Advanced Data Structures

Labwork 1: Binomial Heaps

Labwork 1

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

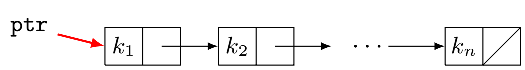
```
struct Node {
   int key;
   Node *sibling;
}
```

Write down a program that performs the following operations:

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

  $$k_1\ k_2\ \ldots\ k_n$$

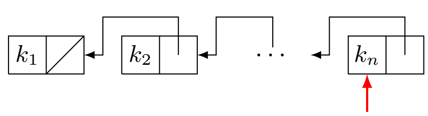
  and creates a pointer `ptr` to the linked list with nodes containing the keys $k_1, \ldots, k_n$, in this order:

- 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 the node with key k_n.

  (Note: You should implement `reverseList`)

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

Labwork 2

The archive `binoheap.zip` contains an incomplete implementation of binomial heaps in `C++`. 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.

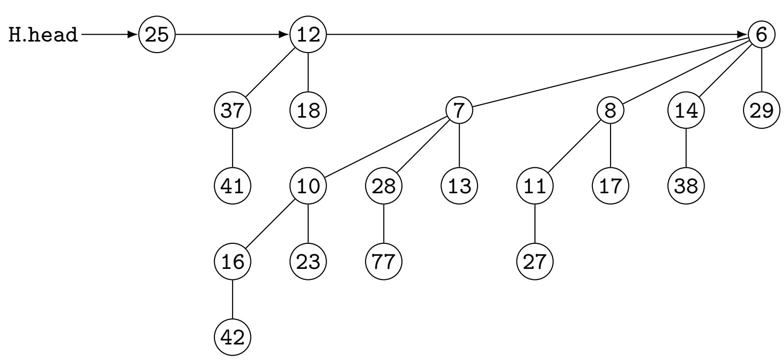
Exercises

1. Suppose that x is a node in a binomial tree within a binomial heap, and assume that $x \rightarrow \texttt{sibling} \neq \texttt{NIL}$.

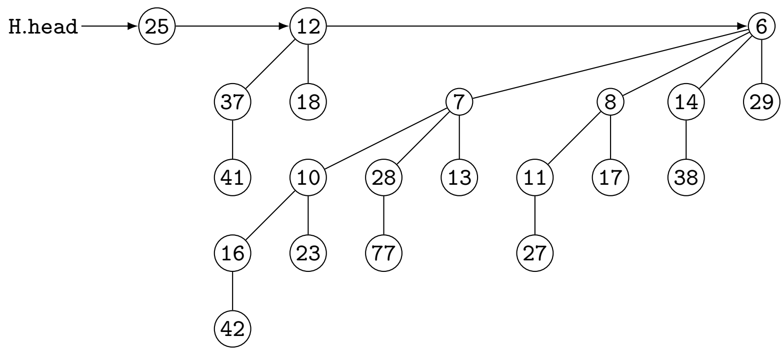
   (a) If x is not a root, how does $x \rightarrow \texttt{sibling} \rightarrow \texttt{degree}$ compare to $x \rightarrow \texttt{degree}$?

   (b) If x is a root, how does $x \rightarrow \texttt{sibling} \rightarrow \texttt{degree}$ compare to $x \rightarrow \texttt{degree}$?

2. Show the binomial heap that results when node with key `24` is inserted into the binomial heap shown below:



3. Show the binomial heap that results when the node with key 28 is deleted from the binomial heap shown below:

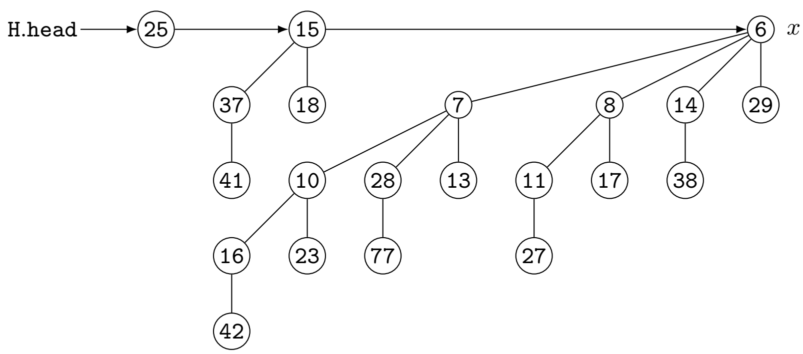


4. Suppose H is a binomial heap implemented as described in the lecture notes. Write the pseudocode for the operation

   `increaseKey(H,` x, k`)`

   which takes as inputs a pointer to a node x in H with $x \to$ `key` $< k$ and increases the key of x to new value k.

   (a) Draw the binomial heap that results after increasing the key of node x in the heap depicted below to new value 12.

   (b) What is the worst runtime complexity of this operation?

   (c) Indicate a binomial heap H with 16 nodes, a node x of H, and a value k such that the operation

   `increaseKey(H,` x, k`)`

   takes the longest possible time.