TD 3 - Lists and Complexity- Master 1 Bio-informatique

December 2017

Exercice 1

Write a Python program to create simple linked lists.

To do that you will implement the following api :

```
def push_front(l, e):
  # Add an element to the beginning of the list.
def first( l ):
  # return the first cell of the list.
def end( l ):
  # return the cell at the end of the list.
def next( l, cell ):
   # return the cell which below this cell or
   #None if it is the cell of the end of the list.

def value( cell ):
   # return the value of the list.
```

Exercice 2

By using the previous implementation of linked list, write the following functions :

```
def push_back(l, e):
  # Add an element to the end of the list.
def insert( l, cell, e ):
  # Create a new cell containing the element e and insert this cell in
  # the list just after the cell \texttt{cell
def copy( l ):
  # return a copy of the list l.
def middle(l):
  # return the element which is in the middle of the list.
def size(l):
  # return the number of elements in the list.
def transfert( l1, l2 ):
  # Move all elements belonging to L2 into L1.
  # L2 is empty at the end of the function.
def search_element(l,e):
  # return the first cell containing e or the cell at the end of the list.
```

Give complexity of each function.

Exercice 3

Do again all the exercises with double linked lists. To do more, you can implement the algorithm of *dancing links* from Knuth : `https://arxiv.org/abs/cs/0011047`.