Algorithms and Data Structures, Academic Year 2016/2017

International Bologna Master in Bioinformatics

October 23, 2017

Please complete the following exercises by applying the concepts that have been illustrated to you during the classes. The score associated with each exercise and the expected time for completion is reported in the first line. Do NOT copy/exchange results (the parameters of each exercise are different).

Exercise 0 (2 points): write your name and surname in the first row of all the sheets you use.

Name:_____ Surname:_____

Exercise 1 (50 points, 60 minutes): While comparing two strings *P* and *T*, an ***inversion* error** occurs whenever two consecutive characters appear in reverse order in the two strings. For example, if *P* = "d a c b" and *T* = "d c a b", one has an inversion error due to "a c" against "c a"; note that in such a case counting for a single inversion error, instead of two substitution errors (or mismatches) leads to a smaller number of overall errors.

One wants to extend the well-known ***k*-*approximate string*-*matching*** problem by adding also inversion errors to the three usual insertion, deletion and substitution errors.

Solve the so extended problem, by modifying first the recurrence relations giving the optimal substructure and then the pseudo-code of the known *k-approximate string-matching* algorithm.

(use additional sheets for this exercise, including the back of this sheet)

Name:_____ Surname:_____

Exercise 2 (20 points, 20 minutes): let g(.) be a constant time function and consider the following pseudo-code where t is a data structure representing a text with an highlighted position (a cursor). Next and prev move the cursor to the right/left. Insert add a character into the text just where the cursor is.

```
f(n,t) {
        for i = 1 to n do
                if g(i) > 0 then t := next(t)
                else if g(i) < 0 then t := prev(t)
                else t := insert(char-of-int(i),t)
}
```

Compute the computational complexity of f(n,t) as a function of n when t is implemented

1. as a pair (A,j) where A is an **array** of characters and j an index into A
2. as a pair (L,j) where L is a **singly-linked list** of characters and j a pointer to a cell of L
3. as a pair (L,j) where L is a **double-linked list** of characters and j a pointer to a cell of L
4. as a **zipper**, i.e. a pair (L1,L2) of singly linked lists of characters where
   - L2 represents the string after the cursor as a list of characters, in normal order
   - L1 represents the string before the cursor as a list of characters in **reversed** order
   - insert adds a character in front of L1
   - next moves the head of L1 to the head of L2; prev does the opposite

   Example: “hello wo|rld” is represented by L1 = [o;w; ;o;l;l;e;h], L2 = [r;l;d]. After one call to prev we have L1 = [w; ;o;l;l;e;h], L2 = [o;r;l;d]

Assuming that a character needs 8 bits of memory and a pointer 64 bits, how many bytes are required to store a text+cursor made of 10 characters when the text is implemented like in 1-4 above?

Solution:

Fill in the table below. Use the back of the sheet to briefly justify the answer.

Implementation	Time complexity of f(n,t)	Size of a 10 chars text+cursor
1. array		
2. singly-linked		
3. double-linked		
4. zipper		

Name:_____ Surname:_____

Exercise 3 (10 points, 15 minutes): Consider the following recursive function.

$G(0) = 0$
$G(n) = G(n-1) + 2n - 1$

1. compute the return value of G on a few inputs and write down the input/output pairs
2. guess a non-recursive definition of G
3. prove by induction on G that your guess is correct

Name:_____ Surname:_____

Exercise 4 (20 points, 25 minutes): Consider the string “p a p p a”. Write (by hand) its corresponding:

- Suffix trie;
- Suffix tree;
- Suffix array;
- Burrows-Wheeler transform;
- LF mapping**.**