PMF Kragujevac
Institut za matematiku i informatiku
Algoritamske strategije, test
28.10.2014.

Date su 2 rekurzivne procedure koje predstavljaju implementacije algoritama za određivanje članova Fibonačijevog niza:

1)
```
def fibonacci(n):
    if n == 0 or n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

2)
```
fibonacci_values = [-1 for 0..(DIM-1)]
fibonacci_values[0] = 1
fibonacci_values[1] = 1

def fibonacci_2(n):
    if fibonacci_values[n] == -1:
        fibonacci_2(n - 1)
        fibonacci_values[n] = fibonacci_values[n - 1] + fibonacci_values[n - 2]
```

U drugom algoritmu se koristi pomoćni niz. Njegova dužina je dovoljna da zapamti članove Fibonačijevog niza koje želimo da izračunamo (n je manje od DIM).

Za jedinicu mere složenosti ovih algoritama uzimamo broj izvršavanja komandi u funkcijama. Izvršavanje komande predstavlja jednu logičku jedinicu posla koji programski kod predstavlja. U prikazanom kodu imamo da funkcija fibonacci zavisno od if provere radi ili 2 koraka (**if** proveru i vraćanje vrednosti) ili 4 koraka (**if** proveru, 2 nova rekurzivna poziva i sabiranje). U funkciji fibonacci_2 za odgovarajuće uslove **if** komande imamo ili 3 koraka (**if** proveru, rekurzivni poziv i sabiranje) ili 1 korak (samo **if** proveru).

Neka su f1, f2, g1 i g2 aritmetičke funkcije. Ovim funkcijama merimo složenost izračunavanja ovih algoritama, redom u pogledu vremena rada, a zatim i upotrebe memorije. Posmatranjem njihovog rada merimo vreme/zauzeće u odnosu na n – željeni redni broj člana Fibonačijevog niza. Za sve funkcije dati ocenu u odnosu na koju funkciju imaju istu brzinu rasta (θ).