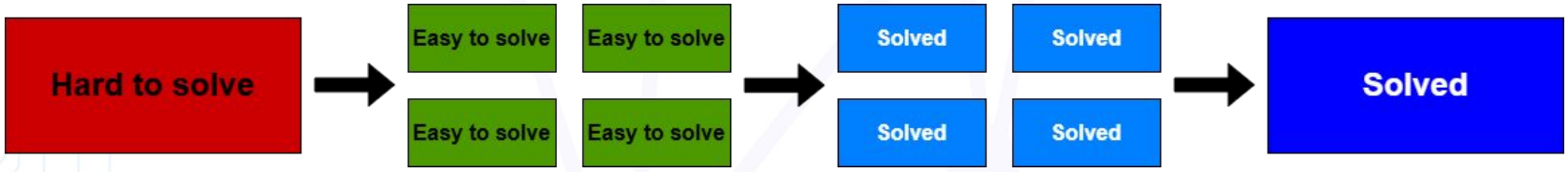
Teorija Algoritama

Strategija: Zavadi pa vladaj

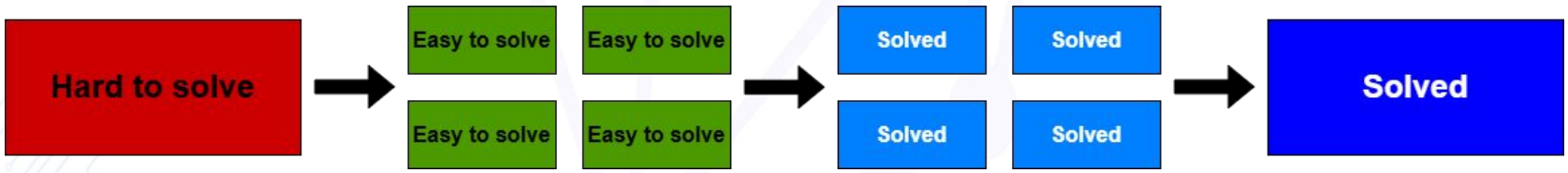
Zavadi pa vladaj pristup

- Jedan od osnovnih metoda za konstrukciju algoritama.
- *Divide and Conquer (lat. Divide et impera)*
  - Možda bolji prevod: “Podeli pa reši”
- Predstavlja pristup rešavanju problema gde se:
  - Jedan početni (veći) problem podeli na više manjih istog tipa, ali pogodnijih za rešavanje;
  - Potom se kombinacijom rešenja manjih problema reši i početni veći problem.



Zavadi pa vladaj pristup

- Osnovni koraci:
  - **Podela** - osnovni problem se dekomponuje na jedan ili više istih problema obima manjeg od početnog;
  - **Rešavanje** - problemi manjeg obima se rešavaju rekurzivno;
  - **Kombinovanje** - rešenja dobijena rešavanjem problema manjeg obima se kombinuju u rešenje početnog (osnovnog) problema.
- Zavadi pa vladaj je u osnovi **rekurzivni** pristup, iako implementacija ovakvih algoritama ne mora nužno biti preko rekurzivnih funkcija.



Zavadi pa vladaj - primeri

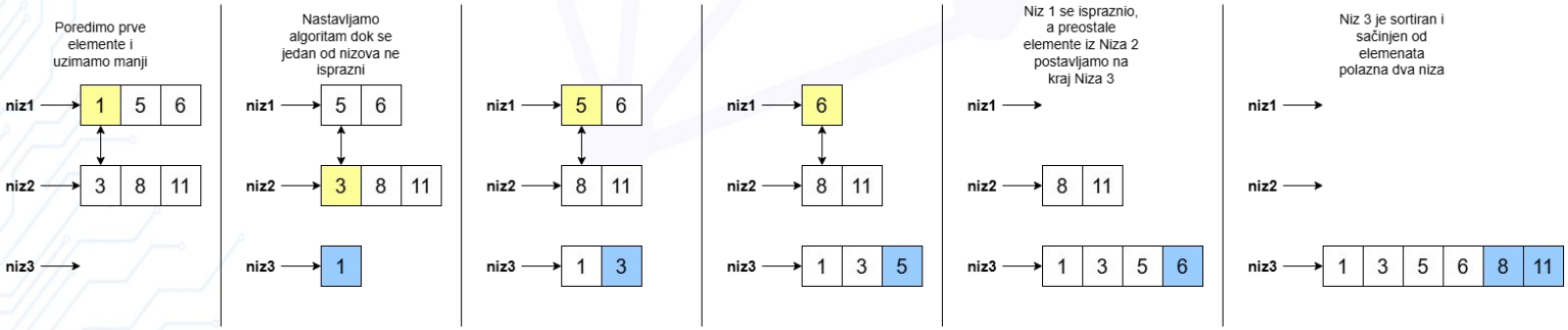
- Stepenovanje - odrediti ***n***-ti stepen realnog broja ***x***;
- *Merge-sort* - sortiranje spajanjem;
- Strassen-ov metod za množenje matrica;
- *Karatsuba* metod za množenje velikih brojeva;
- Binarna pretraga;
- Najveći zajednički delilac.

Stepenovanje

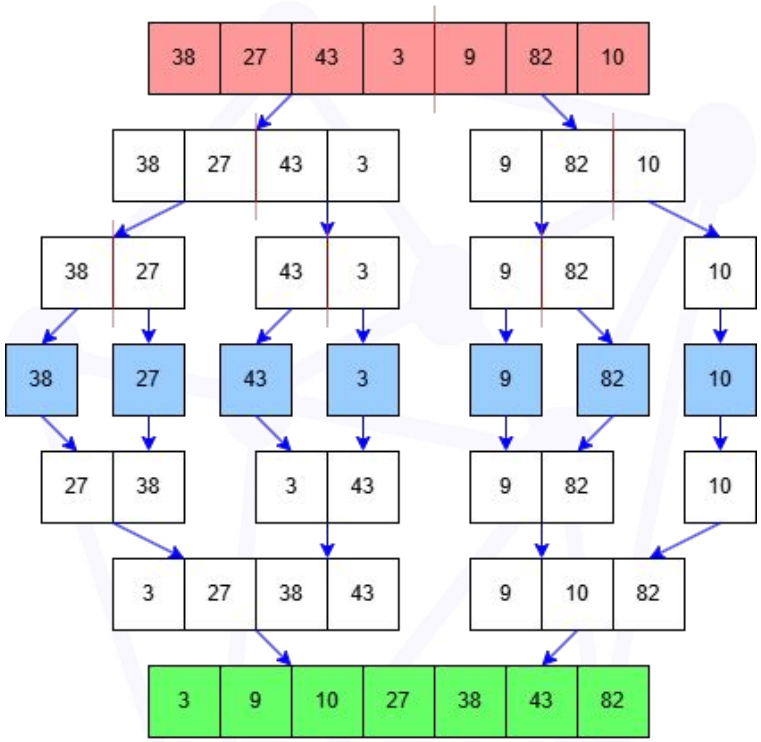
- Osnovni problem: izračunati ***n***-ti stepen od realnog broja ***x***.
- Algoritam:
  - Podela:
    - $n\%2 == 0 \Rightarrow x^n = x^{\frac{n}{2}} \cdot x^{\frac{n}{2}}$
    - $n\%2 == 1 \Rightarrow x^n = x^{\frac{n}{2}} \cdot x^{\frac{n}{2}} \cdot x$
    - Ako je ***n > 1*** rekurzivno rešiti za dvostruko manje ***n***.
  - Rešavanje:
    - Ako je n ≤ 1, vrati ***x*** ako je ***n = 1*** ili **1** ako je ***n = 0***.
  - Kombinovanje:
    - Kvadrirati $x^{\frac{n}{2}}$ i eventualno pomnožiti sa ***x***.
- Primer (2^9)
  - Naivni pristup:
    $2^9 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2$ - 8 operacija
  - *Zavadi pa vladaj* pristup:
    $2^9 = 2^4 \cdot 2^4 \cdot 2$ - 2 operacije;
    $2^4 = 2^2 \cdot 2^2$ - 1 operacija;
    $2^2 = 2 \cdot 2$ - 1 operacija;
    Ukupno 4 operacije.

Merge-sort - sortiranje spajanjem

- *Merge-sort* algoritam:
  - Podela:
    - Početni niz se podeli na dva niza približno jednake dužine;
    - Ponavlja se dok podnizovi ne postanu jedinični.
  - Rešavanje:
    - Ukoliko je niz jediničan (dužine 1), sortiran je.
  - Kombinovanje:
    - Dva sortirana niza se spajaju u jedan sortirani niz.
- Algoritam za spajanje dva sortirana niza:
  - Uporediti prve elemente sortiranih nizova;
  - Manji element (ili veći u zavisnosti od redosleda sortiranja) uzeti i prebaciti u treći niz.
  - Kada se jedan niz isprazni, preostale elemente iz drugog niza dodati na kraj trećeg niza.
  - Treći niz je sortiran i sačinjen je od elemenata početna dva niza.



Merge-sort - primer



- Niz dužine ***n*** možemo podeliti ***log n*** puta.
- Prilikom spajanja takođe ćemo podnizove spojiti ***log n*** puta, a u svakom spajanju ćemo uporediti u najgorem slučaju svih ***n*** elemenata, dakle ***n log n*** operacija.
- Dakle imamo ***log n + n log n*** operacija, tj. ***O(n log n)***.

Množenje matrica

- Osnovni problem: množenje kvadratnih matrica.
- **Naivni pristup:**
  - Pseudo-kod:

```
ALGORITHM SquareMarixMultiply(A, B):
    n = A.rows
    C = Matrix.zeros(n,n) # Initialize the result Matrix filled with zeros
    FOR i FROM 0 TO n-1:
        FOR j FROM 0 TO n-1:
            FOR k FROM 0 TO n-1:
                C[i][j] = C[i][j] + (A[i][k] * B[k][j])
```

  - Ovaj pristup ima **kubnu** vremensku kompleksnost, što svakako nije optimalno.

Množenje matrica

- Jednostavni zavadi pa vladaj pristup:
  - Podela:
    - Matricu podeliti na četiri matrice;
    - Matrice deliti dok ne postanu dimenzija 1×1.
  - Rešavanje:
    - Ako su matrice dimenzije 1×1, problem se svodi na množenje dva broja.
  - Kombinovanje:
    - Od 8 proizvoda matrica dimenzija $\frac{n}{2} \times \frac{n}{2}$ se dobije rezultujuća matrica.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21}, \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22}, \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21}, \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22}, \end{aligned}$$

- Rekurzivna formula:
  - Osam množenja matrica $\frac{n}{2} \times \frac{n}{2}$ i četiri sabiranja: $T(n) = 8 \cdot T\left(\frac{n}{2}\right) + 4 \cdot \left(\frac{n}{2}\right)^2$
  - Primer (8×8 matrica):

$$T(8) = 8 \cdot T(4) + 4 \cdot 4^2 = 8 \cdot \left(8 \cdot T(2) + 4 \cdot 2^2\right) + 4 \cdot (4)^2 = 8 \cdot \left(8 \cdot \left(8 \cdot T(1) + 4 \cdot 1^2\right) + 4 \cdot 2^2\right) + 4 \cdot (4)^2$$

  - Ako je $T(1) = 1$ (množenje dva skalara), $T(8) = 960$.
  - Ako je $8^3 = 512$, a $8^4 = 4096$, možemo doći do zaključka da ovaj pristup takođe ima kubnu vremensku složenost.

Množenje matrica - *Strassen*-ov metod

$$
\begin{aligned}
S_1 &= B_{12} - B_{22},\\
S_2 &= A_{11} + A_{12},\\
S_3 &= A_{21} + A_{22},\\
S_4 &= B_{21} - B_{11},\\
S_5 &= A_{11} + A_{22},\\
S_6 &= B_{11} + B_{22},\\
S_7 &= A_{12} - A_{22},\\
S_8 &= B_{21} + B_{22},\\
S_9 &= A_{11} - A_{21},\\
S_{10} &= B_{11} + B_{12}.
\end{aligned}
\quad\longrightarrow\quad
\begin{aligned}
P_1 &= A_{11} \times S_1,\\
P_2 &= S_2 \times B_{22},\\
P_3 &= S_3 \times B_{11},\\
P_4 &= A_{22} \times S_4,\\
P_5 &= S_5 \times S_6,\\
P_6 &= S_7 \times S_8,\\
P_7 &= S_9 \times S_{10}.
\end{aligned}
\quad\longrightarrow\quad
\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6,\\
C_{12} &= P_1 + P_2,\\
C_{21} &= P_3 + P_4,\\
C_{22} &= P_5 + P_1 - P_3 - P_7.
\end{aligned}
$$

- Rekurzivna formula
  - Ovim metodom imamo 7 množenja matrica $\frac{n}{2} \times \frac{n}{2}$ i 18 sabiranja/oduzimanja:
    $T(n) = 7 \cdot T\left(\frac{n}{2}\right) + 18 \cdot \left(\frac{n}{2}\right)^2$, dakle $T(8) = 2017$.
  - Iako deluje kao da je ovo “neefikasniji” pristup od prethodnog, na kasnijim slajdovima ćemo nekom pouzdanijom metodom dokazati da je zapravo *Strassen*-ov metod efikasniji.

Aritmetika asimptotskih notacija

- $f(n) = O(f(n))$
- $c \cdot O(f(n)) = O(f(n))$
- $O(O(f(n))) = O(f(n))$
- $f_1 = O(g_1), f_2 = O(g_2) \implies f_1 + f_2 = O(max(g_1, g_2))$
- $f_1 = O(g_1), f_2 = O(g_2) \implies f_1 \cdot f_2 = O(g_1 \cdot g_2)$
- $f \cdot O(g) = O(f \cdot g)$

Vremenska složenost

- Vremenska složenost algoritama dobijenog pristupom zavadi pa vladaj zavisi i od broja i veličine potproblema.
- Mogući pristupi za određivanje složenosti algoritma (O i Θ notacije), koji se oslanja na metod zavadi pa vladaj, su:
  - Zamena (supstitucija);
  - Stablo rekurzije;
  - Master metod.

Metod zamene

- Sastoji se od:
  - Pretpostavke funkcije složenosti;
  - Korišćenje matematičke indukcije za dokazivanje da je ta složenost tačna.
- Primeri:
  - Dokazati da je složenost rekurzivnog stepenovanja logaritamska, odnosno ***T(n) = O(log n)***;
  - Dokazati da je složenost *merge sort*-a logaritamsko-linearna, odnostno ***T(n) = O(n log n)***.

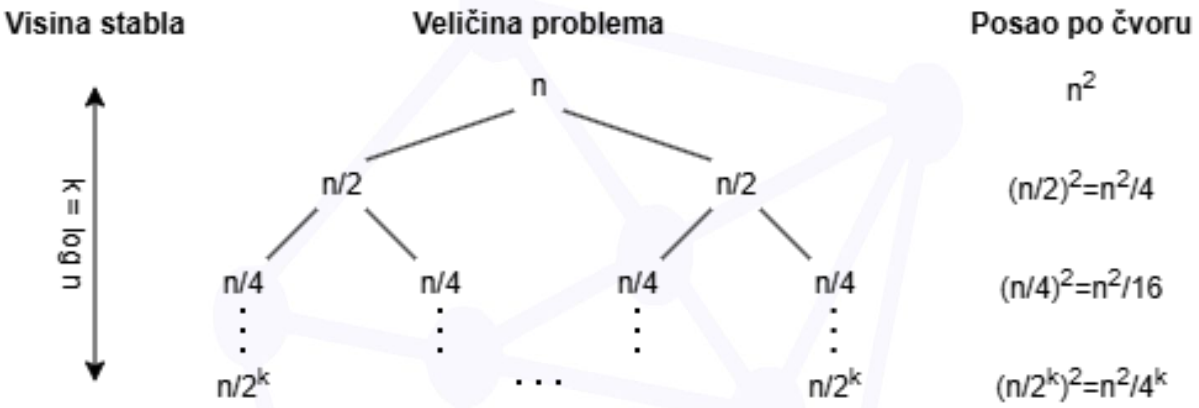
Metod zamene - primer

- Dokazati da je složenost rekurzivnog stepenovanja logaritamska.
- Prvo uspostavimo rekurzivnu formulu: $T(n) = T\left(\frac{n}{2}\right) + 1$
- Pretpostavimo da je $T(n) \leq c \cdot log(n)$, za neko $c > 0$.
- Dokaz izvodimo metodom matematičke indukcije.
- Baza indukcije (n = 2):
  - $T(2) \leq c \cdot log(2) = c$
  - $T(2) = T(1) + 1, T(1) = 0, T(2) = 0 + 1 = 1$
- Induktivni korak:
  - Pretpostavimo da važi tvrdnja za $\frac{n}{2}$, tj. $T\left(\frac{n}{2}\right) \leq c \cdot log\frac{n}{2}$.
  - Iz toga treba dokazati da postoji $c > 0$ tako da važi $T(n) \leq c \cdot log(n)$.
  - $T(n) = T\left(\frac{n}{2}\right) + 1 \leq c \cdot log\frac{n}{2} + 1 = c \cdot (log(n) - log(2)) + 1$
  - $T(n) \leq c \cdot (log(n) - 1) + 1 = c \cdot log(n) - c + 1$
  - Treba da važi $c \cdot log(n) - c + 1 \leq c \cdot log(n)$.
  - $1 - c \leq 0 \rightarrow c \geq 1$, dakle postoji traženo $c > 0$.
- Na osnovu baze indukcije i induktivnog koraka sledi da je $T(n) \leq c \cdot log(n)$.

Metod stabla rekurzije

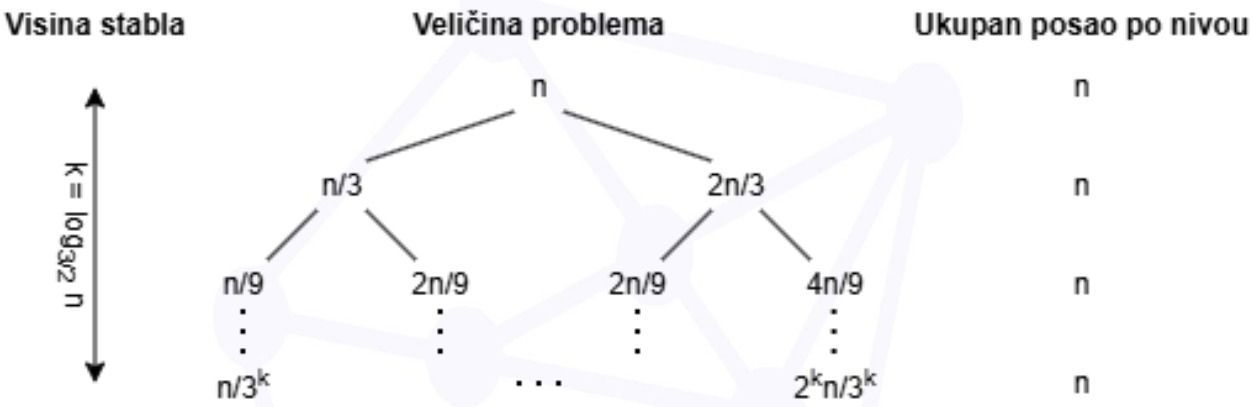
- Intuitivna tehnika.
- Formira se stablo rekurzije:
  - Koren stabla je početni problem;
  - Deca korena su potproblemi;
  - Listovi stabla su “jedinični potproblemi”.
- Svako podstablo ima složenost koja odgovara složenosti kombinacije njegovih potproblema.
- Kako bi se odredio ukupan broj čvorova u stablu potrebno je odrediti visinu stabla rekurzije i broj čvorova na svakom nivou.
- Izračunavanje složenosti metodom stabla rekurzije se svodi na sumiranje članova niza (najčešće geometrijskog).

MSR - primer $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n^2$



- Ukupno vreme $T(n)$ se može prikazati kao:
  $T(n) = n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \cdots + \Theta(n) = n^2\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{\log_2 n - 1}}\right) + \Theta(n)$
  gde je ***Θ(n)*** rad u listovima.
- U zagradama imamo geometrijski niz sa koeficijentom 0.5, čija suma teži u 2.
- Dakle, najbrže rastući član je n^2, što znači da je vremenska složenost kvadratna.

MSR - primer $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2 \cdot n}{3}\right) + n$



- Ukupan rad po nivou je konstantan - ***n***.
- Visina stabla određena je najdužom granom i iznosi $k = log_{\frac{3}{2}} n$.
- Ukupno vreme ***T(n)*** je suma rada kroz sve nivoe i iznosi približno $\sum_{i=0}^{k} n$.
- Dakle, vremenska složenost je linearno-logaritamska.

Master metoda

- Master metoda može da se koristi za određivanje funkcije složenosti za rekurzije tipa: $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$
  gde su ***a*** i ***b*** konstante za koje važi $a \geq 1, b > 1$, a $f(n)$ mora biti asimptotski pozitivna.

1. Identifikacija prethodno pomenutih parametara.
2. Izračunavanje "granične" funkcije $w(n) = n^{log_b a}$, koja nam govori koliko brzo raste broj potproblema u odnosu na njihovu veličinu.
3. Određivanje slučaja poređenjem asimptotskog rasta funkcija ***w(n)*** i ***f(n)***:
   a. $f(n) << w(n)$
      Uslov: $f(n) = O(n^{log_b a - \epsilon}), \epsilon > 0$
      Rešenje: $T(n) = \Theta\left(n^{log_b a}\right)$
   b. $f(n) = w(n)$ (približno isti asimptotski rast)
      Uslov: $f(n) = \Theta\left(n^{log_b a} \cdot log^k n\right)$, gde je ***k*** stepen logaritma funkcije ***f(n)*** (uglavnom ***k = 0***)
      Rešenje: $T(n) = \Theta(n^{log_b a} \cdot log^{k+1} n)$
   c. $f(n) >> w(n)$
      Uslov: $f(n) = \Omega\left(n^{log_b a + \epsilon}\right), \epsilon > 0$, i mora da važi $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n), c < 1$
      Rešenje: $T(n) = \Theta(f(n))$

Kada se Master Metod ne može koristiti?

- Parametar ***a*** nije konstanta (npr. $T(n) = n \cdot T\left(\frac{n}{2}\right)$);
- Parametar ***b ≤ 1***, tj. problem se ne smanjuje;
- Funkcija ***f(n)*** nije polinomijalna (npr. $f(n) = 2^n$);
- Razlika između ***f(n)*** i ***w(n)*** nije polinomijalna (npr. razlika je samo u logaritamskom factoru, a nije u pitanju drugi slučaj)

Master Metod - Primer sa slajda 9.

- Funkcija rekurzije: $T(n) = 8 \cdot T\left(\frac{n}{2}\right) + \Theta\left(n^2\right)$
- Granična funkcija: $w(n) = n^{log_2 8} = n^3$
- Pošto granična funkcija asimptotski raste brže od $\Theta(n^2)$, upadamo u prvi slučaj.
- Dakle rezultat je: $T(n) = \Theta\left(n^{log_2 8}\right) = \Theta\left(n^3\right)$

Master metod - *Strassen*-ov metod (slajd 10)

- Funkcija rekurzije: $T(n) = 7 \cdot T\left(\frac{n}{2}\right) + \Theta\left(n^2\right)$
- Granična funkcije: $w(n) = n^{log_2 7} \approx n^{2.81}$
- Ponovo se nalazimo u prvom slučaju, dakle rešenje je: $T(n) \approx \Theta\left(n^{2.81}\right)$

Master Metod - Primeri za vežbanje

1. $T(n) = 8 \cdot T\left(\frac{n}{2}\right) + n$
2. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \frac{n}{log(n)}$
3. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$
4. $T(n) = 9 \cdot T\left(\frac{n}{3}\right) + n^{\frac{3}{2}}$
5. $T(n) = 3 \cdot T\left(\frac{n}{4}\right) + n^3$
6. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n^2 \cdot log^2 n$
7. $T(n) = T\left(n-1\right) + n$
8. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n^2$