# Barem Examen AA

16 Februarie 2021

## Contents

# 1 Algoritmi Nedeterminiști

Scrieți un algoritm nedeterminist polinomial pentru următoarea problemă și calculați complexitatea sa angelică?

Se dau N secvențe ADN (șiruri de caractere compuse din simbolurile 'A', 'C', 'G', 'T'). Se cere să se identifice dacă cele N secvențe au în comun un subșir de dimensiune cel puțin K.

Un subșir al unei secvențe A este o secvență care poate fi obținută din elemente din A (nu neapărat consecutive), fară a schimba ordinea acestora. De exemplu, secvențele '**CGAT**', '**CA**G**A**T**C**' și 'A**CAGT**' au în comun subșirul 'CAT', de dimensiune 3.

## 1.1 Barem

- **(2p)** Generarea soluției.
- **(3p)** Verificarea constrângerilor.
- **(1p)** Identificarea complexitatății.

## 1.2 Rezolvare

```
function SOLVE(Array < String > A, K)
    solution = "";
    for i = 0; i < K; i++ do
        solution += choice({'A', 'C', 'G', 'T'})
    end for
    for i = 0; i < A.size(); i++ do
        current_seq = A[i];
        current_ptr = 0;
        for j = 0; j < current_seq.size(); j++ do
            if current_seq[j] == solution[current_ptr] then
                current_ptr ++;
                if current_ptr == K then
                    break
                end if
            end if
        end for
        if current_ptr != K then
            fail.
        end if
    end for
    success.
end function
```

O parte dintre voi ați recunoscut probabil problema identificării celui mai lung subșir comun. Această problemă are multe aplicații în domenii diverse precum bioinformatică (*planted motif search*), compresia datelor sau compararea fișierelor (e.g. diff, GIT). Din păcate, pentru un număr arbitrar de secvențe primite la intrare, problema este NP-Complete [1].

Soluția nedeterministă propusă generează un string, cu elemente din alfabetul specificat și apoi verifică dacă acest string reprezintă un subșir pentru toate cele N șiruri primite la intrare. Complexitatea sa este O(K + N * S) = O(N * S), unde S este lungimea maximă a unei secvențe din lista A.

# 2 Teorema Master

$T(n) = 3 * T(\frac{n}{2}) + 2 * T(\frac{n}{2}) + 2048 * n^2 - log^2(n)$.

## 2.1 Barem

- **(0.5p)** identificare corectă: $a = 5, b = 2$
- **(1p)** identificare corectă: $f(n) = 2048 * n^2 - log^2(n) = O(n^2)$
- **(1p)** stabilit că există $\epsilon > 0$ a.î: $f(n) = O(n^{log_b(a)-\epsilon}) = O(n^{log_2(5)-\epsilon})$
- **(1.5p)** concluzionat că putem aplica cazul 1 din Teorema Master și trasă concluzia ca $T(n) = \theta(n^{log_b(a)}) = \theta(n^{log_2(5)})$

# 3 Inducție Structurală

**List 1: Constructori**

$[] :\rightarrow ListT$
$(x : xs) : T \rightarrow ListT \rightarrow ListT$

**List 2:** $(++) : ListN \rightarrow ListN \rightarrow ListN$

**APP1**: $[] + +B = B$

**APP2**: $(x : xs) + +B = x : (xs + +B)$

**List 3:** $member : T \rightarrow ListT \rightarrow Bool$

**M1**: $member(e, []) = False$

**M2**: $member(e, x : xs) = (x == e) \vee member(e, xs)$

**BTree 1: Constructori**

$empty :\to BTreeT$
$node(left, x, right) :\to BTreeT \to T \to BTreeT \to BTreeT$

**BTree 2:** $tolist : BTreeT \to ListT$

**T1**: $tolist(empty) = False$

**T2**: $tolist(node(l, x, r)) = tolist(l) + +(x : tolist(r))$

**BTree 3:** $memberBT : T \to T \to BTreeT \to Bool$

**MB1**: $memberBT(e, empty) = False$

**MB2**: $memberBT(e, node(l, x, r)) =$
$(e == x) \lor memberBT(l) \lor memberBT(r)$

**De demonstrat P1**

$memberBT(e, tree) == member(e, tolist(tree)),$
$\forall e \in T, tree \in BTree.$

## 3.1 Barem

- **(2p)** Demonstrația pentru cazul de bază.
- **(2p)** Specificarea ipotezei inductive și a pasului de inducție.
- **(3p)** Demonstrația pasului de inducție.
- **(1p)** Identificarea proprietății adiționale.
- **(2p)** Demonstrația proprietății adiționale.

## 3.2 Rezolvare

- **Cazul de bază**: $tree == empty$

  Ne propunem să demonstrăm:
  $memberBT(e, empty) = member(e, tolist(empty)).$

  $LHS = memberBT(e, empty) =^{MB1} False.$

  $RHS = member(e, tolist(empty)) =^{T1} member(e, []) =^{M1} False$

  $\implies LHS = RHS$

- **Ipoteza inductivă**:

  P(l): $memberBT(e, l) = member(e, tolist(l))$.

  P(r): $memberBT(e, r) = member(e, tolist(r))$.

- **Pasul de inducție**: $P(l), P(r) \implies P(node(l, x, r))$.

  Ne propunem să demonstrăm:
  $memberBT(e, node(l, x, r)) = member(e, tolist(node(l, x, r)))$.

$$\begin{aligned} LHS &= memberBT(e, node(l, x, r)) \\ &=^{MB2} (e == x) \vee memberBT(e, l) \vee memberBT(e, r) \\ &=^{II} (e == x) \vee member(e, tolist(l)) \vee member(e, tolist(r)) \end{aligned} \tag{1}$$

$$\begin{aligned} RHS &= member(e, tolist(node(l, x, r))) \\ &=^{T2} member(e, tolist(l) + +(x : tolist(r))) \end{aligned} \tag{2}$$

Vom folosi o proprietate adițională:

**PA:** $member(e, A + +B) == member(e, A) \vee member(e, B)$,
$\forall e \in T; A, B \in ListT$

$$\begin{aligned} RHS &=^{PA} member(e, tolist(l)) \vee member(e, tolist(x : tolist(r))) \\ &=^{M2} member(e, tolist(l)) \vee (e == x) \vee member(e, tolist(r)) \end{aligned} \tag{3}$$

$\implies LHS = RHS.$

**Mai avem de demonstrat proprietatea adițională** PA

- **Cazul de bază** $A == []$.
  Ne propunem să demonstrăm:
  $member(e, [] + +B) == member(e, []) \vee member(e, B)$
  $LHS = member(e, [] + +B) =^{A1} member(e, B)$
  $RHS = member(e, []) \vee member(e, B) =^{M1} member(e, B)$
  $\implies LHS = RHS.$
- **Ipoteza inductivă** $PA(xs)$
  $member(e, xs + +B) == member(e, xs) \vee member(e, B)$.

– **Pasul de inducție** $PA(xs) \implies PA(x : xs)$ Ne propunem să demonstrăm:
$member(e, (x : xs) + +B) == member(e, x : xs) \vee member(e, B)$

$$\begin{aligned} LHS &= member(e, (x : xs) + +B) \\ &=^{A2} member(e, x : (xs + +B)) \\ &=^{M2} (e == x) \vee member(e, xs + +B)) \\ &=^{II} (e == x) \vee member(e, xs) \vee member(e, B) \end{aligned} \tag{4}$$

$$\begin{aligned} RHS &= member(e, x : xs) \vee member(e, B) \\ &=^{M2} (e == x) \vee member(e, xs) \vee member(e, B). \end{aligned} \tag{5}$$

$\implies LHS = RHS.$

# References

[1] Laurent Bulteau, Falk Hüffner, Christian Komusiewicz, and Rolf Niedermeier. Multivariate algorithmics for np-hard string problems. *Bulletin-European Association for Theoretical Computer Science*, 114, 2014.