Algorithms 2024/25: Tick 1

A collection of *r* sorted, singly-linked lists containing a total of *n* numbers have been joined end-to-end to form a single list of *n* items. Your task is to construct a single, sorted linked list from all the elements, subject to the requirements below.

Notes:

1. The data is presented as a linked list, not an array.
2. The lengths of each of the *r* lists are unknown (and not necessarily all the same).
3. The range of distinct values among the n numbers is unknown.
4. Your program is only given a pointer to the head of the list; you are not given the values of *n* or *r* as inputs, and there are no markers indicating where the *r* sorted files begin. (This does not preclude your algorithm from determining those things, if possible and desired, but doing so counts towards the running time.)

Propose three algorithms to accomplish the task, subject to the following constraints:

1. At least one of your algorithms should only require constant additional memory (i.e. have O(1) space complexity).
2. At least one of your algorithms should have best and worst case running times proportional to $n \lg n$ (it is permitted to also be proportional to some function of *r*, e.g. $O(nr^2 \lg n)$).
3. At least one of your algorithms should have a running time that is independent of *r*.
4. At least one of your algorithms should rely on a heap and use $O(r)$ additional memory.

(At least one algorithm must meet at least two constraints, by the pigeonhole principle!)

Implement your heap-based algorithm in Java (without using anything in the java.util package, nor any other libraries). Your implementation should be contained in a single Java class which implements the interface below: the *sort()* method is passed a pointer to the head of the linked list to be sorted and should return a pointer to the head of your sorted output list. (You may allocate new ListCell objects but a better solution simply reorders the ListCells in the input list.)

What to submit:

1. Three descriptions of your algorithms. Simple algorithms require a single sentence; more elaborate ideas should be explained sufficiently for someone else in the Part IA cohort to understand the method.
2. Your Java implementation: a single Java source containing a class that implements the interface provided. Helper functions are allowed and encouraged to make your code readable! You may write unit tests and a class containing a main(String[]) function in other files but are not asked to submit those.

Submit >> here << You may re-submit if you wish. **The deadline is 23:59 on Fri 21 Feb 2025**.

```
interface Algs202425Tick1 {
  class ListCell {
    public int val;       // The number stored in this cell.
    public ListCell nxt;  // Reference to the next cell in the list.
  }

  ListCell sort(ListCell head);
}
```

Save this ^^ into Algs202425Tick1.java, then write your solution in Solution.java:

```
public class Solution implements Algs202425Tick1 {...}
```

Dr John Fawcett, Feb 2025