Advanced Data Structures

Last labwork

December 20, 2018

Exercises related to mergeable heaps

1. Consider a simply linked-list of nodes with the following structure (the nodes are linked via the `sibling` pointers)

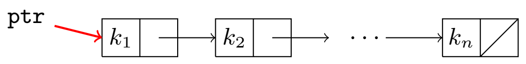
   ```
   struct Node {
      int key;
      Node *sibling;
   }
   ```

   Write down a program that performs the following operations:

   - It reads from the console a line of n integers separated by spaces

     $$k_1\ k_2\ \dots\ k_n$$

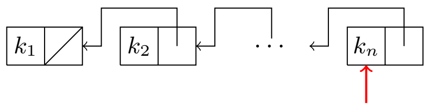
     and creates a pointer `ptr` to the linked list with nodes containing the keys $k_1, \dots, k_n$, in this order:

   - calls the function

     `Node* reverseList(Node *ptr);`

     that reverses te list `ptr` (by making the links to point in the opposite direction), and returns a pointer to its first element.

     (Note: You should implement `reverseList`)

   - Displays the keys of the nodes in the inverted list,by traversing the nodes from head to tail.

2. You can download from the webpage of this lecture

   `http://staff.fmi.uvt.ro/~mircea.marin/lectures/ADS/binoheap.zip`

   an incomplete implementation of binomial heaps. Complete the implementation with the implementation of the capability to extract the node with minimum key from a binomial heap. This amounts to implementing the following functions:

   - `Node* reverseList(Node* l)`
     which should behave the same as the function implemented in the previous exercise.
   - `Node* findMinRoot(Node* l)`
     should return a pointer to the node with minimum key from the linked list of nodes pointed to by `l`. If `l` is the null pointer, the function should return the null pointer.