Uvodu u informatiku – Algoritmi i izračunljivost

Danijela Simić

algoritmi i izračunljivost

1. oktobar 2024.

Sadržaj

Uvod

Uvod

Šta sve računari mogu? Kako odgovoriti na to pitanje?

Uvod

Šta sve računari mogu? Kako odgovoriti na to pitanje?

Da bi dali odgovor moramo znati šta je to algoritam.

Definicija algoritma

Neformalna definicija

Algoritam je precizan opis postupka za rešavanje nekog problema u konačnom broju koraka.

Definicija algoritma

Neformalna definicija

Algoritam je precizan opis postupka za rešavanje nekog problema u konačnom broju koraka.

Neformalno jer: šta može biti korak; koji koracu su dopušteni, a koji nisu; neki koraci se ponavljaju, a neki granaju...

Svaki računarski program je konkretna implementacija nekog algoritma u nekom konkretnom programskom jeziku, na primer, u jeziku Java, C ili C++ i koju računar može da izvrši.

Svaki računarski program je konkretna implementacija nekog algoritma u nekom konkretnom programskom jeziku, na primer, u jeziku Java, C ili C++ i koju računar može da izvrši.

Pošto se svi podaci digitalizacijom mogu zapisati brojevima dovoljno je razmatrati algoritme za izračunavanje funkcija čiji su i argumenti i rezultujuće vrednosti prirodni brojevi.

Svaki računarski program je konkretna implementacija nekog algoritma u nekom konkretnom programskom jeziku, na primer, u jeziku Java, C ili C++ i koju računar može da izvrši.

> Pošto se svi podaci digitalizacijom mogu zapisati brojevima dovoljno je razmatrati algoritme za izračunavanje funkcija čiji su i argumenti i rezultujuće vrednosti prirodni brojevi.

Primeri funkcija: sabiranje, množenje, računanje determinante, računanje sledbenika...

Zasnivanje pojma algoritma

Algoritam možemo opisati kao bilo koji postupak koji može da se izvrši na računaru, preciznije na njegovom procesoru.

Zasnivanje pojma algoritma

UR mašine

UR mašine

UR mašina je apstraktna mašina koja ne postoji u fizičkom obliku ali predstavlja matematičku idealizaciju računara i omogućava zasnivanje pojma algoritma.

UR mašine

UR mašina je apstraktna mašina koja ne postoji u fizičkom obliku ali predstavlja matematičku idealizaciju računara i omogućava zasnivanje pojma algoritma.

Algoritam je onaj i samo onaj postupak koji može da se opiše kao program za UR mašinu.

UR mašine – svojstva

- Raspolaže beskonačnim skupom memorijskih lokacija – registara.

UR mašine – svojstva

- Raspolaže beskonačnim skupom memorijskih lokacija – registara.
- Registri su označeni prirodnim brojevima: $1, 2, 3, \ldots$.

UR mašine – svojstva

- Raspolaže beskonačnim skupom memorijskih lokacija – registara.
- Registri su označeni prirodnim brojevima: $1, 2, 3, \ldots$.
- Svaki od njih u svakom trenutku sadrži neki prirodan broj.

UR mašine – svojstva

- Raspolaže beskonačnim skupom memorijskih lokacija – registara.
- Registri su označeni prirodnim brojevima: $1, 2, 3, \ldots$.
- Svaki od njih u svakom trenutku sadrži neki prirodan broj.
- Stanje registara u nekom trenutku zovemo konfiguracija

r_1	r_2	r_3	$\ldots$

UR mašine - operacije

oznaka	naziv	efekat
$Z(m)$	nula–instrukcija	$r_m := 0$
$S(m)$	instrukcija sledbenik	$r_m := r_m + 1$
$T(m, n)$	instrukcija prenosa	$r_n := r_m$
$J(m, n, p)$	instrukcija skoka	ako je $r_m = r_n$, idi na p-tu; inače idi na sledeću instrukciju

Tabela 1: Tabela URM instrukcija

URM program

URM program P je konačan numerisan niz URM instrukcija.

- Instrukcije se izršavaju redom (počevši od prve), osim u slučaju instrukcije skoka.

URM program

URM program P je konačan numerisan niz URM instrukcija.

- Instrukcije se izršavaju redom (počevši od prve), osim u slučaju instrukcije skoka.
- Izvršavanje programa se zaustavlja onda kada ne postoji instrukcija koju treba izvršiti.

URM program

URM program P je konačan numerisan niz URM instrukcija.

- Instrukcije se izršavaju redom (počevši od prve), osim u slučaju instrukcije skoka.
- Izvršavanje programa se zaustavlja onda kada ne postoji instrukcija koju treba izvršiti.
- Početnu konfiguraciju čini niz prirodnih brojeva $a_1, a_2, \ldots$ koji su upisani u registre od početnog redom.

—

Ako je funkcija koju treba izračunati $f(x_1, x_2, \ldots, x_n)$, onda se podrazumeva da su vrednosti $x_1, x_2, \ldots, x_n$ redom smeštene u prvih n registara.

URM program

URM program P je konačan numerisan niz URM instrukcija.

- Instrukcije se izršavaju redom (počevši od prve), osim u slučaju instrukcije skoka.
- Izvršavanje programa se zaustavlja onda kada ne postoji instrukcija koju treba izvršiti.
- Početnu konfiguraciju čini niz prirodnih brojeva $a_1, a_2, \ldots$ koji su upisani u registre od početnog redom.

  —

  Ako je funkcija koju treba izračunati $f(x_1, x_2, \ldots, x_n)$, onda se podrazumeva da su vrednosti $x_1, x_2, \ldots, x_n$ redom smeštene u prvih n registara.
- Na kraju rada programa, rezultat treba da bude smešten u prvi registar.

URM program

Ako URM program P za početnu konfiguraciju $a_1, a_2, \ldots, a_n$ ne staje sa radom, onda pišemo $P(a_1, a_2, \ldots, a_n) \uparrow$.

URM program

Ako URM program P za početnu konfiguraciju $a_1, a_2, \ldots, a_n$ ne staje sa radom, onda pišemo $P(a_1, a_2, \ldots, a_n) \uparrow$.

Ako program staje sa radom i u prvom registru je, kao rezultat, vrednost b, onda pišemo $P(a_1, a_2, \ldots, a_n) \downarrow b$.

URM izračunljivost

URM izračunljivost

Funkcija je URM- *izračunljiva* ako postoji URM program koji je izračunava.

URM program – primer1

$f(x, y) = x + y$

$$x + y = x + \underbrace{1 + 1 + \cdots + 1}_{y}$$

URM program – primer1

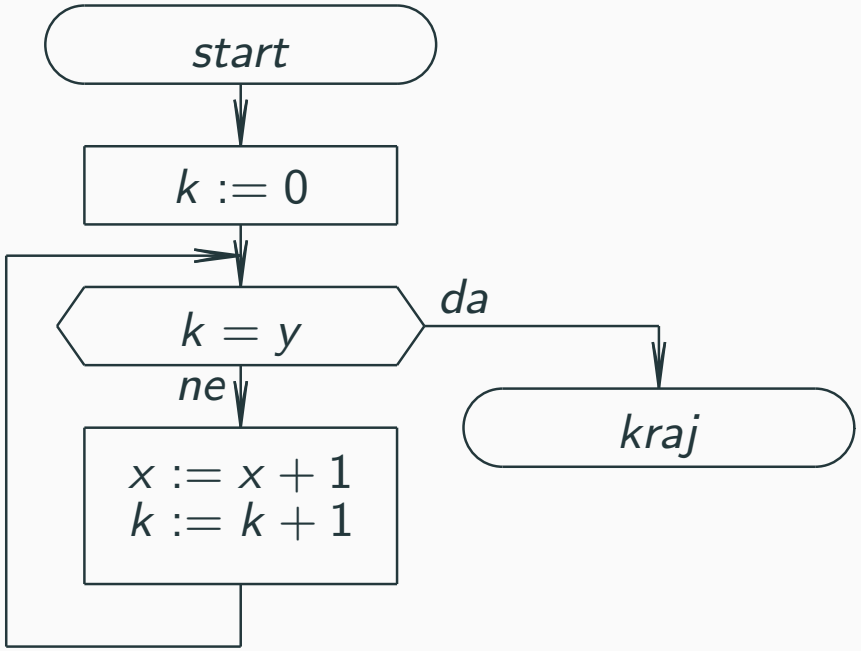
$$f(x, y) = x + y$$

$$x + y = x + \underbrace{1 + 1 + \cdots + 1}_{y}$$

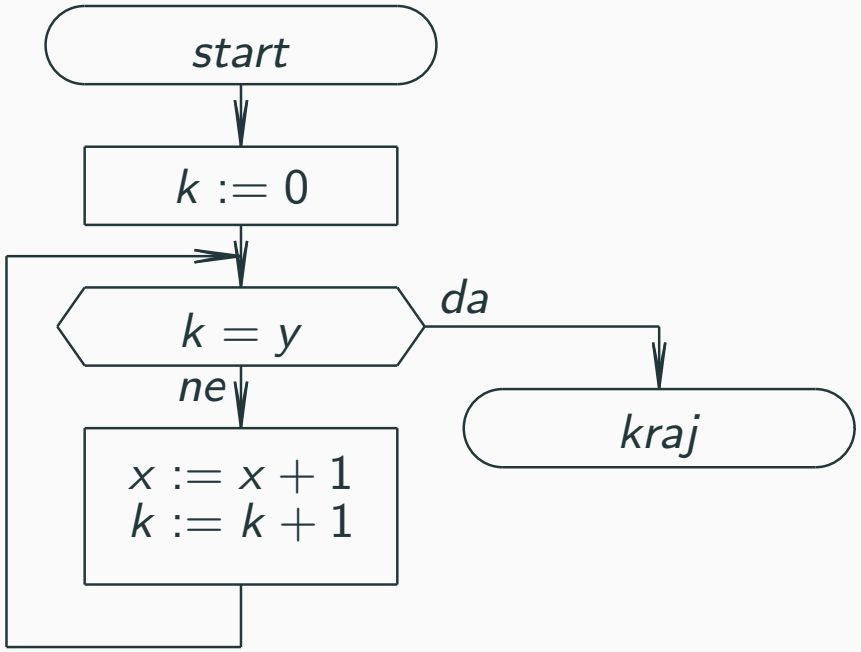
x	y	$\ldots$	$\ldots$

$x+k$	y	k	$\ldots$

URM program – primer1



URM program – primer1



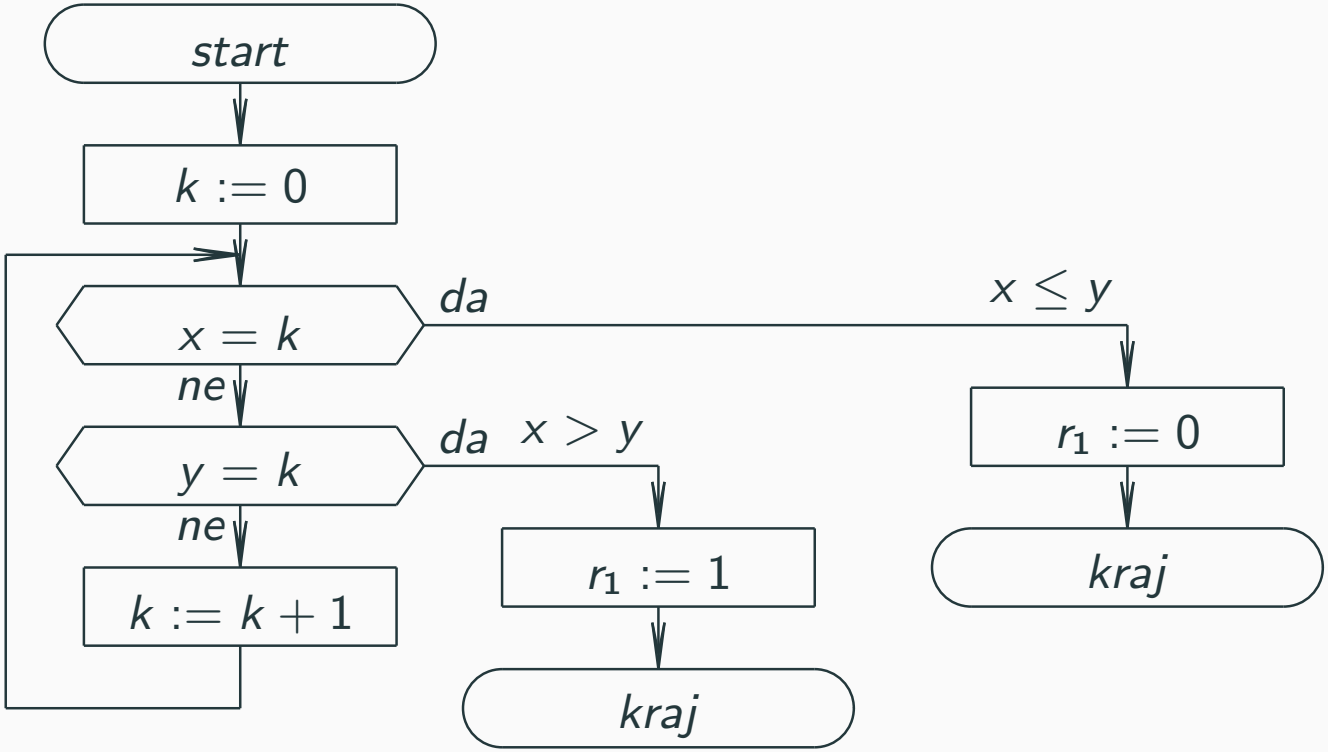
$x+k$	y	k	...

1. $Z(3)$
2. $J(3, 2, 100)$
3. $S(1)$
4. $S(3)$
5. $J(1, 1, 2)$

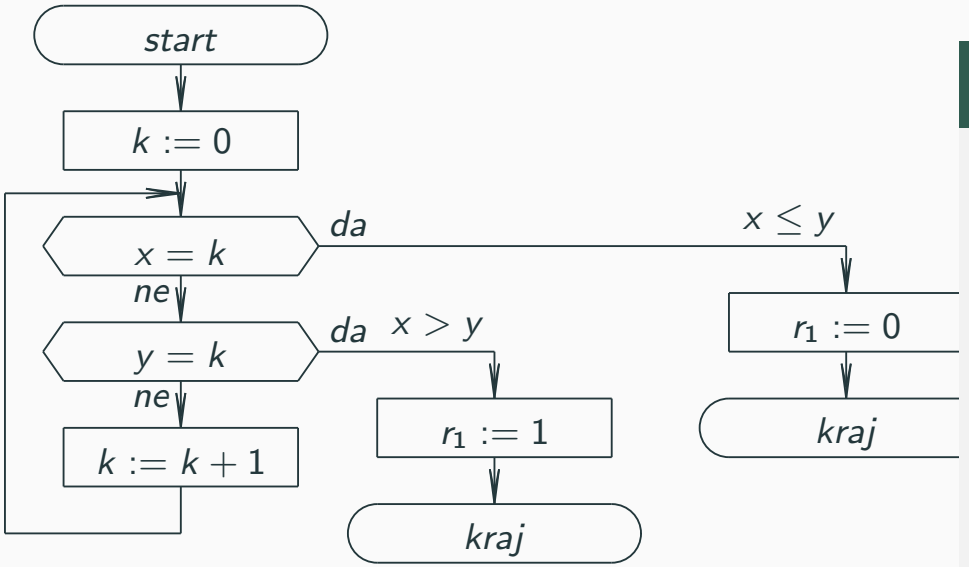
URM program – primer2

$$f(x,y) = \begin{cases} 0 & \text{, ako } x \leq y \\ 1 & \text{, inače} \end{cases}$$

URM program – primer2



URM program – primer2



x	y	k	...

1.	$Z(3)$	$k = 0$
2.	$J(1, 3, 6)$	$x = k?$
3.	$J(2, 3, 8)$	$y = k?$
4.	$S(3)$	$k := k + 1$
5.	$J(1, 1, 2)$	
6.	$Z(1)$	$r_1 := 0$
7.	$J(1, 1, 100)$	kraj
8.	$Z(1)$	
9.	$S(1)$	$r_1 := 1$

URM program – primer3

- Kako izračunati $\sqrt{2}$?

URM program – primer3

- Kako izračunati $\sqrt{2}$?
- Zahtev:izračunati njegovu k-tu cifru.

URM program – primer3

- Kako izračunati $\sqrt{2}$?
- Zahtev:izračunati njegovu k-tu cifru.
- Na osnovu $n = [\sqrt{x}] \Leftrightarrow n^2 \leq x < (n+1)^2$ može se pokazati da je izračunljiva naredna funkcija:

$$f(x) = [\sqrt{x}]$$

URM program – primer3

- Kako izračunati $\sqrt{2}$?
- Zahtev:izračunati njegovu k-tu cifru.
- Na osnovu $n = [\sqrt{x}] \Leftrightarrow n^2 \leq x < (n+1)^2$ može se pokazati da je izračunljiva naredna funkcija:

$$f(x) = [\sqrt{x}]$$

- Izračunljiva je i funkcija $g(k) = [10^k \cdot \sqrt{2}]$

URM program – primer4

$$f(x) = \begin{cases} 0 & \text{, ako je } x = 0 \\ \textit{nedefinisano} & \text{, inače} \end{cases}$$

URM program – primer4

$$f(x) = \begin{cases} 0 & \text{, ako je } x = 0 \\ \textit{nedefinisano} & \text{, inače} \end{cases}$$

1. $Z(2)$
2. $J(1, 2, 100)$
3. $J(1, 1, 2)$

Zasnivanje pojma algoritma

Enumeracija URM programa

Enumeracija URM programa

- Koliko uopšte ima različitih URM program?

Enumeracija URM programa

- Koliko uopšte ima različitih URM program?
- Ima ih beskonačno, ali postoje razne vrste beskonačnosti.

Enumeracija URM programa

- Koliko uopšte ima različitih URM program?
- Ima ih beskonačno, ali postoje razne vrste beskonačnosti.

Enumeracija URM programa

- Koliko uopšte ima različitih URM program?
- Ima ih beskonačno, ali postoje razne vrste beskonačnosti.

> Za dva skupa kaže se da *imaju istu kardinalnost* ako i samo ako je između njih moguće uspostaviti bijektivno preslikavanje.

Enumeracija URM programa

- Koliko uopšte ima različitih URM program?
- Ima ih beskonačno, ali postoje razne vrste beskonačnosti.

> Za dva skupa kaže se da *imaju istu kardinalnost* ako i samo ako je između njih moguće uspostaviti bijektivno preslikavanje.

> Za skupove koji imaju istu kardinalnost kao skup prirodnih brojeva kaže se da su prebrojivi.

Primeri: parni, neparni brojevi itd...

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , i deset cifara za zapis brojeva.

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , i deset cifara za zapis brojeva.
- Može se napisati samo konačno mnogo programa koji imaju ukupno k simbola, $k = 1, 2, 3, \ldots$.

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , i deset cifara za zapis brojeva.
- Može se napisati samo konačno mnogo programa koji imaju ukupno k simbola, $k = 1, 2, 3, \ldots$.
- Možemo u niz poređati sve URM programe: najpre sve one dužine 1, pa onda dužine 2, pa dužine 3, itd.

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , i deset cifara za zapis brojeva.
- Može se napisati samo konačno mnogo programa koji imaju ukupno k simbola, $k = 1, 2, 3, \ldots$.
- Možemo u niz poređati sve URM programe: najpre sve one dužine 1, pa onda dužine 2, pa dužine 3, itd.

Teorema

Različitih URM programa ima prebrojivo mnogo.

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , i deset cifara za zapis brojeva.
- Može se napisati samo konačno mnogo programa koji imaju ukupno k simbola, $k = 1, 2, 3, \ldots$.
- Možemo u niz poređati sve URM programe: najpre sve one dužine 1, pa onda dužine 2, pa dužine 3, itd.

Teorema

Različitih URM programa ima prebrojivo mnogo.

Za bilo koju vrednost i, smatramo da je funkcija koju izračunava program P_i algoritamski izračunljiva. I obratno, ako je neka funkcija algoritamski izračunljiva onda mora da postoji broj i, takav da program P_i izračunava tu funkciju.

Enumeracija URM programa

- U zapisu URM programa koriste samo sledeći simboli: Z, S, T, J, (,), , , i deset cifara za zapis brojeva.
- Može se napisati samo konačno mnogo programa koji imaju ukupno k simbola, $k = 1, 2, 3, \ldots$.
- Možemo u niz poređati sve URM programe: najpre sve one dužine 1, pa onda dužine 2, pa dužine 3, itd.

Teorema

Različitih URM programa ima prebrojivo mnogo.

Za bilo koju vrednost i, smatramo da je funkcija koju izračunava program P_i algoritamski izračunljiva. I obratno, ako je neka funkcija algoritamski izračunljiva onda mora da postoji broj i, takav da program P_i izračunava tu funkciju.

- Na sličan način može se pokazati da i na bilo kom

Zasnivanje pojma algoritma

Drugi pristupi zasnivanju pojma algoritma

Drugi pristupi zasnivanju pojma algoritma

- *Tjuringove mašine* (Tjuring)
- *rekurzivne funkcije* (Gedel i Klini)
- λ-*račun* (Čerč)
- *Blok dijagrami* (poluformalni način opisa algoritma)

Za sistem izračunavanja koji je dovoljno moćan da izvrši sva izračunavanja koja može da izvrši Tjuringova mašina kaže se da je Tjuring potpun (engl. *Turing complete*).

Drugi pristupi zasnivanju pojma algoritma

Može se rigorozno dokazati da su klase izračunljivih funkcija identične za sve navedene formalizme.

Drugim rečima, svi navedeni formalizmi su Tjuring potpuni i ekvivalentni.

Algoritam je svaki postupak kojim se izračunava neka izračunljiva funkcija.

Zasnivanje pojma algoritma

Savremeni računari i izračunljivost

Savremeni računari i izračunljivost

Da li savremeni računari mogu da izvrše svaki algoritam, što bi bilo očekivano?

Savremeni računari i izračunljivost

Da li savremeni računari mogu da izvrše svaki algoritam, što bi bilo očekivano?

Savremeni računari (tj. savremeni procesori koji su centralni deo računara) očigledno mogu da izvrše sve instrukcije koje ima UR mašina, pa mogu da izračunaju sve funkcije koje mogu da izračunaju nabrojani formalizmi za izračunavanje, tj. sve izračunljive funkcije.

Savremeni računari i izračunljivost

Da li savremeni računari mogu da izvrše svaki algoritam, što bi bilo očekivano?

Savremeni računari (tj. savremeni procesori koji su centralni deo računara) očigledno mogu da izvrše sve instrukcije koje ima UR mašina, pa mogu da izračunaju sve funkcije koje mogu da izračunaju nabrojani formalizmi za izračunavanje, tj. sve izračunljive funkcije.

Ne mogu da izračunaju ništa što ne mogu ti formalizmi!

Savremeni računari i izračunljivost

Da li savremeni računari mogu da izvrše svaki algoritam, što bi bilo očekivano?

> Savremeni računari (tj. savremeni procesori koji su centralni deo računara) očigledno mogu da izvrše sve instrukcije koje ima UR mašina, pa mogu da izračunaju sve funkcije koje mogu da izračunaju nabrojani formalizmi za izračunavanje, tj. sve izračunljive funkcije.

Ne mogu da izračunaju ništa što ne mogu ti formalizmi!

Njihova moć je i manja: svi navedeni formalizmi za izračunavanje podrazumevaju beskonačnu raspoloživu memoriju. Zbog toga savremeni računari (koji raspolažu konačnom memorijom) nisu Tjuring potpuni, mada mogu da izrše sve URM programe koji ne

Zasnivanje pojma algoritma

Savremeni programski jezici i izračunljivost

Savremeni programski jezici i izračunljivost

Za veliku većinu savremenih programskih jezika važi da su Tjuring-kompletni.

Savremeni programski jezici i izračunljivost

Za veliku većinu savremenih programskih jezika važi da su Tjuring-kompletni.

Razlike:

- Nabrojani formalizmi teže da budu što jednostavniji.
- Savremeni programski jezici teže da budu što udobniji za programiranje te uključuju veliki broj operacija.

Čerč-Tjuringova teza

Čerč-Tjuringova teza

Klasa intuitivno izračunljivih funkcija identična je sa klasom formalno izračunljivih funkcija.

- Da li čovek zaista može efektivno izvršiti sva izračunavanja definisana nekom od formalizacija izračunljivosti?
- Da li sva izračunavanja koja čovek intuitivno ume da izvrši zaista mogu da budu opisana korišćenjem bilo kog od precizno definisanih formalizama izračunavanja?

Čerč-Tjuringova teza

Čerč-Tjuringova teza

Klasa intuitivno izračunljivih funkcija identična je sa klasom formalno izračunljivih funkcija.

- Da li čovek zaista može efektivno izvršiti sva izračunavanja definisana nekom od formalizacija izračunljivosti?
- Da li sva izračunavanja koja čovek intuitivno ume da izvrši zaista mogu da budu opisana korišćenjem bilo kog od precizno definisanih formalizama izračunavanja?

Skup funkcija koje može da izračuna čovek == Skup funkcija koje može da izvrši bilo koji savremeni računar == Skup funkcija koje može da izvrši UR mašina.

Zasnivanje pojma algoritma

Neizračunljivost i neodlučivost

Neizračunljivost i neodlučivost

Da li postoje funkcije nad brojevima koje su jasno definisane i totalne, a nisu izračunljive?

Neizračunljivost i neodlučivost

Da li postoje funkcije nad brojevima koje su jasno definisane i totalne, a nisu izračunljive?

Drugim rečima, postoje li problemi koji ne mogu da se reše algoritamski?

Neizračunljivost i neodlučivost

Zašto nam je bitno izučavanje ovakvih problema?

Algoritamski nerešivi ili neodlučivi – primer1

- Da li postoji algoritam kojim se mogu dokazati sve matematičke teoreme?

Algoritamski nerešivi ili neodlučivi – primer2

- Data je jednačina $p(x_1, \ldots, x_n) = 0$ gde je p polinom sa celobrojnim koeficijentima.
  Zadatak: Konstruisati opšti algoritam kojim se određuje da li proizvoljna zadata diofantska jednačina ima racionalnih rešenja.

Algoritamski nerešivi ili neodlučivi – primer3

- Neka su data dva konačna skupa reči.
  Pitanje je da li je moguće nadovezati nekoliko reči prvog skupa i, nezavisno, nekoliko reči drugog skupa tako da se dobije ista reč.
- $\{a, ab, bba\}$ i $\{baa, aa, bb\}$: $bba \cdot ab \cdot bba \cdot a = bb \cdot aa \cdot bb \cdot baa$.
- $\{ab, bba\}$ i $\{aa, bb\}$: ne postoji.
- Zadatak: konstruisati opšti algoritam koji za proizvoljna dva zadata skupa reči određuje da li tražena nadoveziva-nja postoje.

Algoritamski nerešivi ili neodlučivi – primer4

Halting problem

Konstruisati opšti algoritam koji proverava da li se proizvoljni zadati program P zaustavlja za date ulazne parametre.

Zaustavljanje programa i halting problem

Zaustavljanje programa i halting problem

- Da li se neki program zaustavlja za neku ulaznu vrednost?

Zaustavljanje programa i halting problem

- Da li se neki program zaustavlja za neku ulaznu vrednost?
- Da li postoji ulazna vrednost za koju se program zaustavlja?

Zaustavljanje programa i halting problem

Da li postoji URM program koji na ulazu dobija drugi URM program P i neki broj y i ispituje da li se program P zaustavlja za ulazni parametar y?

Zaustavljanje programa i halting problem

Neodlučivost halting problema

Neka je funkcija h definisana na sledeći način:

$$h(x, y) = \begin{cases} 1, & \text{ako se program } P_x \text{ zaustavlja za ulaz } y \\ 0, & \text{inače.} \end{cases}$$

Ne postoji program koji izračunava funkciju h, tj. ne postoji program koji za proizvoljne zadate vrednosti x i y može da proveri da li se program P_x zaustavlja za ulazni argument y.

Neodlučivost halting problema – dokaz

- Neka program H izračunava funkciju h.

Neodlučivost halting problema – dokaz

- Neka program H izračunava funkciju h.
- $H'(x) = H(x, x)$ izračunava $h(x, x)$.

Neodlučivost halting problema – dokaz

- Neka program H izračunava funkciju h.
- $H'(x) = H(x, x)$ izračunava $h(x, x)$.
- $H'(x) = 0$ ako se P_x ne zaustavlja za ulaz x.
  $H'(x) = 1$ ako se P_x se zaustavlja za x.

Neodlučivost halting problema – dokaz

- Neka program H izračunava funkciju h.
- $H'(x) = H(x, x)$ izračunava $h(x, x)$.
- $H'(x) = 0$ ako se P_x ne zaustavlja za ulaz x.
  $H'(x) = 1$ ako se P_x se zaustavlja za x.
- Neka je $Q(x)$ program takav da:
  $Q(x) = 0$ ako se P_x se zaustavlja za x
  $Q(x)$ se ne zaustavlja (beskonačna petlja) ako se P_x zaustavlja za x

$$Q(x) \downarrow 0 \text{ ako je } P_x(x) \uparrow$$

$$Q(x) \uparrow \text{ ako je } P_x(x) \downarrow$$

Neodlučivost halting problema – dokaz

Q se nalazi u nizu svih programa.

$$P_k(x) \downarrow 0 \text{ ako je } P_x(x) \uparrow$$

$$P_k(x) \uparrow \text{ ako je } P_x(x) \downarrow$$

Neodlučivost halting problema – dokaz

Šta ako je $k = x$?

$$P_k(k) \downarrow 0 \text{ ako je } P_k(k) \uparrow$$

$$P_k(k) \uparrow \text{ ako je } P_k(k) \downarrow$$

Kontradikcija.

Zaustavljanje programa i halting problem

- Halting problem je neodlučiv, tj. ne postoji opšti postupak kojim se za proizvoljni zadati program može utvrditi da li se on zaustavlja za zadate vrednosti argumenata.

Zaustavljanje programa i halting problem

- Halting problem je neodlučiv, tj. ne postoji opšti postupak kojim se za proizvoljni zadati program može utvrditi da li se on zaustavlja za zadate vrednosti argumenata.
- Za mnoge konkretne programe, može se utvrditi da li se zaustavljaju ili ne.

Zaustavljanje programa i halting problem

- Halting problem je neodlučiv, tj. ne postoji opšti postupak kojim se za proizvoljni zadati program može utvrditi da li se on zaustavlja za zadate vrednosti argumenata.
- Za mnoge konkretne programe, može se utvrditi da li se zaustavljaju ili ne.
- Zaustavljanje svakog programa mora se ispitivati zasebno i koristeći specifičnosti tog programa.

Zaustavljanje programa i halting problem

- Halting problem je neodlučiv, tj. ne postoji opšti postupak kojim se za proizvoljni zadati program može utvrditi da li se on zaustavlja za zadate vrednosti argumenata.
- Za mnoge konkretne programe, može se utvrditi da li se zaustavljaju ili ne.
- Zaustavljanje svakog programa mora se ispitivati zasebno i koristeći specifičnosti tog programa.
- U programima u kojima su petlje jedine naredbe koje mogu dovesti do nezaustavljanja potrebno je dokazati zaustavljanje svake pojedinačne petlje.

Zaustavljanje programa i halting problem

- Halting problem je neodlučiv, tj. ne postoji opšti postupak kojim se za proizvoljni zadati program može utvrditi da li se on zaustavlja za zadate vrednosti argumenata.
- Za mnoge konkretne programe, može se utvrditi da li se zaustavljaju ili ne.
- Zaustavljanje svakog programa mora se ispitivati zasebno i koristeći specifičnosti tog programa.
- U programima u kojima su petlje jedine naredbe koje mogu dovesti do nezaustavljanja potrebno je dokazati zaustavljanje svake pojedinačne petlje.
-

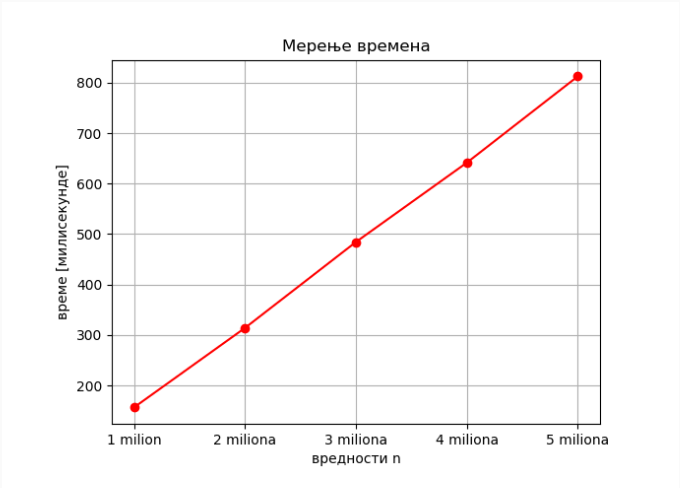
Algoritmika i vremenska i prostorna složenost izračunavanja

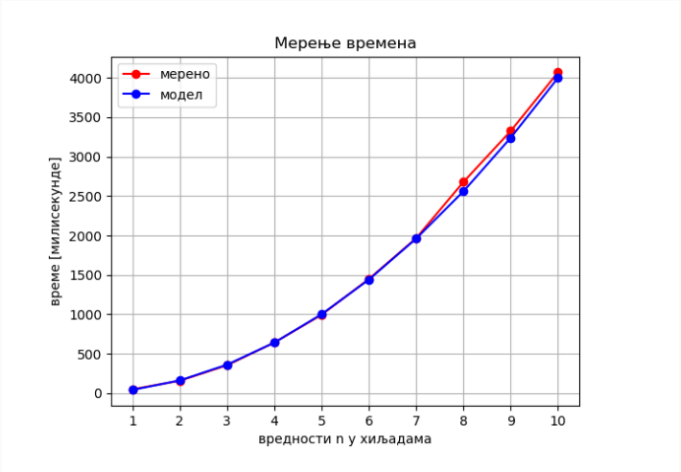
Algoritmika i vremenska i prostorna složenost izračunavanja

Koliko izvršavanje programa zahteva vremena i prostora (memorije)?

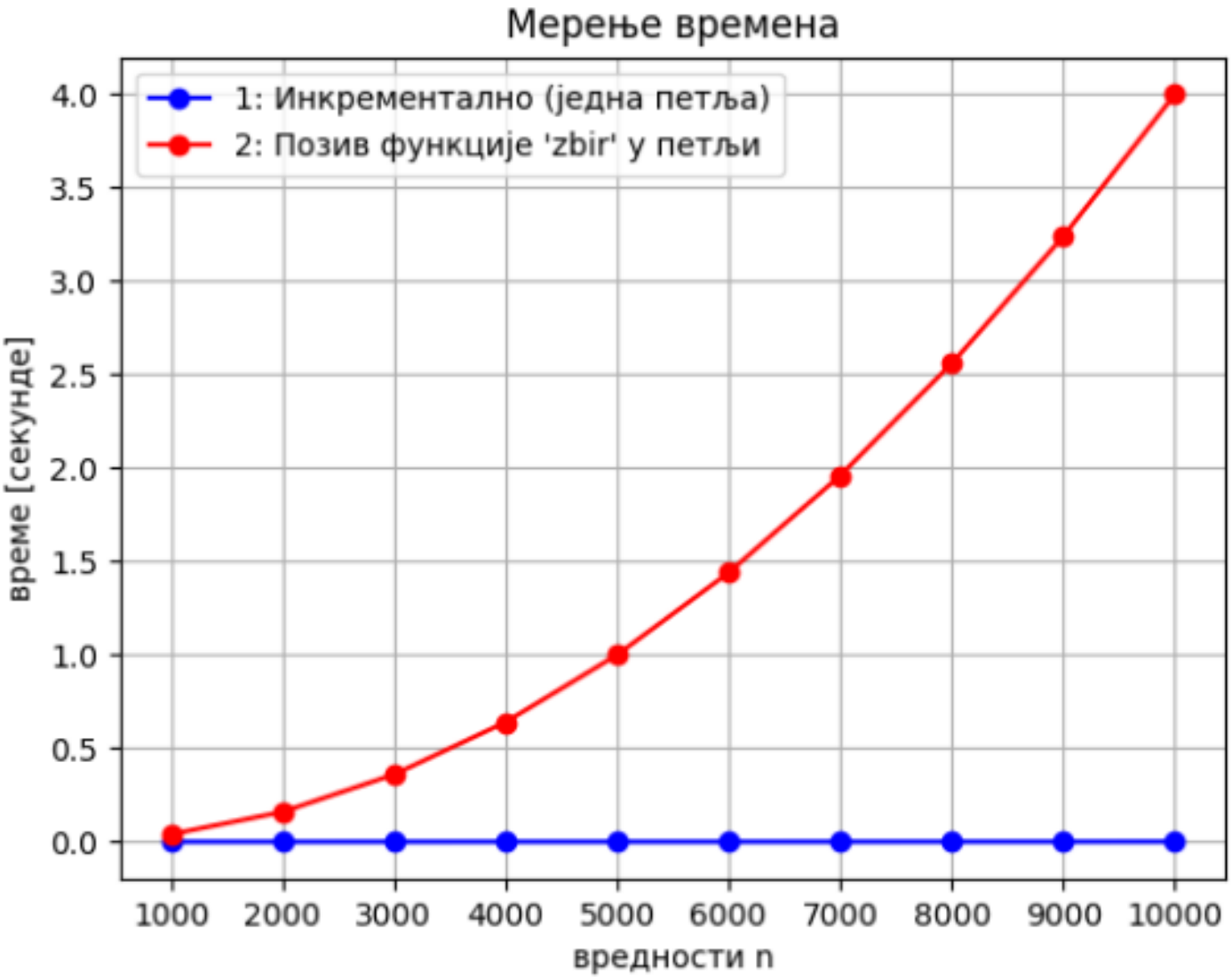
Najčešće se složenost algoritma određuje tako da ukazuje na to koliko on može utrošiti vremena i prostora u **najgorem slučaju**.

Efikasno i neefikasno rešenje – poređenje





Efikasno i neefikasno rešenje – poređenje



Efikasno i neefikasno rešenje – poređenje

Na 1000 puta sporijem računaru se izvršava efikasnije rešenje.