1.

* Create a pointer to a child node, *firstNode*, for McCreights algorithm
* Build a suffix tree using McCreights algorithm
* Begin at the leaf, *firstNode*:
* Let edge label be α.
* While current node is not root:
  + Move to the parent node
    - Let the edge label of this parent node be beta
    - If beta + ‘$’ != alpha:
      * Return “not periodic”
* Print string[α]

McCreights algorithm builds the tree in O(n) time.

Traversal up the edge path of leaf node 1 is linear time.

Evaluation of internal nodes is linear time.

Total time complexity = O(n) + 1 + 1 = O(n)

2.

* Let the size of the string be n
* Create a pointer to a node, *firstNode* for McCreights algorithm
* Build a suffix tree using McCreights algorithm
* Move to *firstNode*:
* While current node is not root:
  + Move to the parent node
  + Let the string depth of this internal node be beta
  + For each child of this internal node, find the child with the greatest string depth that is a leaf node and not leaf node 1 (follow all children to all leaves)
    - Let the string depth of this leaf node be alpha
    - If alpha + beta = n:
      * Return beta
  + Move to next parent
* If node is root node, return “no beta detected”

McCreights algorithm builds the tree in O(n) time.

Traversal up the edge path of leaf node 1 is linear time.

Evaluation of children nodes is O(n) time.

Total time complexity = O(n) + 1 + O(n) = O(2n) = O(n)