Lab Discussion 3

EXERCISE 1. Explore the various sorting algorithms on ALIST and DLIST defined in the previous labs:

(i) Run the algorithms on different lists and measure performance (to see a difference this only makes sense for very large lists).

(ii) Extend the classes by definitions of *quicksort* and *radixsort* algorithms.

(iii) Go into details of the complexity of selection and insertion sort. Determine a precise number of operations for the two algorithms, not just the O-class.

EXERCISE 2. Continue the exercise with class RLIST with a new *equi-join* function.

(i) For two records $r_1 = (a_1 : v_1, \ldots, a_n : v_n)$ and $r_2 = (b_1 : u_1, \ldots, b_m : u_m)$ provide pairwise different indices $\{i_1, \ldots, i_k\}$ and $\{j_1, \ldots, j_k\}$ for some $k \leq \min(n, m)$. The records r_1 and r_2 can be *joined* iff $a_{i_x} = b_{j_x}$ holds for all $1 \leq x \leq k$. In this case the *equi-join* $r_1 \bowtie r_2$ is the combined record of r_1 and r_2 with $n + m$ attributes.

Extend the equi-join to two structured lists resulting in the list of all joins of records r_1, r_2 from the two lists.

(ii) Analyse the complexity of the *equi-join* function and discuss how sorting can be used to improve complexity.

(iii) Implement sorting on structured lists, where the records have a key attribute. Order list elements according to a total order on the key values.

(iv) Implement the *equi-join* function exploiting the sorting function developed in (iii).

EXERCISE 3.

(i) Explore a list-of-blocks representation for sequences, a simple data structure for sequences that combines the advantages of linked lists and unbounded arrays and is more space-efficient. Use a linked list, where each item stores an array of K elements for some large constant K.

(ii) Implement such a data structure.

(iii) Implement merge and mergesort for this data structure. During merge reuse emptied input blocks for the output sequence.

(iv) Compare the time complexity of mergesort for this data structure eith linked lists and unbounded arrays. Pay attention to constant factors.

EXERCISE 4. Reverse the order of elements on a stack S

(i) using two additional stacks;

(ii) using one additional queue;

(iii) using one additional stack and some additional non-array variables.