Assignment No. 4: Merge k Ordered Lists Efficiently

Allocated time: 2 hours

Implementation

You are required to implement **correctly** and **efficiently** an O(n*log*k) method for **merging k sorted sequences**, where n is the total number of elements. (Hint: use a heap, see seminar no. 2 notes).

Implementation requirements:

- Use linked lists to represent the k sorted sequences and the output sequence

Input: k ordered lists of numbers $< a_1^i, a_2^i, \dots, a_{m_i}^i >$, $\sum_{i=1}^{k} m_i = n$

Output: a permutation of the union of the input sequences: $< a_1' \leq a_2' \leq \cdots \leq a_n' >$

Evaluation

! Before you start to work on the algorithm evaluation code, make sure you have a correct algorithm! You will have to show your algorithm works on a small-sized input (e.g. k=4, n=20).

We will make the average case analysis of the algorithm. Remember that, in the average case, you have to repeat the measurements several times. Since both **k** and **n** may vary, we will make each analysis in turn:

1. Choose, in turn, 3 constant values for k (k1=5, k2=10, k3=100); generate k **random** sorted lists for each value of k so that the sum of elements in all the lists n varies between 100 and 10000, with a maximum increment of 400 (we suggest 100); you may split the n elements equally between the k lists; run the algorithm for all values of n (for each value of k); generate a chart that represents the sum of assignments and comparisons done by the merging algorithm for each value of k as a curve (total 3 curves).

2. Set n = 10.000; the value of k must vary between 10 and 500 with an increment of 10; generate k **random** sorted lists for each value of k so that the sum of elements in all the lists is 10000; you may split the n elements equally between the k lists; test the merging algorithm for each value of k and generate a chart that represents the sum of assignments and comparisons as a curve.

3. Interpret your charts.