ASIMPTOTSKO VREME IZVRŠAVANJA

VREME IZVRŠAVANJA ALGORITMA

- Iako je moguće za neke algoritme odrediti tačno vreme izvršavanja, to nije praktično (previše sitnih koraka o kojima treba voditi računa)
- Dobar uvid u složenost algoritma zasniva se na pravoj meri pojednostavljenja njihove analize
- Analiza algoritma pojednostavljuje se do krajnjih granica kako bi se pravilno odredilo asimptotsko vreme izvršavanja algoritma
  - Vreme izvršavanja algoritma samo za veliki broj ulaznih podataka
- Ne interesuje nas apsolutno tačno vreme izvršavanja nekog algoritma, već samo to koliko brzo raste vreme izvršavanja algoritma sa povećanjem veličine ulaza algoritma
- Uobičajeno algoritam koji je asimptotski efikasniji je ujedno najbolji izbor za većinu slučajeva, osim kada je ulaz mali

ASIMPTOTSKO VREME IZVRŠAVANJA …

- Prilikom analize algoritama mogu se dobiti vrlo komplikovane funkcije za njihova vremena izvršavanja
  - $T_1(n) = 10n^3 + n^2 + 40n + 80$
  - $T_2(n) = 17n \log n - 23n + 10$
- Izrazi mogu biti još komplikovaniji
- Da li složenost funkcije zamagljuje sliku o odgovarajućem algoritmu?
- Da li na osnovu $T_1(n)$ i $T_2(n)$ možemo zaključiti koji je od dva algoritma brži?
- Nacrtati funkcije za $T_1(n)$ i $T_2(n)$ i uporediti ih (videti za različite n argumenta koja funkcija ima manje vrednosti)
- Da li je to praktično?
- Ali nije ni potrebno u analizi algoritama

… ASIMPTOTSKO VREME IZVRŠAVANJA …

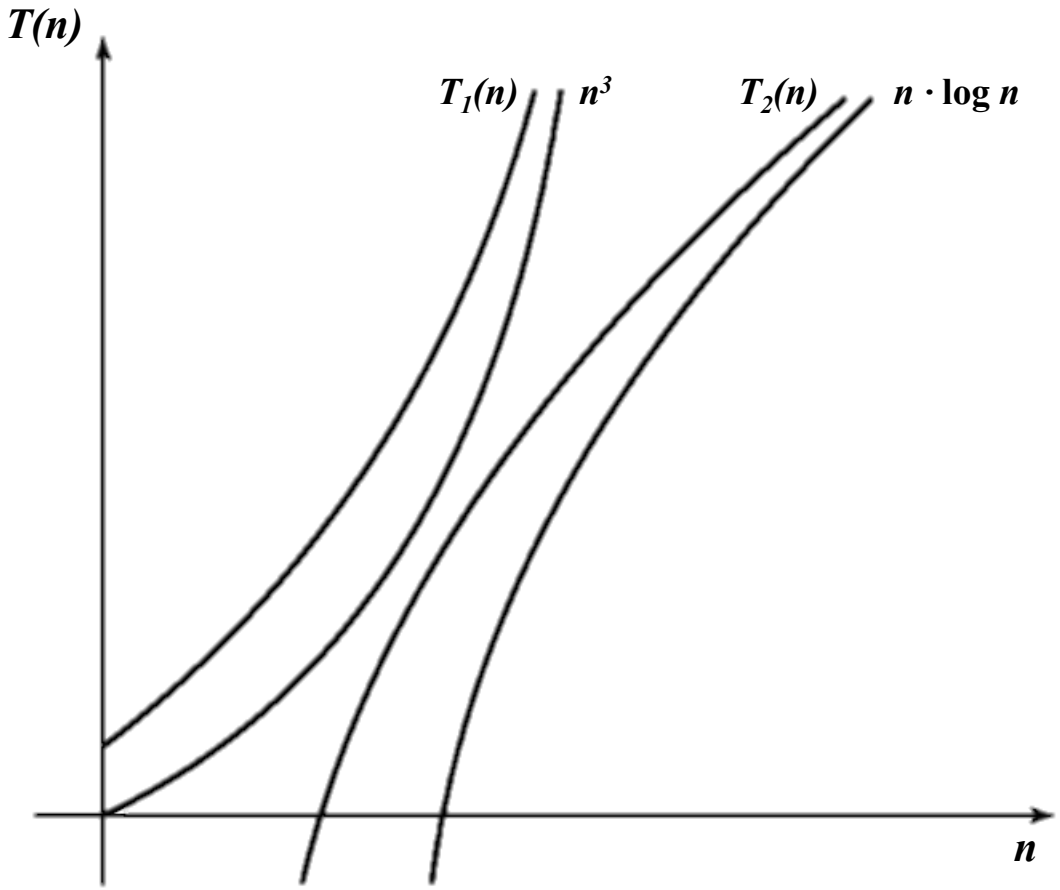
- Zašto nije potrebno crtati i analizirati cele funkcije?
  - $T_1(n) = 10n^3 + n^2 + 40n + 80$
  - $T_2(n) = 17n \log n - 23n + 10$
- Za ***n = 1 000 000***, koliko je ***$T_1(n)$*** i ***$T_2(n)$*** ?
  - ***$T_1(1000000)$ = 10 000 001 000 040 000 000***
  - ***$T_2(1000000)$ = 79 000 010***
- A koliko je ***n^3*** i ***n* log(*n*)** ?
  - ***$(1000000)^3$ = 1 000 000 000 000 000 000***
  - ***1000000 * log(1000000) = 6 000 000***
- Upravo se zbog toga kaže za funkciju ***$T_1(n)$*** da je reda veličine ***n^3*** , dok se za funkciju ***$T_2(n)$*** kaže da je reda veličine ***n* · log(*n*)**, ali se za ***n*** podrazumeva da je to neka jako velika vrednost

… ASIMPTOTSKO VREME IZVRŠAVANJA …

- Analiza se pojednostavljuje tako što se za vreme izvršavanja nekog algoritma uzima funkcija koja za velike vrednosti njenog argumenta najbolje aproksimira tačnu funkciju vremena izvršavanja tog algoritma
- To približno vreme izvršavanja se naziva asimptotsko vreme izvršavanja
- To je mera brzine rasta tačnog vremena izvršavanja algoritma sa povećanjem broja njegovih ulaznih podataka
- Ovo pojednostavljivanje analize opravdano je iz tri razloga:
  - Brzina je važna kada je broj ulaznih podataka velik
  - Za dovoljno velike ulaze brzinom izvršavanja algoritma preovlađuje njen dominantni monom
  - Jednostavna funkcija za vreme izvršavanja se uzima bez konstanti, jer tačna vrednost konstante uz dominantan monom nije značajna za računare različite snage

… ASIMPTOTSKO VREME IZVRŠAVANJA …

- Dominantni monom preovlađuje funkcijom vremena izvršavanja, pa se zato može reći da je $T_1(n) \approx n^3$ i $T_2(n) \approx n \log(n)$,



… ASIMPTOTSKO VREME IZVRŠAVANJA …

- Pretpostaviti da imamo 3 algoritma A1, A2 i A3 čija su vremena izvršavanja data funkcijama 2^n, $5n^2$, $100n$
- Različite vrednosti funkcija za različite ulazne vrednosti n, kao i vreme izvršavanja na računaru koji izvršava 10^6 instrukcija u sekundi:

n	2^n	$5n^2$	$100n$
1	2	5	100
10	1024	500	1000
100	2^{100}	5000	10000
1000	2^{1000}	$5 \cdot 10^6$	100000

n	A1	A2	A3
1	1μs	5μs	100μs
10	1ms	0,5ms	1ms
100	2^{70}g	0,05s	0,01s
1000	2^{970}g	5s	0,1s

- Vidi se da asimptotsko vreme dovoljno dobro procenjuje složenost
- Dovoljno je reći da je asimptotsko vreme: 2^n, n^2, n
- Uvode se specifične asimptotske notacije

… ASIMPTOTSKO VREME IZVRŠAVANJA …

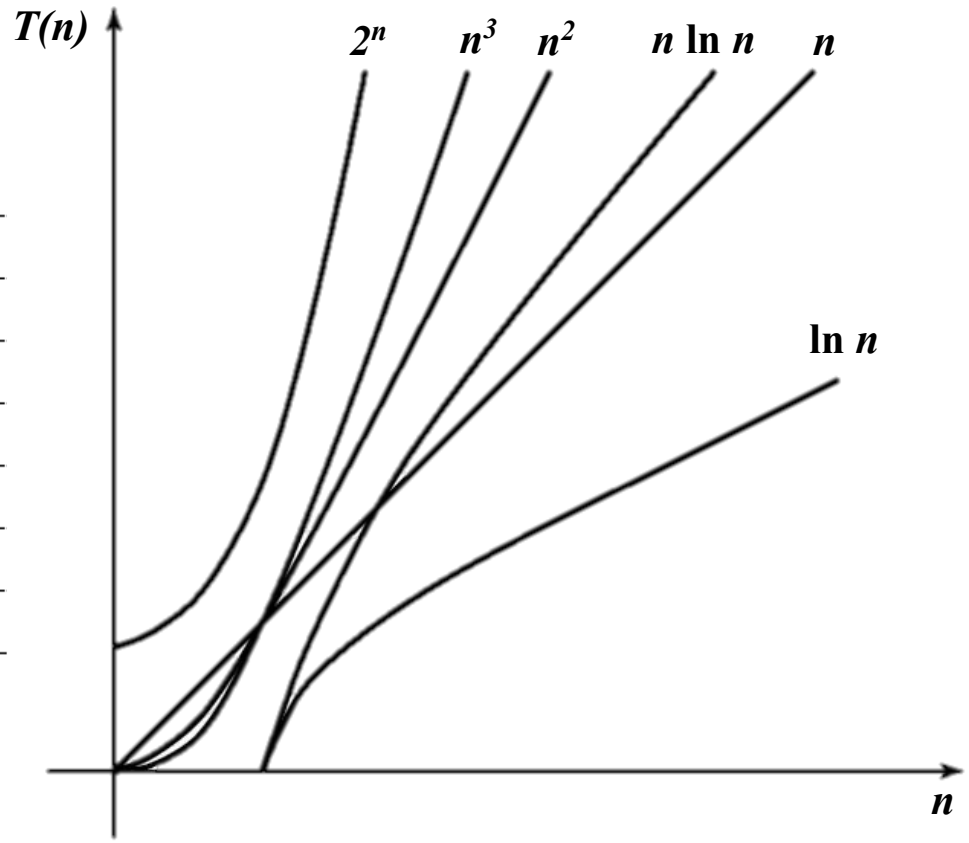
- Pojednostavljenom analizom algoritma dobija se nekoliko standardnih funkcija koje najčešće opisuju vremena izvršavanja algoritma
- Dobro je znati ove funkcije, jer olakšavaju analizu kompleksnosti algoritma (brži i lakši uvid u performanse algoritma)
- Prema brzini rasta, poređane od najbržih ka najsporijim:

Funkcija	Neformalno ime
1	Konstantna funkcija
$\log n$	Logaritamska funkcija
$\sqrt{n}$	Korenska funkcija
n	Linearna funkcija
$n \log n$	Linearno-logaritamska funkcija
n^2	Kvadratna funkcija
n^3	Kubna funkcija
2^{n^ϵ}	Subeksponencijalna funkcija
2^n	Eksponencijalna funkcija
$n!$	Faktorijal

… ASIMPTOTSKO VREME IZVRŠAVANJA

- Brzina algoritma je bolja ukoliko se njegova funkcija vremena izvršavanja nalazi brže vrhu tabele
- Algoritam je brži ukoliko je njegova asimptotska funkcija rasta manja, tj. ako je stepen rasta sporiji

Funkcija	Neformalno ime
1	Konstantna funkcija
$\log n$	Logaritamska funkcija
n	Linearna funkcija
$n \log n$	Linearno-logaritamska funkcija
n^2	Kvadratna funkcija
n^3	Kubna funkcija
2^n	Eksponencijalna funkcija



ASIMPTOTSKE NOTACIJE

ASIMPTOTSKE NOTACIJE …

- Algoritme upoređujemo prema njihovim funkcijama asimptotskog vremena izvršavanja
- Što je asimptotska funkcija manja, algoritam je brži
- Za mnogo slučajeve je jasno kada je funkcija manja od druge
- Za složenije slučajeve je potrebno imati preciznu definiciju toga šta se podrazumeva kada se tvrdi da je neka funkcija manja od druge
- Asimptotska notacija omogućava da se na **precizan način** uvede **relativni poredak za funkcije**
- Asimptotske notacije se koriste da upravo označe na koje se vreme odnose (npr. **worste-case**) ali i na izvršavanje u bilo kom vremenu
- Opisuju se i definišu „suvom“ matematikom

… ASIMPTOTSKE NOTACIJE

- Postoji nekoliko asimptotskih notacija:
  - ***O***-notacija (veliko o)
  - ***Θ***-notacija (veliko teta)
  - **Ω**-notacija (veliko omega)
  - ***o***-notacija (malo o)
  - ***ω***-notacija (malo omega)
- U praksi se najviše koristi ***O***-notacija, jer ju je najlakše dokazati i odrediti
- Za sve funkcije u asimptotskim notacijama važi da su pozitivne, nenegativne funkcije, što važi i za njihove argumente
  - Posmatra se skup funkcija koje preslikavaju skup prirodnih brojeva u skup nenegativnih realnih brojeva
- Zašto?

O-NOTACIJA …

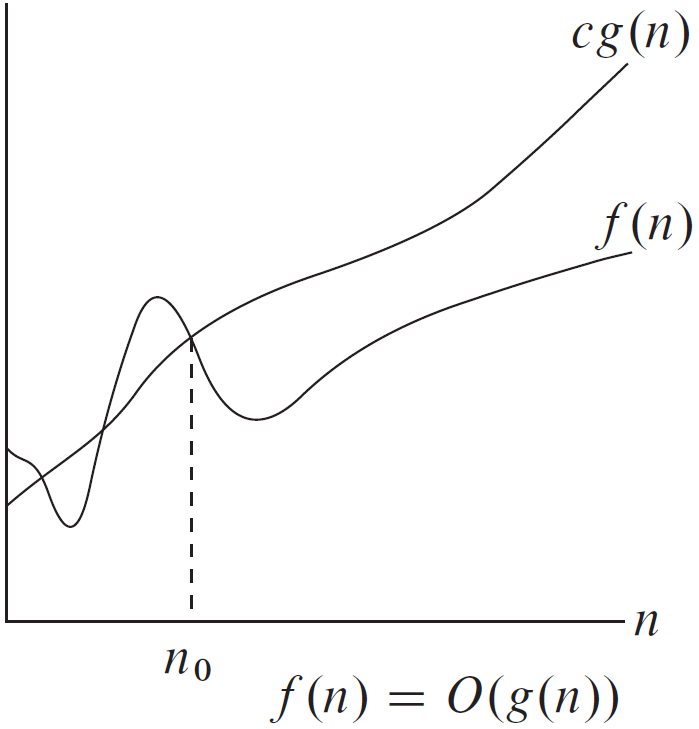
- Koristi se za precizno definisanje svojstva da je neka funkcija manja od druge
- Ako za 2 funkcije ***f*** i ***g*** kažemo da je ***f*** manje od ***g***, onda se intuitivno misli da je ***f(n) ≤ g(n)***, za sve vrednosti argumenta ***n***
- Zahtev nije toliko strog, tj. ne mora biti za sve vrednosti od ***n***, za asimptotsko ponašanje funkcije, dovoljno je da reći da je ***f(n) ≤ g(n)***, za sve osim za konačno mnogo vrednosti argumenta ***n***
- Zašto?
- Nas zanima rast funkcije, tj. kako se ona ponaša kada raste ***n***, tako da u početku funkcija ***f*** ne mora biti manja od ***g***
- ***O***-notacija, ***f(n) = O(g(n))*** upravo predstavlja zapis kojim se predstavlja svojstvo da je funkcija ***f*** manja od funkcije ***g***, tj. da funkcija ***g*** dominira nad funkcijom ***f***

… *O*-NOTACIJA …

- Definicija:

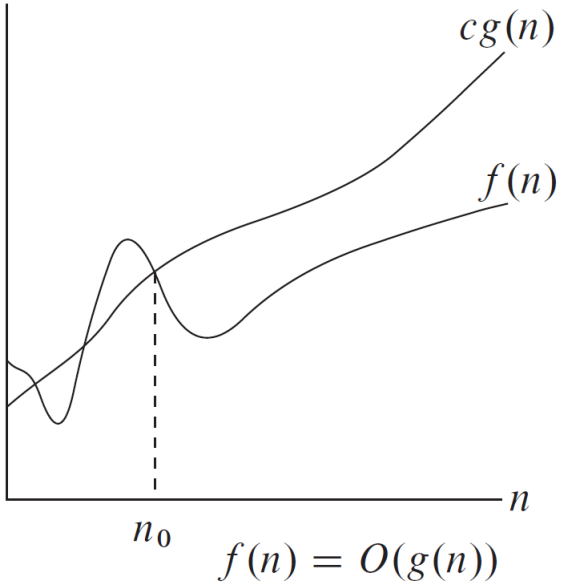
  Za dve nenegativne funkcije ***f***, ***g: N→R*⁺** kažemo da je ***f(n) = O(g(n))*** ako postoje pozitivne konstante ***c*** i n_0 takve da je ***f(n) ≤ c · g(n)***, za svako $n \geq n_0$

- Zapis ***f(n) = O(g(n))*** formalno znači da postoji neka tačka n_0 takva da za sve vrednosti ***n*** od te tačke, funkcija ***f(n)*** je ograničena vrednošću proizvoda neke pozitivne konstante ***c*** i funkcije ***g(n)***
- Drugim rečima, ***O***-notacija označava da funkcija ***g(n)*** predstavlja asimptotsku gornju granicu za funkciju ***f(n)***



… *O*-NOTACIJA …

- Šta to zapravo znači?
- Neka funkcija ***f(n)*** predstavlja vreme izvršavanja ***T(n)*** nekog algoritma
- Neka je $g(n) = n^2$
- Ako se dokaže da je $T(n) = O(n^2)$, onda je sigurno pokazano (zanemarujući konstante) da je vreme izvršavanja algoritma ograničeno kvadratnom funkcijom



- ***O***-notacijom se dobija samo gornja granica vremena izvršavanja
- Što je sasvim dovoljno kada se određuje **worst-case** slučaj
- Pravo vreme rada algoritma ograničava se najgorim vremenom njegovog izvršavanja

… *O*-NOTACIJA …

- U konkretnom slučaju, za konkretnu funkciju se ***T(n) = O(g(n))*** može pokazati praćenjem formalne definicije ***O***-notacije pronalaženjem konkretnih vrednosti konstanti $n_0 > 0$ i $c > 0$ takve da je zadovoljeno $T(n) < c \cdot g(n)$ za svako $n > n_0$
- Na primer:

$$\mathbf{T}(n) = n^2 - n + 1$$

- Za $n_0 = 1$ i za $c = 2$ dobijamo:

$$n^2 - n + 1 \leq n^2 + 1 \leq n^2 + n^2 = 2n^2$$

- Po definiciji se može zaključiti da je $T(n) = O(n^2)$
- Da li je to jedina kombinacija konstanti n_0 i c ?

… *O*-NOTACIJA …

- Praktičan postupak dobijanja dominirajuće funkcije se može značajno ubrzati ukoliko se iskoriste opšte osobine ***O***-notacije
  - Konstantni faktori nisu važni
  - Dominantni monomi su najvažniji
  - Pravilo sabiranja
  - Pravilo množenja
  - Pravilo tranzitivnosti
- Poznavajući ovo može se, najčešće, relativno lako odrediti dominirajuća funkcija vremena izvršavanja jednostavnih algoritama

… O-NOTACIJA …

- **Konstantni faktori nisu važni:**

  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{c} \cdot \boldsymbol{T}(\boldsymbol{n}))$

  za svako $\boldsymbol{c} > \boldsymbol{0}$.

  Tako da ako $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{n}^{\boldsymbol{2}})$, onda može biti i
  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\mathbf{3} \cdot \boldsymbol{n}^{\boldsymbol{2}})$ ili čak
  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\mathbf{0,001} \cdot \boldsymbol{n}^{\boldsymbol{2}})$

- **Dominantni monomi su najvažniji:**

  Ako je $\boldsymbol{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}) + \boldsymbol{g}(\boldsymbol{n}))$ i $\mathbf{g}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))$ onda je
  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{f}(\boldsymbol{n}))$

  Na primer

  Ako je $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{n}^{\boldsymbol{2}} - \boldsymbol{n} + \mathbf{1}$ onda je $\boldsymbol{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{n}^{\boldsymbol{2}})$, jer je
  $\boldsymbol{n}^{\boldsymbol{2}} - \boldsymbol{n} = \boldsymbol{O}(\boldsymbol{n}^{\boldsymbol{2}})$ i $\mathbf{1} = \boldsymbol{O}(\boldsymbol{n}^{\boldsymbol{2}})$

… O-NOTACIJA …

- **Pravilo sabiranja:**

  $$\boldsymbol{O(f(n) + g(n)) = O\big(f(n)\big) + O(g(n))}$$

  Npr.

  Ako je $\boldsymbol{T(n) = O(n) + O(17)}$, onda je $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O(n + 17)}$, odnosno $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O(n)}$ na osnovu prethodnog pravila

- **Pravilo množenja:**

  $$\boldsymbol{O(f(n) \cdot g(n)) = O\big(f(n)\big) \cdot O(g(n))}$$

  Npr.

  Ako je $\boldsymbol{T(n) = n \cdot O(1)}$, onda je $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{n \cdot O(1) = O(n) \cdot O(1) = O(n)}$

… O-NOTACIJA …

- **Pravilo tranzitivnosti:**

  Ako je $\boldsymbol{T(n) = O(f(n))}$ i $\boldsymbol{f(n) = O(g(n))}$ onda je i $\boldsymbol{T(n) = O(g(n))}$

  Npr.

  Ako je $\boldsymbol{T(n) = O(n^2)}$,
  kako je očigledno $\boldsymbol{n^2 = O(n^3)}$,
  onda je i $\boldsymbol{T(n) = O(n^3)}$

… *O*-NOTACIJA …

- Za malo složenije algoritme potrebno je poznavati i relativne odnose nekih osnovnih funkcija:
  - Izraz n^b dominira nad izrazom n^a za $b > a$

    na primer: n^3 dominira nad n^2, tj. $n^2 = O(n^3)$
  - Svaka eksponencijalna funkcija dominira nad svakom polinomskom funkcijom

    na primer: n^5 dominira nad n^2, ali 2^n dominira i nad n^5 i nad n^2, tj. $n^2 = O(n^5) = O(2^n)$
  - Svaka polinomska funkcija dominira nad svakom logaritamskom funkcijom

    na primer: $(\log n)^3 = O(n)$

    Odavde sledi da je: $n \cdot \log n = n \cdot O(n) = O(n^2)$

… *O*-NOTACIJA …

- Ko je pažljivo pratio, došao je do zaključka da je ***O***-notacija …

  pomalo dvosmislena

- Zašto?
- Kako izražavamo vreme izvršavanja nekog algoritma?
- Ako je vreme izvršavanja nekog algoritma:

  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{n}^{\mathbf{2}} - \boldsymbol{n} + \mathbf{1} = \boldsymbol{O}(\boldsymbol{n}^{\mathbf{2}})$

- Šta još možemo reći?
- Pravilom tranzitivnosti možemo reći i da je:

  $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{n}^{\mathbf{3}})$, ili čak $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{n}^{\mathbf{10}})$

- **Ali da li to oslikava pravu složenost ?**

… *O*-NOTACIJA …

- Prirodno je tražiti najbolju (najbližu) dominirajuću funkciju za vreme izvršavanja
- Nema smisla složenost funkcije $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{n}^{\mathbf{2}} - \boldsymbol{n} + \mathbf{1}$ oceniti sa $\boldsymbol{O}(\boldsymbol{n}^{\mathbf{3}})$ iako je tehnički to tačno
- Ako se primeni i pravilo da konstante nisu važne, onda se bez kraja mogu navoditi sve bliže i bliže granice,
  na primer: $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\mathbf{0.01} \cdot \boldsymbol{n}^{\mathbf{2}})$ ili $\mathbf{T}(\boldsymbol{n}) = \boldsymbol{O}(\mathbf{0.0001} \cdot \boldsymbol{n}^{\mathbf{2}})$
- Po pravilu, kada god je to moguće, uvek se koristi ***O***-notacija koja je najprostija
- To znači da se iskazuje jednim monomom uz koji stoji konstantni faktor vrednosti 1
- Jednostavnost i najbolja ocena nisu uvek ostvarivi zajedno, ali je to u praksi retko

… *O*-NOTACIJA

- Originalna definicija sa početka priče o ***O***-notaciji se mora modifikovati da bi bila matematički ispravna, jer **=** nije simetrična kada se koristi ***O***-notacija
- Naime,

  $\boldsymbol{f_1(n) = O}(\text{g(n)})$ i $\boldsymbol{f_2(n) = O}(\text{g(n)})$

  ne implicira i

  $\boldsymbol{f_1(n) = f_2(n)}$

- Zato se, u striktnom smislu, ne govori o funkciji, već o klasi funkcija koje zadovoljavaju određenu formu:

  $$\boldsymbol{O(g(n)) = \{f: N \to R^+ | (\exists c > 0)(\exists n_0 \in N)(\forall n \geq n_0) f(n) \leq c \cdot g(n)\}}$$

- Međutim, zbog jednostavnosti se nauštrb preciznosti tradicionalno više koristi prva, jednostavnija, definicija

Ω-NOTACIJA …

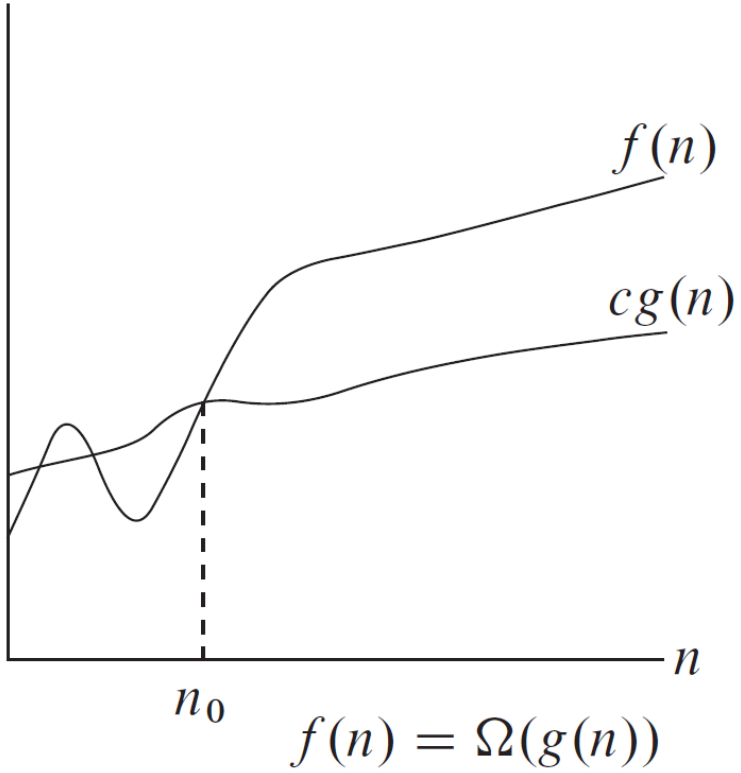
- Veoma slična ***O***-notaciji, ali sa suprotne strane
- Koristi se za precizno definisanje svojstva da je neka funkcija veća od neke druge
- Ako za 2 funkcije ***f*** i ***g*** kažemo da je ***f*** veća od ***g***, onda se intuitivno misli da je $f(n) \geq g(n)$, za sve ili neke vrednosti argumenta ***n***
- Ni ovde zahtev nije toliko strog, tj. ne mora biti za sve vrednosti od ***n***, za asimptotsko ponašanje funkcije, dovoljno je da reći da je $f(n) \geq g(n)$, za sve osim za konačno mnogo vrednosti argumenta ***n***
- **Ω**-notacija, $f(n) = \Omega(g(n))$ predstavlja zapis kojim se predstavlja svojstvo da je funkcija ***f*** veća od funkcije ***g***, tj. da funkcija ***f*** dominira nad funkcijom ***g***
- Koristi se da označi da se neki algoritam mora izvršiti bar za neko određeno vreme, **best-case** scenario, vreme izvršavanja algoritma nikada neće imati bolji rast od **Ω**-zapisane funkcije

… Ω-NOTACIJA …

- Definicija:

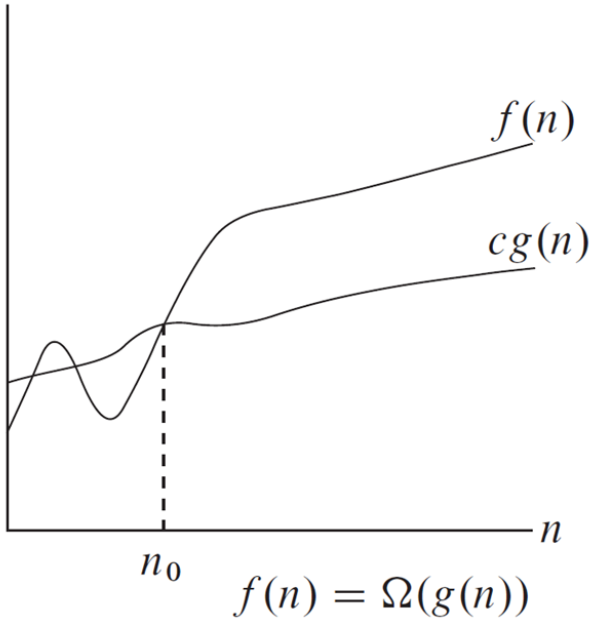
  Za dve nenegativne funkcije ***f***, ***g: N→R***$^{+}$ kažemo da je ***f(n) = Ω(g(n))*** ako postoje pozitivne konstante ***c*** i n_0 takve da je ***f(n) ≥ c · g(n)***, za svako ***n ≥*** n_0

- Zapis ***f(n) = Ω(g(n))*** formalno znači da postoji neka tačka n_0 takva da za sve vrednosti ***n*** od te tačke, funkcija ***f(n)*** ograničava vrednost proizvoda neke pozitivne konstante ***c*** i funkcije ***g(n)***
- Drugim rečima, **Ω**-notacija označava da funkcija ***g(n)*** predstavlja asimptotsku donju granicu za funkciju ***f(n)***



… Ω-NOTACIJA …

- Šta **Ω**-notacija zapravo znači?
- Ako se kaže da je vreme izvršavanja algoritma ***Ω(g(n))*** to znači da bez obzira koliko je velik ulaz, za bilo koju vrednost ***n***, vreme izvršavanja za taj ulaz je minimum ***g(n)*** pomnoženo sa nekom konstantom, čak i za dovoljno veliko ***n***
- Takođe, daje se i donja granica vremena za **best-case** scenario (bolje od ovoga ne može!)



- Setite se algoritma za INSERTION sortiranje
- Koje je vreme izvršavanja za **worste-case** scenario?
- Koje je vreme izvršavanja za **best-case** scenario?

… Ω-NOTACIJA …

- Setite se algoritma za INSERTION sortiranje

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n}(t_j-1) + c_7\sum_{j=2}^{n}(t_j-1) + c_8(n-1)\,.$$

… Ω-NOTACIJA …

- Setite se algoritma za INSERTION sortiranje
- Koji je **best-case** scenario?
  - Niz je već sortiran
  - $\boldsymbol{T(n) = a \cdot n + b = n}$
  - Da li je to možda **Ω** ?
  - $\boldsymbol{T(n) = \Omega(n)}$

```
INSERTION-SORT(A)                                   cost    times
1  for j = 2 to A.length                            c1      n
2      key = A[j]                                   c2      n − 1
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].                  0       n − 1
4      i = j − 1                                    c4      n − 1
5      while i > 0 and A[i] > key                   c5      ∑_{j=2}^{n} t_j
6          A[i + 1] = A[i]                          c6      ∑_{j=2}^{n} (t_j − 1)
7          i = i − 1                                c7      ∑_{j=2}^{n} (t_j − 1)
8      A[i + 1] = key                               c8      n − 1
```

- Koji **worste-case** scenario
  - Niz je sortiran, ali obrnuto od toga što želimo
  - $\boldsymbol{T(n) = a \cdot n^2 + b \cdot n + c = n^2}$
  - Da li je to možda **O** ?
  - $\boldsymbol{T(n) = O(n^2)}$

… Ω-NOTACIJA

- Kao i kod ***O***-notacije originalna definicija sa početka priče o **Ω**-notacije se mora modifikovati da bi bila matematički ispravna, jer **=** nije simetrično kada se koristi **Ω** -notacija
- Da podsetim,

  $\boldsymbol{f_1(n) = \Omega(g(n))}$ i $\boldsymbol{f_2(n) = \Omega(g(n))}$

  ne implicira i

  $\boldsymbol{f_1(n) = f_2(n)}$

- Zato se, u striktnom smislu, ne govori o funkciji, već o klasi funkcija koje zadovoljavaju određenu formu:

$$\boldsymbol{\Omega(g(n)) = \{f: N \to R^+ | (\exists c > 0)(\exists n_0 \in N)(\forall n \geq n_0) f(n) \geq c \cdot g(n)\}}$$

Θ-NOTACIJA …

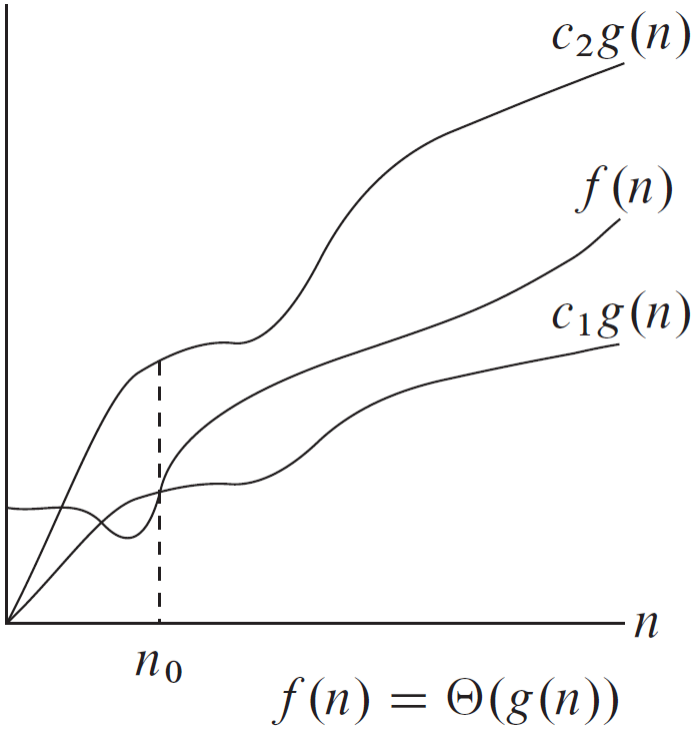
- ***Θ***-notacija predstavlja asimptotski najbolju ocenu za vreme izvršavanja, tj. za funkciju ***f(n)***
- Kada se ona koristi u analizi algoritama, ne daje se samo gornja granica izvršavanja (ili samo donja), već i njegova donja granica, to znači da je vreme izvršavana najbolje moguće ocenjeno
- Standardno se daleko manje koristi od ***O***-notacije, jer je daleko teže odrediti i donju i gornju granicu (daleko lakše je odrediti samo gornju granicu)

… Θ-NOTACIJA …

- Definicija:

  Za dve nenegativne funkcije ***f***, ***g: N→R⁺*** kažemo da je ***f(n) = Θ(g(n))*** ako postoje pozitivne konstante c_1, c_2 i n_0 takve da je za svako $n \geq n_0$
  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

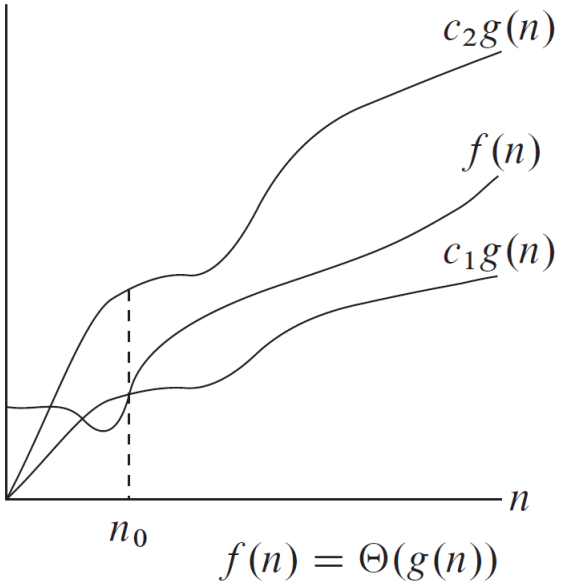
- Zapis ***f(n) = Θ(g(n))*** formalno znači da postoji neka tačka n_0 takva da za sve vrednosti ***n*** od te tačke, funkcija ***f(n)*** ograničava vrednost proizvoda neke pozitivne konstante c_1 i funkcije ***g(n)***, ali u isto vreme je ograničena vrednošću proizvoda neke pozitivne konstante c_2 i funkcije ***g(n)***



- Drugim rečima, ***Θ***-notacija označava da funkcija ***g(n)*** predstavlja strogu asimptotsku granicu (i gornju i donju) za funkciju ***f(n)***

… Θ-NOTACIJA …

- Šta to zapravo znači?
- Neka funkcija ***f(n)*** predstavlja vreme izvršavanja ***T(n)*** nekog algoritma
- Neka je $g(n) = n^2$
- Ako se dokaže da je $T(n) = \Theta(n^2)$, onda je sigurno pokazano (zanemarujući konstante) da je vreme izvršavanja algoritma strogo ograničeno kvadratnom funkcijom i sa gornje i sa donje strane



- Ako se kaže da je vreme izvršavanja algoritma $\Theta(g(n))$ to znači da bez obzira koliko je velik ulaz, za bilo koju vrednost $n \geq n_0$, vreme izvršavanja za taj ulaz je minimum ***g(n)*** pomnoženo sa nekom konstantom c_1, odnosno vreme izvršavanja za taj ulaz je maksimalno ***g(n)*** pomnoženo sa nekom konstantom c_2 za dovoljno veliko ***n***

… Θ-NOTACIJA …

- Opšte osobine ***O***-notacije važe i za ***Θ***-notaciju (kao i za **Ω**-notaciju), što značajno ubrzava praktičan postupak dobijanja ograničavajuće funkcije
  - Konstantni faktori nisu važni
  - Dominantni monomi su najvažniji
  - Pravilo sabiranja
  - Pravilo množenja
  - Pravilo tranzitivnosti
- Poznavajući ovo može se, najčešće, relativno lako odrediti ograničavajuća funkcija vremena izvršavanja jednostavnih algoritama

… Θ-NOTACIJA …

- Na šta se svodi određivanje ***Θ***-notacije?
- Na primer: da bi dokazali da je $\frac{1}{2}n^2 - 3n = \Theta(n^2)$, treba pronaći konstante c_1, c_2 i n_0 takve da je za svako $n \geq n_0$:

$$c_1 \cdot n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 \cdot n^2$$

- Deljenjem sa n^2 dobijamo:

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

- Desna strana nejednakosti postaje tačna $n \geq 1$ i $c_2 = 1/2$
- Leva strana nejednakosti postaje tačna za $n \geq 7$ i $c_1 = 1/4$
- Za $n_0 = 7$, $c_1 = 1/4$ i $c_2 = 1/2$, pokazali smo da je

$$\frac{1}{2}n^2 - 3n = \Theta(n^2)$$

… Θ-NOTACIJA …

- Za $n_0 = 7$, $c_1 = 1 / 14$ i $c_2 = 1 / 2$,

  pokazali smo da je

  $$\frac{1}{2}n^2 - 3n = \Theta(n^2)$$

- Naravno, postoje i druge vrednosti za koje je

  $$\frac{1}{2}n^2 - 3n = \Theta(n^2),$$

  ali važno je da neke vrednosti postoje (makar samo jedna kombinacija)

- Ove vrednosti umnogome zavise i od vrste konstanti i monoma koji čine ***T(n)***,

  druga funkcija koju određuje ***Θ(n)*** bi zahtevala drugačije konstante

… Θ-NOTACIJA …

- Na sličan način se može dokazati i

  $$6n^3 \neq \Theta(n^2)$$

- Pretpostaviti da se mogu pronaći konstante c_2 i n_0 takve da je za svako $n \geq n_0$:

  $$6n^3 \leq c_2 \cdot n^2$$

- Deljenjem sa n^2 dobijamo:

  $$n \leq \frac{c_2}{6}$$

  što nikako ne može biti tačno za promenljivo n (koje konstantno raste)

… Θ-NOTACIJA …

- Kao i kod ***O***-notacije originalna definicija sa početka priče o ***Θ***-notaciji se mora modifikovati da bi bila matematički ispravna, jer **=** nije simetrična kada se koristi ***Θ***-notacija
- Da podsetim,

  $\boldsymbol{f_1(n) = \Theta(g(n))}$ i $\boldsymbol{f_2(n) = \Theta(g(n))}$

  ne implicira i

  $\boldsymbol{f_1(n) = f_2(n)}$

- Zato se, u striktnom smislu, ne govori o funkciji, već o klasi funkcija koje zadovoljavaju određenu formu:

$$\Theta(g(n)) = \left\{ \begin{matrix} f: N \to R^+ | (\exists c_1 > 0)(\exists c_2 > 0)(\exists n_0 \in N)(\forall n \geq n_0) \\ c_2 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \end{matrix} \right\}$$

… Θ-NOTACIJA

- Teorema:

  Za dve nenegativne funkcije ***f***, ***g: N→R***$^{+}$ kažemo da je ***f(n) = Θ(g(n))*** ako i samo ako je ***f(n) = O(g(n))*** i ***f(n) = Ω(g(n))***

- **Dokažite!**

JEDNAKOST I JEDNAČINE ASIMPT. NOTACIJA …

- Jednakost $\boldsymbol{f(n) = O(n^2)}$ nije strogo tačna, već $\boldsymbol{f(n) \in O(n^2)}$
- **Zašto?**
- Ali šta ako je asimptotska notacija deo neke formule?
- Na primer: $\boldsymbol{2n^2 + 3n + 1 = 2\,n^2 + \Theta(n)}$?
- Uopšteno, kada se asimptotska notacija nađe u formuli, ona se interpretira kao neka asimptotska funkcija koju nije potrebno imenovati
  - Znači može biti bilo koja funkcija iz skupa $\boldsymbol{\Theta(n)}$
- To znači da se $\boldsymbol{2n^2 + 3n + 1 = 2\,n^2 + \Theta(n)}$ može tumačiti kao $\boldsymbol{2n^2 + 3n + 1 = 2\,n^2 + f(n)}$, gde je $\boldsymbol{f(n) \in \Theta(n)}$ što u ovom slučaju može biti i $\boldsymbol{f(n) = 3n + 1}$, a to zaista jeste $\boldsymbol{\Theta(n)}$

… JEDNAKOST I JEDNAČINE ASIMPT. NOTACIJA …

- Prateći logiku sa prethodnog slajda moguće je napisati da je vreme izvršavanja nekog algoritma (npr. MERGE sortiranja):

$$\boldsymbol{T(n) = 2T(\frac{n}{2}) + \Theta(n)}$$

to znači, da pošto nas interesuje samo asimptotsko ponašanje funkcije ***T(n)*** nema potrebe da se definiše funkcija nižeg stepena na tačan način, one su sve uključene u anonimnoj funkciji $\Theta(\boldsymbol{n})$ koja predstavlja čitavu klasu tih funkcija (i koja definiše kompleksnost svih tih funkcija)

… JEDNAKOST I JEDNAČINE ASIMPT. NOTACIJA

- U nekim slučajevima asimptotska notacija se može naći i sa leve strane jednačine
- Na primer: $\boldsymbol{2n^2 + \Theta(n) = \Theta(n^2)}$
- Ovo se tumači da nije važno koja anonimna funkcija je odabrana na levoj strani jednačine, postoji način da se odabere anonimna funkcija sa desne strane istog znaka koja će učiniti jednačinu validnom

- Može se napraviti niz takvih izraza u jednačini:

  $\boldsymbol{2n^2 + 3n + 1 = 2n^2 + \Theta(n) = \Theta(n^2)}$

  Prva jednačina kaže da postoji funkcija $\boldsymbol{f(n) \in \Theta(n)}$ takva da je $\boldsymbol{2n^2 + 3n + 1 = 2n^2 + f(n)}$ za svako $\boldsymbol{n}$

  Druga jednačina kaže da za bilo koju funkciju postoji $\boldsymbol{g(n) \in \Theta(n)}$ postoji funkcija $\boldsymbol{h(n) \in \Theta(n^2)}$ takva da je $\boldsymbol{2n^2 + g(n) = h(n)}$ za svako $\boldsymbol{n}$

  Ovo ujedno implicira i $\boldsymbol{2n^2 + 3n + 1 = \Theta(n^2)}$

o-NOTACIJA …

- ***O***-notacija može ali i ne mora biti čvrsta (*tight*) asimptotska gornja granica (sve ono o čemu smo već diskutovali)
- Veza $2n^2 = O(n^2)$ je asimptotski čvrsta gornja granica (neposredno iznad), ali veza $2n = O(n^2)$ nije asimptotski čvrsta gornja granica
- Za asimptotske gornje granice funkcija koje su po prirodi čvrste koristi se ***o***-notacija
- Definicija:

  Za dve nenegativne funkcije ***f***, ***g: N→R***$^+$ kažemo da je ***f(n) = o(g(n))*** ako za svaku pozitivnu konstantu ***c*** postoji konstanta n_0 takva da je $f(n) \leq c \cdot g(n)$ za svako $n \geq n_0$

- Zapis ***f(n) = o(g(n))*** označava da je brzina rasta funkcije ***f(n)*** reda veličine manja od brzine rasta funkcije ***g(n)***
- To znači: $2n^2 \neq o(n^2)$, ali zato $2n = o(n^2)$

… o-NOTACIJA

- ***O***-notacija i ***o***-notacija su slične, ali dok za ***f(n) = O(g(n))*** veza ***f(n) ≤ c · g(n)*** važi samo za neke konstante ***c > 0***, za ***f(n) = o(g(n))*** to važi za svako ***c > 0***
- Vrednost funkcije ***f(n)*** postaje mnogo manja (gotovo beznačajna) u poređenju sa funkcijom ***g(n)*** kako ***n*** teži beskonačnosti, odnosno

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

- U striktnom smislu, kao i kod ***O***-notacije ne govori se o funkciji, već o klasi funkcija koje zadovoljavaju određenu formu:

$$o(g(n)) = \left\{ \begin{matrix} f: N \to R^{+} | (\exists c \in N)(\exists n_0 \in N)(\forall c \wedge \forall n \geq n_0) \\ f(n) \leq c \cdot g(n) \end{matrix} \right\}$$

ω-NOTACIJA …

- ***ω***-notacija je za **Ω**-notaciju isto što i ***o***-notacija za ***O***-notaciju
- Koristi se za označavanje asimptotske donje granice koja je čvrsta
- Definicija:

  Za dve nenegativne funkcije ***f***, ***g: N→R*⁺** kažemo da je ***f(n) = ω(g(n))*** ako za svaku pozitivnu konstantu ***c*** postoji konstanta n_0 takva da je $f(n) \geq c \cdot g(n)$ za svako $n \geq n_0$

- Zapis ***f(n) = ω(g(n))*** označava da je brzina rasta funkcije ***f(n)*** reda veličine veća od brzine rasta funkcije ***g(n)***
- To znači: $n^2/2 = \omega(n)$, ali zato $n^2/2 \neq \omega(n^2)$

… ω-NOTACIJA

- ***ω***-notacija i **Ω**-notacija su slične, ali dok za ***f(n) = Ω(g(n))*** veza ***f(n) ≥ c · g(n)*** važi samo za neke konstante ***c > 0***, za ***f(n) = ω(g(n))*** to važi za svako ***c > 0***
- Vrednost funkcije ***g(n)*** postaje mnogo manja (gotovo beznačajna) u poređenju sa funkcijom ***f(n)*** kako ***n*** teži beskonačnosti, odnosno

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

- U striktnom smislu, kao i kod **Ω**-notacije ne govori se o funkciji, već o klasi funkcija koje zadovoljavaju određenu formu:

$$\omega(g(n)) = \left\{ \begin{matrix} f: N \to R^{+} | (\exists c \in N)(\exists n_0 \in N)(\forall c \wedge \forall n \geq n_0) \\ f(n) \geq c \cdot g(n) \end{matrix} \right\}$$

POREĐENJE ASIMPTOTSKIH FUNKCIJA …

- Kao što su neke već pokazane, mnoge osobine realnih brojeva mogu se primeniti i na asimptotsko poređenje
- Pretpostaviti da su funkcije asimptotski pozitivne
- Onda važi
  - Tranzitivnost
  - Refleksivnost
  - Simetričnost
  - Komutativnost

- Jedino što ne važi je pravilo da za svaka dva realna broja mora biti tačno samo jedno od sledeća tri tvrđenje: $a < b$, $a = b$, $a > b$

  Zašto?

  Zato što se ne mogu sve funkcije asimptotski porediti

… POREĐENJE ASIMPTOTSKIH FUNKCIJA …

- Tranzitivnost:

$$\boldsymbol{f(n) = \Theta(g(n)) \wedge g(n) = \Theta\big(h(n)\big) \rightarrow f(n) = \Theta(h(n))}$$

$$\boldsymbol{f(n) = O(g(n)) \wedge g(n) = O\big(h(n)\big) \rightarrow f(n) = O(h(n))}$$

$$\boldsymbol{f(n) = \Omega(g(n)) \wedge g(n) = \Omega\big(h(n)\big) \rightarrow f(n) = \Omega(h(n))}$$

$$\boldsymbol{f(n) = o(g(n)) \wedge g(n) = o\big(h(n)\big) \rightarrow f(n) = o(h(n))}$$

$$\boldsymbol{f(n) = \omega(g(n)) \wedge g(n) = \omega\big(h(n)\big) \rightarrow f(n) = \omega(h(n))}$$

- Refleksivnost:

$$\boldsymbol{f(n) = \Theta(f(n))}$$

$$\boldsymbol{f(n) = O(f(n))}$$

$$\boldsymbol{f(n) = \Omega(f(n))}$$

… POREĐENJE ASIMPTOTSKIH FUNKCIJA …

- Simetričnost:

$$\boldsymbol{f(n) = \Theta(g(n))}\ \textit{ako i samo ako je}\ \boldsymbol{g(n) = \Theta(f(n))}$$

- Transponovana simetričnost:

$$\boldsymbol{f(n) = O(g(n))}\ \textit{ako i samo ako je}\ \boldsymbol{g(n) = \Omega(f(n))}$$

$$\boldsymbol{f(n) = o(g(n))}\ \textit{ako i samo ako je}\ \boldsymbol{g(n) = \omega(f(n))}$$

… POREĐENJE ASIMPTOTSKIH FUNKCIJA

- Intuitivno značenje asimptotskih notacija je:

$f(n) = O(g(n))$	je slično	$a \leq b$
$f(n) = \Omega(g(n))$	je slično	$a \geq b$
$f(n) = \Theta(g(n))$	je slično	$a = b$
$f(n) = o(g(n))$	je slično	$a < b$
$f(n) = \omega(g(n))$	je slično	$a > b$

- Kaže se da je ***f(n)*** asimptotski manja od ***g(n)*** ako i samo ako je ***f(n) = o(g(n))***, dok se za ***f(n)*** kaže da je asimptotski veća od ***g(n)*** ako i samo ako je ***f(n) = ω(g(n))***