Data Structures
Homework #3

Due: Nov 10, 2020

1. Let x and y be real number with $0 < x < y$. Prove that n^x is $O(n^y)$, but n^y is **NOT** $O(n^x)$.

2. Describe, step by step, the output for the following sequence of deque ADT operations: `addFirst`(3), `addLast`(8), `addLast`(9), `addFirst`(5), `removeFirst`(), `removeLast`(), `first`(), `addLast`(7), `removeFirst`(), `last`(), `removeLast`().

3. There is a simple, but inefficient, algorithm, called *bubble-sort*, for sorting a sequence S of n comparable elements. This algorithm scans the sequence $n-1$ times, where, in each scan, the algorithm compares the current element with the next one and swaps them if they are out of order. Give a pseudo-code description of bubble-sort that is as efficient as possible assuming S is implemented with a doubly linked list. What is the running time of this algorithm (using the Big-Oh notation)?
   hint: Avoid index-based operations.

4. Suppose that you are given an $m \times n$ matrix A. Now you are asked to check if matrix A has an entry $A[i][j]$, which is the smallest value in row i and the largest value in column j. For example, the following matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ has such an entry at $A[3][1]$ with value 7 and the matrix $\begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 6 \\ 11 & 10 & 5 \end{pmatrix}$ has no such an entry. Please have a solution to determine the location of such an entry if one exists in the given matrix. What is the running time of your method.

5. Suppose we are given an integer N and would like to have all the binary numbers smaller than or equal to N. In order to have the results generated, one can use a ***queue*** to help. Please give an approach that uses a ***queue*** to generate the results, write your approach with pseudo-code, and discuss the time as well as space complexity for your approach. Please note that the output should have the binary numbers in increasing order.

```
Input: 10
Output: ['1', '10', '11', '100', '101', '110', '111',
        '1000', '1001', '1010']
Explanation: 10 binary numbers are generated sequentially

Input: 13
Output: ['1', '10', '11', '100', '101', '110', '111',
        '1000', '1001', '1010', '1011', '1100', '1101']
Explanation: 13 binary numbers are generated sequentially
```

6. (**Programming problem 1**)
Recall Problem **??**. Please use Python to implement the pseudo-code you provide for solving the problem with the approach, named as `generateBinaryNumbers()`. Please note that your program *must use a* ***queue*** as an auxiliary tool. When you submit your homework, you need to indicate where the queue is in your program and how it helps. Hence, you need to

   (1) implement the approach as a function named as `generateBinaryNumbers()`,

   (2) implement a **queue** data structure to manage the items used for the approach, and

   (3) give a short statement for indicating where and how the queue is used in your program and discussing the time and space complexity.

7. (**Programming problem 2**)
Given two ***sorted*** **linked lists**, merge them in place (on the given linked lists) without using extra space (only one or two additional space allowed) and without modifying links of the first linked list. The solution should preserve the sorted order of elements in both lists. More precisely, if m and n are the number of nodes in the first and second linked lists respectively, then the first m smallest nodes in both lists combined should become in the resulting first sorted linked list. The remaining nodes will go to the resulting second sorted linked list.
Examples:

```
First List : 2, 6, 9, 10, 15
Second List: 1, 4, 5, 20
Output:
First List : 1, 2, 4, 5, 6
Second List: 9, 10, 15, 20
```

   A simple solution is to use the idea of merge procedure in *merge-sort* algorithm. After merging both lists, we assign the first m smallest nodes to the first linked list and the rest n nodes to the second linked list, where m and n are the numbers of nodes in the first and second linked lists respectively. This can be done in $O(m + n)$ time and constant space.

   The above solution violates the problem constraints that modifying the links of the first list is not allowed and may use linear space for an additional linked list using for the interim resulting linked list. Please provide a program to merge the two given sorted linked lists in place according to the problem constraints and discuss the time and space complexity. Things you need to provide include:

   (1) Since the input is Python lists of integers, you need to have the function, `readInputListintoLinkedList()`, to read the integers and generate the corresponding linked lists.

   (2) The classes `Node` and `linkedList` should be defined and implemented with methods.

   (3) The functions for the merging process, `mergeLinkedLists()`.

   (4) A short statement for discussing the time and space complexity.

About submitting this homework

1. For problem 1, 2, 3, 4 and 5, Please

   (1) write all of your solutions on the papers of size A4,

   (2) leave you name and student ID on the first page, and

   (3) hand in your solutions for problem 1, 2, 3, 4 and 5 to me in class

2. For problem 6 and 7, things to be submitted include:

   (1) please finish each problem right after the problem description in the `F20u-HW3prog.ipynb` file provided on the **i-school(Plus)** (`https://istudy.ntut.edu.tw/learn/index.php`) platform; and

   (2) please upload the completed `.ipynb` file with the filename as `HW3_studentID.ipynb` to **i-school(Plus)**

3. **Late work** is not acceptable. Remember, the **deadline** is the midnight of **Nov 10**, 2020.

4. Honest Policy: We encourage students to discuss their work with the peer. However, each student should write the program or the problem solutions on her/his own. Those who copy others work will get 0 on the homework grade.