Laboratorul 1 – Proiectarea algoritmilor
Seria CD

Mihai Nan – mihai.nan.cti@gmail.com

Anul universitar 2020 – 2021

1 Probleme propuse

1. Implementați o funcție care ridică un număr natural x la o putere naturală n. Pentru acest exercițiu, funcția propusă trebuie să implementeze un algoritm **iterativ** (care **nu folosește recursivitate** – ***nerecursiv***).

   Determinați complexitatea temporală a algoritmului implementat de funcția propusă.

2. Implementați o funcție care ridică un număr natural x la o putere naturală n. Pentru acest exercițiu, funcția propusă trebuie să implementeze un algoritm recursiv care împarte problema inițială în 1 și $n-1$.

   $$x^n = x^1 \cdot x^{n-1}$$

   Precizați care este recurența de complexitate pentru algoritmul recursiv implementat de funcția propusă. Folosind una dintre metodele studiate în cadrul cursului de Analiza Algoritmilor (*metoda iterativă*, *metoda arborelui de recurență*, *metoda substituției*, *Teorema Master*), calculați recurența de complexitate obținută.

3. Implementați o funcție care ridică un număr natural x la o putere naturală. Pentru acest exercițiu, funcție propusă trebuie să implementeze un algoritm recursiv care împarte problema inițială în $\frac{n}{2}$ și $\frac{n}{2}$ dacă n este număr par sau în 1, $\frac{n-1}{2}$ și $\frac{n-1}{2}$ dacă n este număr impar.

   $$x^n = \begin{cases} 1 & \text{dacă } n = 0 \\ x^{\frac{n}{2}} \cdot x^{\frac{n}{2}} & \text{dacă } n > 0 \text{ și } n \text{ este par} \\ x^1 \cdot x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} & \text{dacă } n \text{ este impar} \end{cases}$$

   Precizați care este recurența de complexitate pentru algoritmul recursiv implementat de funcția propusă. Folosind una dintre metodele studiate în cadrul cursului de Analiza Algoritmilor (*metoda iterativă*, *metoda arborelui de recurență*, *metoda substituției*, *Teorema Master*), calculați recurența de complexitate obținută.

Important

Există vreo diferență, din perspectiva complexității temporale, între următoarele două abordări?

(a) Realizăm două apeluri recursive ale funcției pentru a calcula valoarea ce trebuie returnată;

(b) Realizăm un singur apel recursiv, reținem rezultatul într-o variabilă și apoi refolosim rezultatul determinat anterior pentru a calcula valoarea ce trebuie returnată.

4. Analizați complexitățile temporale obținute pentru funcțiile propuse anterior și determinați care variantă este cea mai eficientă.

5. Implementați o funcție **recursivă** care calculează elementul aflat pe poziția n în șirul lui Fibonacci. Funcția propusă trebuie să implementeze un algoritm recursiv care împarte problema inițială în următoarea variantă:

$$fib(nr) = \begin{cases} 0 & \text{dacă } nr = 0 \\ 1 & \text{dacă } nr = 1 \\ fib(nr-1) + fib(nr-2) & \text{dacă } nr > 1 \end{cases}$$

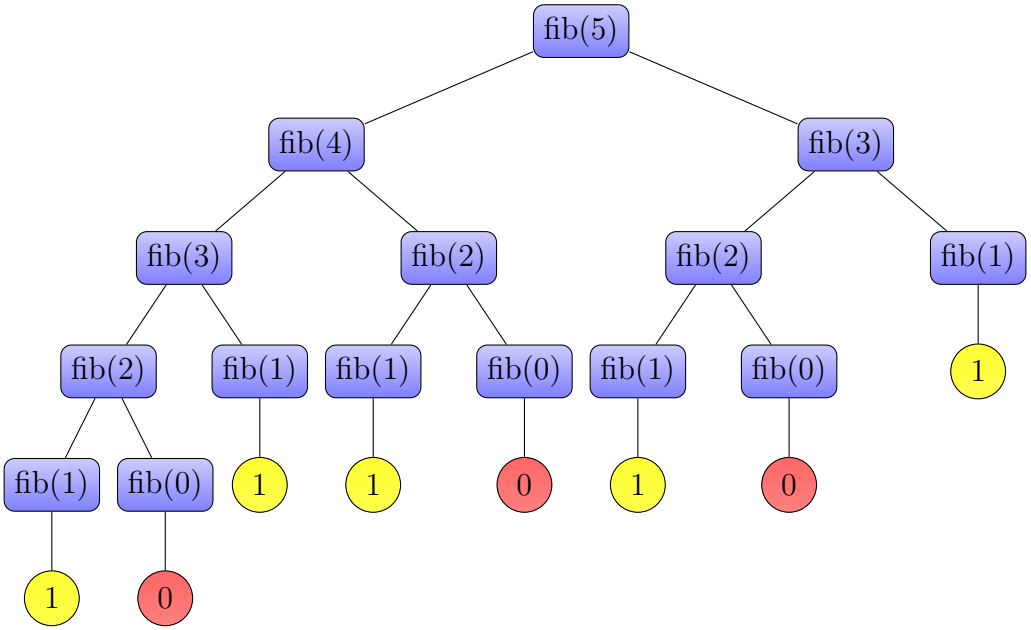

Figure 1: Arborele apelurilor recursive apelate pentru a calcula $fib(5)$

Precizați care este recurența de complexitate pentru algoritmul recursiv implementat de funcția propusă. Folosind una dintre metodele studiate în cadrul cursului de Analiza Algoritmilor (*metoda iterativă*, *metoda arborelui de recurență*, *metoda substituției*, *Teorema Master*), calculați recurența de complexitate obținută.

6. Implementați o funcție **nerecursivă** care calculează elementul aflat pe poziția n în șirul lui Fibonacci. Funcția propusă trebuie să implementeze un algoritm iterativ care folosește un vector în care va reține elementele aflate în șirul lui Fibonacci. Inițial, în acest vector se vor adăuga două valorile $vect[0] = 0$ și $vect[1] = 1$, după care restul valorilor se vor calcula folosind formula $vect[i] = vect[i-1] + vect[i-2]$, pentru $\forall i, i > 1$.

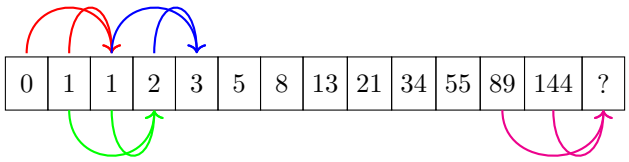

Figure 2: Exemplu de completare a vectorului folosit de funcție

Determinați complexitatea temporală a algoritmului implementat de funcția propusă.