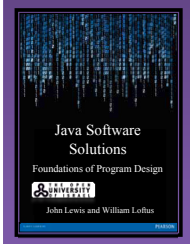
מבוא למדעי המחשב ושפת Java

11.5: מיון-מיזוג ברשימות

מרצה: תמר וילנר



מיון-מיזוג Merge-Sort

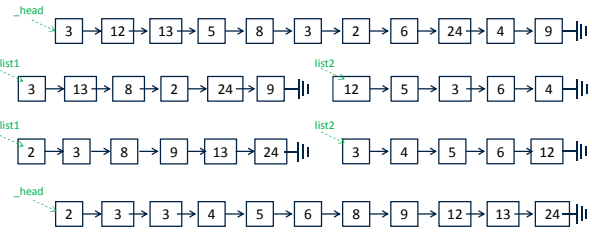
אלגוריתם שמתאים במיוחד לרשימות, למרות שאפשר ליישם אותו גם על מערכים.

עובד בשיטת "הפרד ומשול".

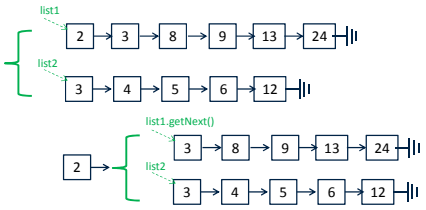
האלגוריתם:

1. חלק את הרשימה לשתי רשימות שוות (עד כדי 1)
2. מיין באמצעות מיון-מיזוג כל אחד מחצאי הרשימות
3. מזג את שני החצאים הממוינים לרשימה אחת ממוינת.

מיון-מיזוג



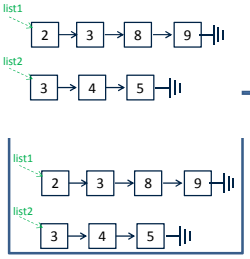
שיטה למיזוג שתי רשימות ממוינות merge



Merge – הקוד:

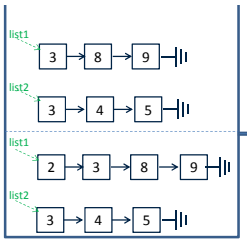
```
private IntNode merge(IntNode list1, IntNode list2) {
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
       list1.setNext(merge (list1.getNext(), list2));
       return list1;
   } // end if
   else {
       list2.setNext(merge (list1, list2.getNext()));
       return list2;
   } // end else
}
```


מיזוג שתי רשימות ממוינות - סימולציה



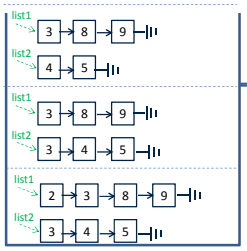

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
       list1.setNext(merge (list1.getNext(), list2));
       return list1;
   } // end if
   else {
       list2.setNext(merge (list1, list2.getNext()));
       return list2;
   } // end else
}
```

מיזוג שתי רשימות ממוינות - סימולציה




```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```


מיזוג שתי רשימות ממוינות - סימולציה




```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```


מיזוג שתי רשימות ממוינות - סימולציה

list1 = (8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (3,4,5)
list1 = (2,3,8,9)	list2 = (3,4,5)

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```


מיזוג שתי רשימות ממוינות - סימולציה

list1 = (8,9)	list2 = (5)
list1 = (8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (3,4,5)
list1 = (2,3,8,9)	list2 = (3,4,5)

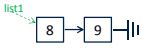
```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```


מיזוג שתי רשימות ממוינות - סימולציה

list1 = (8,9)	list2 = (null)
list1 = (8,9)	list2 = (5)
list1 = (8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (3,4,5)
list1 = (2,3,8,9)	list2 = (3,4,5)

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

מוחזר:

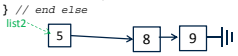


מיזוג שתי רשימות ממוינות - סימולציה

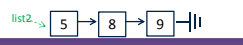
list1 = (8,9)	list2 = (5)
list1 = (8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (4,5)
list1 = (3,8,9)	list2 = (3,4,5)
list1 = (2,3,8,9)	list2 = (3,4,5)

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

מה שחזר:



מוחזר:



מיזוג שתי רשימות ממוינות - סימולציה

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

list1 = (8,9) list2 = (4,5)

list1 = (3,8,9) list2 = (4,5)

list1 = (3,8,9) list2 = (3,4,5)

list1 = (2,3,8,9) list2 = (3,4,5)

מה שחזר:

מיזוג שתי רשימות ממוינות - סימולציה

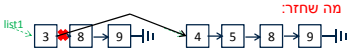
```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

list1 = (3,8,9) list2 = (4,5)

list1 = (3,8,9) list2 = (3,4,5)

list1 = (2,3,8,9) list2 = (3,4,5)

מה שחזר:



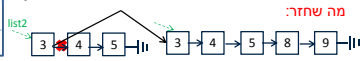
מיזוג שתי רשימות ממוינות - סימולציה

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

list1 = (3,8,9) list2 = (3,4,5)

list1 = (2,3,8,9) list2 = (3,4,5)

מה שחזר:

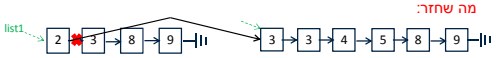


מיזוג שתי רשימות ממוינות - סימולציה

```
private IntNode merge(IntNode list1, IntNode list2)
{
   if (list1 == null)        return list2;
   if (list2 == null)        return list1;

   if (list1.getValue() < list2.getValue()) {
      list1.setNext(merge (list1.getNext(), list2));
      return list1;
   } // end if
   else {
      list2.setNext(merge (list1, list2.getNext()));
      return list2;
   } // end else
}
```

list1 = (2,3,8,9) list2 = (3,4,5)

מה שחזר:



מיזוג שתי רשימות ממוינות - סימולציה

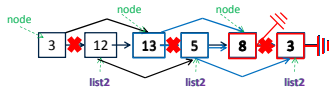
בסופו של התהליך מוחזרת הרשימה

חלוקת רשימה לשניים - Split

```
private IntNode split(IntNode node)
{
    if (node == null || node.getNext() == null)
        return null;
    IntNode list2 = node.getNext();
    node.setNext(list2.getNext());
    list2.setNext(split(list2.getNext()));
    return list2;
}
```

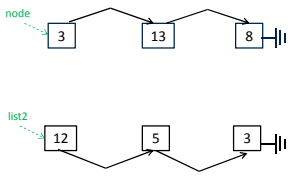
חלוקת רשימה לשתיים - סימולציה

```
private IntNode split(IntNode node)
{
    if (node == null || node.getNext() == null)
        return null;
    IntNode list2 = node.getNext();
    node.setNext(list2.getNext());
    list2.setNext(split(list2.getNext()));
    return list2;
}
```



חלוקת רשימה לשתיים - סימולציה

בסופו של התהליך יש לנו שני חצאי רשימות:



השיטה הפרטית mergeSort

```
private IntNode mergeSort (IntNode node)
{
    if (node == null || node.getNext() == null)
        return node;   // checks for empty or single list

    IntNode list2 = split (node);

    node = mergeSort (node);
    list2 = mergeSort (list2);

    return merge (node, list2);
}    // end merge_sort
```


השיטה הציבורית mergeSort

השיטה הציבורית במחלקה IntList:

```
public void mergeSort ()
{
    _head = mergeSort(_head);
}
```

השיטה הציבורית קוראת לשיטה הפרטית. היא מבצעת את המיון ולא מחזירה כלום. ראש הרשימה - _head יצביע בסוף התהליך על ראש הרשימה הממוינת.

מה הסיבוכיות של mergeSort

סיבוכיות זמן:

השיטה split שמחלקת את הרשימה לשני חצאים – $O(n)$

השיטה merge שממזגת שתי רשימות לאחת ממוינת – $O(n)$

כמה רמות כאלו יש לנו? $\log_2 n$

לכן בסה"כ סיבוכיות הזמן של mergeSort היא $O(n \log_2 n)$

תמיד!

לא רק במקרה הממוצע. גם במקרה הגרוע ביותר.

סיבוכיות מקום:

קבועה (בלי להתייחס למחסנית הרקורסיה)