Laboration 1

Due 2024-11-21

The task is to implement (using Python) and evaluate algorithms for sorting a stack.

- **Part I**

  Implement a stack with push and pop operations using instances of linked list data structure:

  - *push*: Add an element to the top of the stack.
  - *pop*: Remove and return the top element from the stack.
  - *isEmpty*: Return True if the stack is empty, False otherwise.

- **Part II**

  Given a stack of n elements and an empty stack, the goal is to sort these elements, by returning a stack containing the elements in sorted order (with the smallest element on top). We are allowed to use only these two stacks (with the implementation in Part I) together with a constant number of additional memory cells/registers. Moreover, the operations allowed are push, pop, isEmpty, comparison between two input elements, read, and write.

  - Describe your algorithm in text (step-wise);
  - Calculate the *exact* number of operations used by your algorithm in the worst case (*do not use asymptotic notations*); and
  - Implement your algorithm in *Python*. Built-in functions or methods for stacks must not be used.
  - Investigate the performance of your algorithm by testing your code on *large* data sets. The time complexity of the algorithm can be measured by both the number of main operations (such as push, pop, and comparison) performed and the total running time (*excluding the one needed to generate the test data*) taken by the algorithm. The algorithmic efficiency testing *may* be carried out by conducting simulation experiments with various input settings (for example, random data, already sorted data, almost sorted data, ...).

- **Submission**

Each group must turn in *one* report describing and illustrating above algorithm designs and experimental results. The report can be written in either Swedish or English and should not be handwritten.

Before submitting your report, you should discuss your solution to the laboration (design, implementation, test, and report) with your lab-assistants.