# Algoritamska tehnika: PODIJELI PA VLADAJ

**Problem:**

**Ulaz:** sortiran niz realnih brojeva $\langle a_1, a_2, \ldots, a_n \rangle$
element $x$

**Izlaz:** indeks elementa $x$ u polju $A$, ako je
$x \in a_j$ za neki $j \in \{1, \ldots, n\}$
NULL, inače

Ideja:
- podijeli: podijeli polje A na dva potpolja u odnosu na „srednji" element
- vladaj: rekurzivno pretraži jedno potpolje
- kombiniraj: trivijalno (ako u jednom potpolju nije, onda je u drugom)

Primjer: niz $A = \langle 3, 5, 7, 8, 9, 12, 15 \rangle$

$x = 9$

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 5 | 7 | 8 | 9 | 12 | 15 |

$9 > 8 \longrightarrow$ nije u lijevom potpolju

| | | |
|---|---|---|
| 9 | 12 | 15 |

$9 < 12$

| |
|---|
| 9 |

Izlaz: 5, tj: $A[5] = 9$

Pseudokod:

```
BINARY SEARCH (A, x, p, r)

while p ≤ r
    do m ← ⌊(p+r)/2⌋
    if x = A[m]
        then return m
    if x > A[m]
        then p ← m+1
        else r ← m-1;
    end if
end while
```

Rekurzivna verzija

```
BINARY SEARCH (A, x, p, r)
~~while~~
if p > r then
    return NIL
m ← ⌊(p+r)/2⌋
if x = A[m] return m;
else if x > A[m]
    return BINARY-SEARCH
else
    retur BINARY-SEARCH
end if
```

* Korektnost: Domaća zadaća (Invarijanta ???)

* Vremenska složenost:

$$T(n) = T(n/2) + \Theta(1)$$

Master metoda:

$a = 1$ , $b = 2$ $\Rightarrow$ $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) \in \Theta(1)$

$\Rightarrow$ slučaj 2. $\boxed{T(n) = \Theta(\lg n)}$

Zaključak: Brže od linearnog pretraživanja ali polje mora biti sortirano !

# POTENCIRANJE BROJA

roblem:

Ulaz: $a \in \mathbb{R}$, $n \in \mathbb{N}_0$

Izlaz: $a^n$

naivni algoritam za računanje n-te potencije realnog broja a
naučili smo u osnovnoj školi.
zasnovan je na definiciji n-te potencije:

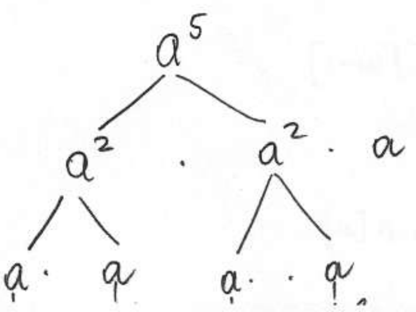
$$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n \text{ puta}}, \quad n \geq 1$$

razmotrit ćemo primjenu tehnike podijeli-pa-vladaj na
vaj problem:

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2}, & n \text{ paran} \\ a^{n-1/2} \cdot a^{n-1/2} \cdot a, & n \text{ neparan} \end{cases}$$

```
NthPower (a, n)
if  n = 0
    then return 1;
else
    if n % 2 = 0
        then return NthPower(a, n/2) · NthPower(a, n/2)
    else return NthPower(a, n-1/2) · NthPower(a, n-1/2) · a
end if
```

mjer:



## Vremenska složenost algoritama za potenciranje brojeva

- naivni pristup: $\Theta(n)$
  - operaciju možemo proveli smo točno $n-1$ puta
- podijeli-pa-vladaj: NTHPOWER algoritam
  - algoritam je rekurziran
  - vrijeme izvršavanja dano je sljedećom rekurzijom:

$$T(n) = T(n/2) + \Theta(1)$$

Master metoda: $a=1$, $b=2$ $\quad n^{\log_2 1} = n^0 = 1 \quad f(n) = \Theta(1)$

2. slučaj Master teorema: $T(n) = \Theta(n^0 \lg n)$

$$\boxed{T(n) = \Theta(\lg n)}$$

## FIBONACCIJEVI BROJEVI

Problem:

Ulaz: $n \in \mathbb{N}_0$

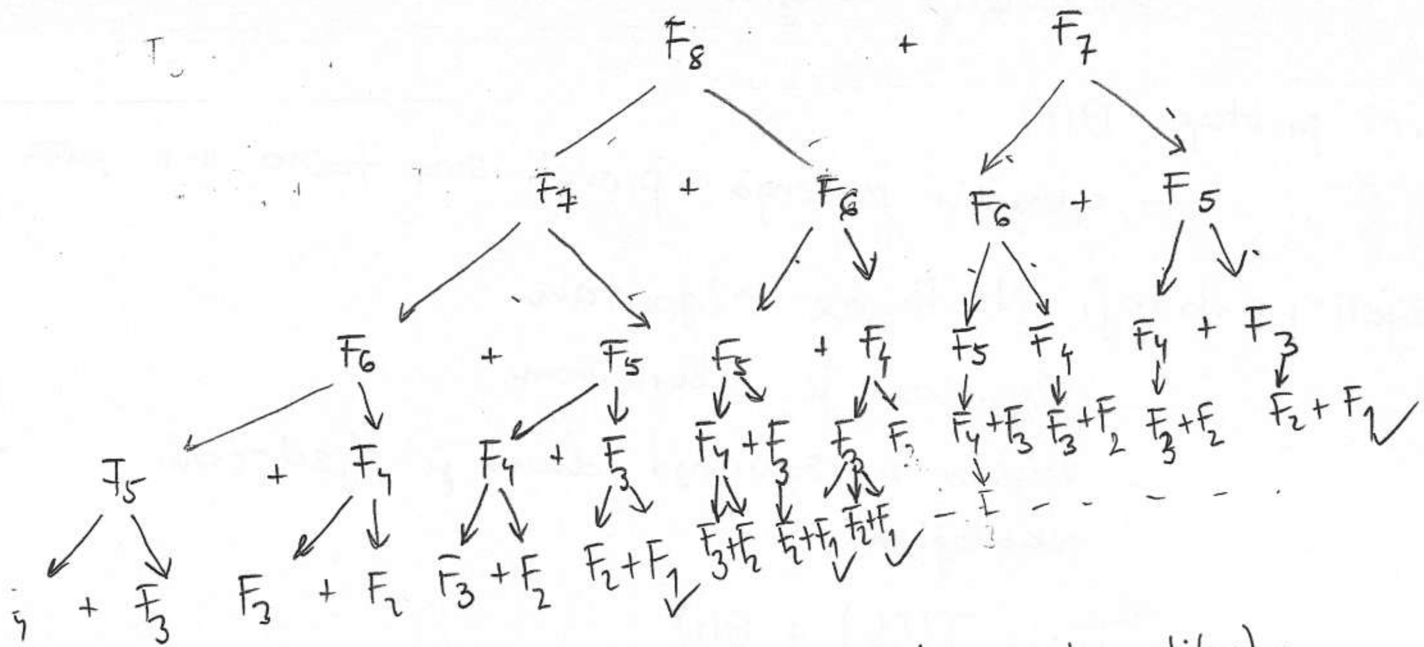
Izlaz: $n$-ti Fibonaccijev broj $F_n$

- rekurzivna definicija

$$F_n = \begin{cases} 0 & , n=0 \\ 1 & , n=1 \\ F_{n-1} + F_{n-2} & , n \geq 2 \end{cases}$$

```
NAIVE-FIBONACCI (n)
if n=0 OR n=1
    then return 1
else
    return FIBONACCI(n-1) + FIBONACCI(n-2)
```

Primjer: $n=9$



rješenje rekurzije (kombinatorna i diskretna matematika):

$$T(n) = \Omega(\Phi^n), \quad \Phi = \frac{1+\sqrt{5}}{2}$$

$\Phi$ → zlatni rez

vrijeme izvršavanja $T(n) = T(n-1) + T(n-2) + \Theta(1)$

Računanje od dna rekurzije:

ideja: izračunati $F_0, F_1, \ldots, F_n$ po redu računajući svaki član koristeći prethodna dva člana

$F_2 = F_1 + F_0$

$F_3 = F_2 + F_1$

$F_4 = F_3 + F_2$

$F_5 = F_4 + F_3$

⋮

$F_{n-1} = F_{n-2} + F_{n-3}$

$F_n = F_{n-1} + F_{n-2}$

```
BOTTOM UP FIBONACCI (n)
Fpp = 0;
Fp = 1;
for i = 2 to n
    F ← Fp + Fpp
    Fpp ← Fp
    Fp ← F
end for
return F
```

polje potrebno???

- vremenska složenost algoritma BOTTOMUP FIBONACCI: $\Theta(n)$
- uvijek izvršimo točno $n-1$ operacija zbrajanja i $3(n-1)+2$ operacija pridruživanja

$$\begin{aligned} T(n) &= n-1 + 3(n-1) + 2 \\ &= n-1 + 3n - 3 + 2 \\ &= 4n - 2 = \Theta(n) \end{aligned}$$

- Zašto je BOTTOM UP brži od NAIVEFIBONACCI algoritma?

Ako pogledamo u stablo koje opisuje rekurzivne pozive algoritma za primjer $n=9$ vidjet ćemo da smo u različitim pozivima nepotrebno ponavljali računanje Fibonaccijevih brojeva koje smo već izračunali u nekom od prijašnjih poziva. Kod BOTTOM UP algoritma svaki Fibonaccijev broj $F_i$ računamo točno jednom pomoću prethodna dva, $F_{i-1}$ i $F_{i-2}$, za i≥2 koje smo spremili u pomoćne varijable.

# MNOŽENJE MATRICA

Problem:

Ulaz: $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times r}$ $\quad m, n, r \in \mathbb{N}$

Izlaz: $C \in \mathbb{R}^{m\times r}$ $\quad C = A \cdot B$

prisjetimo se definicije množenja matrica A i B.
Neka su $m, n, r \in \mathbb{N}$, $A \in \mathbb{R}^{m\times n}$ i $B \in \mathbb{R}^{n\times r}$. Produkt matrica A i B je matrica $C \in \mathbb{R}^{m\times r}$ t.d

$$[C]_{ij} = \sum_{k=1}^{n} [A]_{ik} \cdot [B]_{kj}$$

za $i \in \{1, \ldots, m\}$ i $j \in \{1, \ldots, r\}$.

pseudokod "naivnog" algoritma:

```
MATRIX-MULTIPLY (A, B)
m ← numberOfRows(A)
if numberOfColumns(A) = numberOfRows(B)
|   then  n ← numberOfRows(B)
else
|   print: "Greška! Produkt od A i B nije definiran";
endIf
r ← numberOfColumns(B)
for i ← 1 to m
|   for j ← 1 to r
|   |   C[i][j] ← 0;
|   |   for k ← 1 to n
|   |   |   C[i][j] ← C[i][j] + A[i][k] · B[k][j]
|   |   endFor
|   endFor
endFor
```

## Vremenska složenost algoritma MATRIX-MULTIPLY

Pretpostavimo da je $m = n = r$. Tada je broj operacija

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (3n+1) = 3n^3 + n^2 = \Theta(n^3)$$

# PODIJELI - PA - VLADAJ ALGORITAM

oblem:

Ulaz: $A \in R^{m \times n}$, $B \in \mathbb{R}^{n \times r}$, $m, n, r \in \mathbb{N}$

Izlaz: $C \in R^{m \times r}$, $C = A \cdot B$

EJA: primijeniti strategiju PODIJELI - PA - VLADAJ

retpostavimo da je $n = m = r$; te da je $n \bmod 2 = 0$

$a \in \mathbb{R}^{n/2 \times n/2}$

$$A = \begin{array}{c} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\ \begin{array}{cc} \underbrace{\quad}_{n/2} & \underbrace{\quad}_{n/2} \end{array} \end{array} \qquad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{array}{l} \}\, n/2 \\ \}\, n/2 \end{array} \qquad C = \begin{bmatrix} r & s \\ t & u \end{bmatrix}$$

$$A \cdot B = C$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} r & s \\ t & u \end{bmatrix}$$

$$\left.\begin{aligned} r &= a \cdot e + bg \\ s &= a \cdot f + + b \cdot h \\ t &= c \cdot e + d \cdot g \\ u &= c \cdot f + d \cdot h \end{aligned}\right\}$$

8 **množenja** blok matrica veličine $(n/2) \times (n/2)$

4 zbrajanja blok matrica veličine $(n/2) \times (n/2)$

## Analiza vremena izvršavanja

· rekurzija:

$$T(n) = 8 \cdot T(n/2) + \Theta(n^2)$$

rekurzivno množenje

zbrajanje matrica

Master metoda:

$$n^{\log_b a} = n^{\log_2 8} = n^3 \Rightarrow f(n) = \Theta(n^2) = O(n^{\log_2 8 - 1})$$

$$\Rightarrow \boxed{T(n) = \Theta(n^3)}$$

- zaključak: Nismo poboljšali vrijeme izvršavanja u odnosu na standardno množenje

STRASSENOVO MNOŽENJE MATRICA

- ideja: umjesto 8 množenja provesti 7 množenja ali više operacija zbrajanja i oduzimanja koje su „jeftinije".

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$P_1 = a \cdot (f - h)$

$P_2 = (a+b) \cdot h$

$P_3 = (c+d) \cdot e$

$P_4 = d \cdot (g-e)$

$P_5 = (a+d)(e+h)$

$P_6 = (b-d) \cdot (g+h)$

$P_7 = (a-c)(e+f)$

$r = P_5 + P_4 - P_2 + P_6$

$s = P_1 + P_2$

$t = P_3 + P_4$

$u = P_5 + P_1 - P_3 - P_7$

Provjera:

$r = P_5 + P_4 - P_2 + P_6 = a \cdot e + b \cdot g$

$s = P_1 + P_2 = a \cdot f + b \cdot h$

$t = P_3 + P_4 = c \cdot e + d \cdot h$

$u = P_5 + P_1 - P_3 - P_7 = c \cdot f + d \cdot g$

Algoritam:

STRASSEN-MULTIPLY (A, B)

1. Podijeli: - podijeli A i B u 2×2 blok matrice veličine $(n/2) \times (n/2)$.
- definiraj $P_1, P_2, \dots, P_7, r, s, t, u$ prema definiciji

2. Vladaj: - izvrši 7 množenja matrica veličine $(n/2) \times (n/2)$ rekurzivno

3. Spoji (kombiniraj):
- izračunaj C prema definiciji od $r, s, t, u$ (zbrajanje i oduzimanje matrica veličine $(n/2) \times (n/2)$

Analiza vremena izvršavanja

$$T(n) = 7T(n/2) + \Theta(n^2)$$

Master metoda:

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \overset{1.\text{ slučaj}}{\Rightarrow} f(n) = \Theta(n^2) \qquad \varepsilon = 1/2$$

$$\Rightarrow T(n) = \Theta(n^{\lg 7})$$

Zadatak: Odrediti $n_0$ takav da je za $n \geq n_0$ Strassenov algoritam brži od standardnog množenja matrica.

**Primjer:** Pomnožite sljedeće matrice $A \in R^{4\times 4}$ i $B \in R^{4\times 4}$ koristeći Strassenov algoritam:

$$A = \begin{array}{cc} a & b \\ \multicolumn{2}{c}{\left[\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{array}\right]} \\ c & d \end{array} \qquad B = \begin{array}{cc} e & f \\ \multicolumn{2}{c}{\left[\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{array}\right]} \\ g & h \end{array}$$

Podijeli:

$$f-h = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & -1 \end{bmatrix}$$

$$a+b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$$

$$c+d = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$$

$$g-e = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$a+d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}$$

$$e+h = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 1 & 1 \end{bmatrix}$$

$$b-d = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$g+h = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}$$

$$a-c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$e+f = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$P_1 = a \cdot (f-h) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$P_2 = (a+b) \cdot h = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix}$$

$$P_3 = (c+d) \cdot e = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix}$$

$\vdots$

D.Z.