Weekly Assignment 2

10th September 2019

Deadline: 17th September 2019, 3.30pm.

Exercise 1.

Write an algorithm to find the middle element of a given linked list. Your algorithm should determine the middle element in single iteration and take care of both of the following cases:

- **Case 1:** If the given linked list is $1- > 2- > 3- > 4- > 5$ then the output should be 3.
- **Case 2:** If the list has an even number of elements and there are two middle elements, then the output should be the second one. For example, if the input is $1- > 2- > 3- > 4- > 5- > 6$ then the output should be 4.

Exercise 2.

Consider the incomplete Algorithm 1 displayed below.

1. Complete the functions and procedures of Algorithm 1.
2. Give the time complexity of each function.
3. What is displayed on the screen?[1]
4. Write a function `empty()` that empties the stack.

Exercise 3.

Let S be a stack of size $n \geq 1$. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that push and pop operation take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation.

[1] recall that `WriteLn(variable)` writes the content of `variable` in the terminal.

Algorithm 1: Incomplete pseudocode for stack as array of strings.

```
variable: stack: array of strings
variable: no_elements: integer
/* creates empty stack                                          */
Procedure createStack()
    no_elements ← 0
    SetLength(stack, no_elements)   // sets size of the array of
     strings "stack" to "no_elements"
/* tests if the stack is empty (may return true of false)       */
Function isEmpty()
    xxx TODO xxx
/* pushes "element" to the stack                                */
Procedure push(element)
    xxx TODO xxx
/* removes and returns last element on stack                    */
Function pop()
    xxx TODO xxx
/* returns top element                                          */
Function top()
    xxx TODO xxx
/* displays elements in the stack                               */
Procedure display()
    xxx TODO xxx
createStack()
push("AAA")
display()                                                  // 1
push("BBB")
push("CCC")
display()                                                  // 2
name ← pop()
WriteLn(name)
display()                                                  // 3
```

For $m \geq 1$, define the stack-life of m as the time elapsed from the end of `Push`(m) to the start of the `Pop`() operation that removes m from S. Find out the average stack-life of an element of this stack.

Exercise 4.

Apply the BFS algorithm to the directed graph displayed in Figure 1, with as source vertex node 1. Specify the content of the queue before/after each iteration of the while loop, the predecessor function π, and the distance function d.

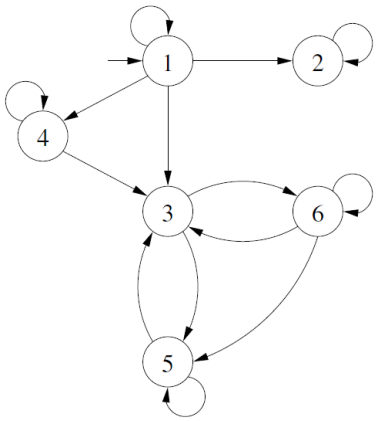

Figure 1: A directed graph.

Exercise 5.

Consider a directed graph G with vertex s. Let C be the set of cycles in G that visit vertex s. Give an algorithm that returns `true` and a cycle from C with minimal length in case C is nonempty, and `false` otherwise. Explain why your algorithm is correct.

Exercise 6.

Give an algorithm that checks if a undirected and connected graph has a cycle. Explain why your algorithm is correct.