Programiranje 2

Formalna definicija algoritma

Milena Vujošević Janičić
Jelena Graovac

www.matf.bg.ac.rs/~milena
www.matf.bg.ac.rs/~jgraovac

Matematički fakultet

Pregled

Pregled

Algoritam

- Šta je algoritam?

Algoritam

Neformalno, algoritam je precizan opis postupka za rešavanje nekog problema u konačnom broju koraka.

- Koje algoritme znate?

Algoritam

- Šta je algoritam?

Algoritam

Neformalno, algoritam je precizan opis postupka za rešavanje nekog problema u konačnom broju koraka.

- Koje algoritme znate?
- Algoritmi u matematici (sabiranje, množenje, nzd...)
- Algoritmi u računarstvu (minimum, maksimum, sortiranje...)

Algoritam

- Šta je algoritam?

Algoritam

Neformalno, algoritam je precizan opis postupka za rešavanje nekog problema u konačnom broju koraka.

- Koje algoritme znate?
- Algoritmi u matematici (sabiranje, množenje, nzd...)
- Algoritmi u računarstvu (minimum, maksimum, sortiranje...)

Formalizacija

- Kako znamo da za neki problem postoji algoritam koji ga rešava?
- Da li za svaki problem postoji algoritam koji ga rešava?

Formalizacija

- Kako znamo da za neki problem postoji algoritam koji ga rešava?
- Da li za svaki problem postoji algoritam koji ga rešava?
- Da li postoji problem za koji ne postoji algoritam koji ga rešava?

Formalizacija

- Kako znamo da za neki problem postoji algoritam koji ga rešava?
- Da li za svaki problem postoji algoritam koji ga rešava?
- Da li postoji problem za koji ne postoji algoritam koji ga rešava?
- Kako znamo da za neki problem ne postoji algoritam koji ga rešava?

Formalizacija

- Kako znamo da za neki problem postoji algoritam koji ga rešava?
- Da li za svaki problem postoji algoritam koji ga rešava?
- Da li postoji problem za koji ne postoji algoritam koji ga rešava?
- Kako znamo da za neki problem ne postoji algoritam koji ga rešava?

Formalizacija pojma algoritma

Da bi moglo da se diskutuje o tome šta se može, a šta ne može izračunati algoritamski, potrebno je najpre precizno definisati pojam algoritma.

Formalizacija

- Kako znamo da za neki problem postoji algoritam koji ga rešava?
- Da li za svaki problem postoji algoritam koji ga rešava?
- Da li postoji problem za koji ne postoji algoritam koji ga rešava?
- Kako znamo da za neki problem ne postoji algoritam koji ga rešava?

Formalizacija pojma algoritma

Da bi moglo da se diskutuje o tome šta se može, a šta ne može izračunati algoritamski, potrebno je najpre precizno definisati pojam algoritma.

Formalizacija

- (Hilbert, 1928.) Da li postoji algoritam kojim se (pojednostavljeno rečeno) mogu dokazati sve matematičke teoreme?
- Dovoljno je razmatrati algoritme za izračunavanje funkcija čiji su i argumenti i rezultujuće vrednosti prirodni brojevi (a drugi matematički i nematematički algoritmi se, digitalizacijom, mogu svesti na taj slučaj).

Formalizacija

- Jedan primer sistema izračunavanja koji se može koristiti u cilju formalizacije pojma algoritma je *mašina sa beskonačno mnogo registara (engl. Unlimited Register Machine — URM)*.
- *Blok dijagrami* se mogu smatrati poluformalnim načinom opisa algoritama. Oni će biti korišćeni u rešavanju nekih zadataka, radi lakšeg razumevanja.

Pregled

Unlimited register machine

- UR mašina je verovatno najbliža savremenom stilu programiranja: ur mašina ima ograničen (mali) skup instrukcija, programe i memoriju

URM instrukcije

Osnovne instrukcije UR mašine su:

- Z(m) — nula instrukcija
- S(m) — instrukcja sledbenik
- T(m,n) — instrukcija prenosa
- J(m,n,p) — instrukcija skoka

Instrukcije čitaju brojeve i upisuju ih u registre beskonačne trake koja služi kao prostor za čuvanje podataka, tj. kao memorija mašine. Stanje registara u nekom trenutku zovemo konfiguracija.

URM program

- URM program je konačan numerisan niz URM instrukcija.
- Instrukcije se izvršavaju redom (počevši od prve), osim u slučaju instrukcije skoka.

URM početna i krajnja konfiguracija

- Početnu konfiguraciju čini niz prirodnih brojeva upisanih u početne registre koji odgovaraju argumentima funkcije.
- Podrazumeva se da na kraju rada programa rezultat treba da bude smešten u prvi registar

Primer

Napisati URM program koji izračunava funkciju $f(x, y) = x + y$

Ideja: $x + y = x + 1 + 1 + ... + 1$

Nacrtajmo prvo početnu konfiguraciju registara

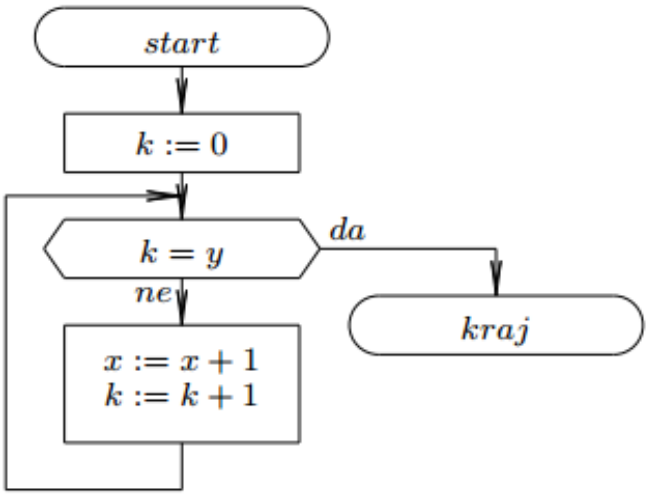
R_1	R_2	R_3	$\ldots$
x	y	$\ldots$	$\ldots$

a zatim radnu konfiguraciju registara

R_1	R_2	R_3	$\ldots$
$x+k$	y	k	$\ldots$

Primer

Nacrtajmo sada dijagram toka ovog algoritma



Primer

Na kraju, napišimo URM program ovog algoritma

```
1. Z(3)          /*Brojac koliko puta smo dodali 1*/
2. J(3,2,100)    /*Ako smo dodali y puta, zavrsava se program*/
3. S(1)          /*Sledbenik dodaje 1 na tekucu vrednost prvog registra*/
4. S(3)          /*Uvecavamo brojac*/
5. J(1,1,2)      /*Vracamo se na drugu instrukciju*/
```

Primer

Napisati URM program koji vraća vrednost 0, ako je $x \leq y$, a vraća 1 inače.

Početna konfiguracija registara

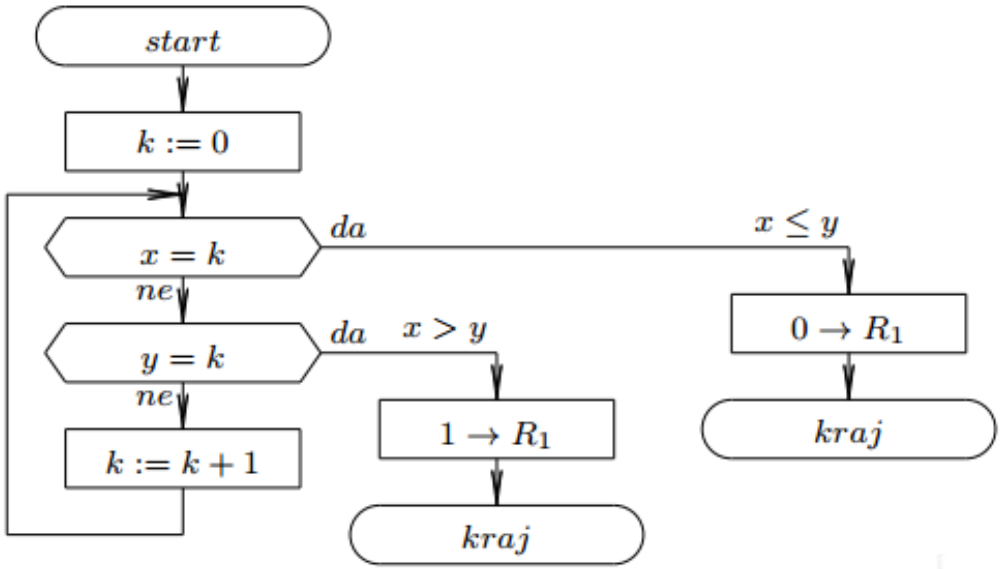
R_1	R_2	R_3	$\ldots$
x	y	$\ldots$	$\ldots$

Radna konfiguracija registara

R_1	R_2	R_3	$\ldots$
x	y	k	$\ldots$

Primer

Dijagram toka ovog algoritma



Primer

URM program ovog algoritma:

```
1. Z(3)          /*k=0*/
2. J(1,3,6)      /*x ?= k*/
3. J(2,3,8)      /*y ?= k*/
4. S(3)          /*k++*/
5. J(1,1,2)
6. Z(1)          /*nula se smesta u prvi registar*/
7. J(1,1,100)    /*kraj*/
8. Z(1)
9. S(1)          /*jedinica se smesta u prvi registar*/
```

URM izračunljivost

- Kažemo da URM program izračunava funkciju f ako za svaku n-torku argumenata za koju je funkcija f definisana i ima vrednost b istovremeno važi da URM program staje sa radom i da izračunava istu tu vrednost b.
- Funkcija je URM izračunljiva ukoliko postoji URM program koji je izračunava.

Pregled

Formalizmi

- Pitanjem formalizacije pojma algoritma 1920-ih i 1930-ih godina nezavisno se bavilo nekoliko istaknutih matematičara i uvedeno je nekoliko raznorodnih formalizama, tj. nekoliko raznorodnih sistema izračunavanja. Osim URM-a koji je prikazan, tu su i:
  - Tjuringove mašine
  - Rekurzivne funkcije
  - Lambda račun
  - Postove mašine
  - Markovljevi algoritmi

Formalizmi

- Navedeni sistemi nastali su pre savremenih računara i većina njih podrazumeva određene **apstraktne mašine**.
- Opis postupka je dovoljno precizan ukoliko je moguće njime konstruisati neku mašinu koja će taj postupak izvršiti.

Programski jezici

- I savremeni programski jezici predstavljaju precizne opise algoritama i moguće ih je pridružiti gore navedenoj listi.
- Razlika je u tome što nabrojani formalizmi teže da budu što jednostavniji tj. da koriste što manji broj operacija i što jednostavnije modele mašina, dok savremeni programski jezici teže da budu što udobniji za programiranje te uključuju veliki broj operacija.

Ekvivalentnost formalizama

- Za sistem izračunavanja koji je dovoljno bogat da može da simulira Tjuringovu mašinu i tako da izvrši sva izračunavanja koja može da izvrši Tjuringova mašina kaže se da je Tjuring potpun (engl. Turing complete).
- Za sistem izračunavanja koji izračunava identičnu klasu funkcija kao Tjuringova mašina kažemo da je Tjuring ekvivalentan (engl. Turing equivalent)

Ekvivalentnost formalizama

- Ako se neka funkcija može izračunati u nekom od navednih formalizama, onda kažemo da je ona izračunljiva u tom formalizmu.
- Iako su navedni formalizmi međusobno veoma različiti, može se dokazati da su klase izračunljivih funkcija identične za sve njih, tj. svi oni formalizuju isti pojam algoritma i izračunljivosti.
- Drugim rečima, svi navedeni formalizmi su Tjuring ekvivalentni.

Formalizmi i programski jezici

- Svi navedeni formalizmi za izračunavanje, podrazumevaju u nekom smislu, beskonaču raspoloživu memoriju.
- Zbog toga savremeni računari (koji raspolažu konačnom memorijom) opremljeni savremenim programskim jezicima nisu Tjuring potpuni, u strogom smislu.
- Svako izračunavanje koje se može opisati nekim programskim jezikom može se opisati i kao program za Tjuringovu mašinu.

Pregled

Čerč-Tjuringova teza

- Koliko formalne definicije algoritama uspevaju da pokriju naš intuitivni pojam algoritma?
- Da li je zaista moguće efektivno izvršiti sva izračunavanja definisana nekom od formalizacija izračunljivosti i, obratno, da li sva izračunavanja koja intuitivno umemo da izvršimo 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.

Ovo tvrđenje je hipoteza, a ne teorema i ne može biti formalno dokazano.

Pregled

Nadovezivanje reči

- 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.
- Na primer, za skupove $\{a, ab, bba\}$ i $\{baa, aa, bb\}$ jedno rešenje je $bba \cdot ab \cdot bba \cdot a = bb \cdot aa \cdot bb \cdot baa$
- Za skupove $\{ab, bba\}$ i $\{aa, bb\}$ rešenje ne postoji, jer se nadovezivanjem reči prvog skupa uvek dobija reč čija su poslednja dva slova različita, dok se nadovezivanjem reči drugog skupa uvek dobija reč čija su poslednja dva slova ista.

Nadovezivanje reči

- Zadatak je utvrditi da li postoji opšti algoritam koji za proizvoljna dva zadata skupa reči određuje da li tražena nadovezivanja postoje.
- Ovaj problem je algoritamski nerešiv

Diofantske jednačine

- Diofantska jednačina je jednačina oblika $P = 0$, gde je P polinom s celobrojnim koeficijentima i proizvoljnim brojem nepoznatih.
- Zadatak je utvrditi da li postoji opšti algoritam kojim se određuje da li proizvoljna zadata Diofantska jednačina ima racionalnih rešenja.
- Ovaj problem je algoritamski nerešiv.

Hilbert, 1928

- Zadatak je utvrditi da li postoji opšti algoritam koji za proizvoljni zadati skup aksioma i zadato tvrđenje proverava da li je tvrđenje posledica aksioma.
- Rešenje ovog problema proisteklo je iz radova Čerča, Tjuringa i Gedela.
- Ovaj problem je algoritamski nerešiv.

Halting problem

- Zadatak je utvrditi da li postoji opšti algoritam koji proverava da li se proizvoljni zadati program zaustavlja za date ulazne parametre.
- Pitanje zaustavljanja računarskih programa je jedno od najznačajnijih pitanja računarstva i programiranja.
- Često je veoma važno da li se neki program zaustavlja za neku ulaznu vrednost, da li se zaustavlja za ijednu ulaznu vrednost i slično.

Halting problem

- Za mnoge konkretne programe i za mnoge konkretne ulazne vrednosti, na ovo pitanje može se odgovoriti.
- No, nije očigledno da li postoji opšti postupak kojim se za proizvoljni dati program i proizvoljne vrednosti ulaznih argumenata može proveriti da li se program zaustavlja ako se pokrene sa tim argumentima.
- Halting problem je algoritamski nerešiv problem.

Odnos izračunljivih i neizračunljivih funkcija

- Koliko ima različitih URM programa?
- Koliko ima izračunljivih funkcija?
- Koliko ima neizračunljivih funkcija?

Kojih funkcija ima više?

Može se dokazati da funkcija jedne promenljive koje za ulazni prirodni broj vraćaju isključivo 0 ili 1 ima neprebrojivo mnogo, dok programa ima samo prebrojivo mnogo. Iz toga direktno sledi da ima neprebrojivo mnogo funkcija koje nisu izračunljive.

Odnos izračunljivih i neizračunljivih funkcija

- Koliko ima različitih URM programa?
- Koliko ima izračunljivih funkcija?
- Koliko ima neizračunljivih funkcija?

Kojih funkcija ima više?

Može se dokazati da funkcija jedne promenljive koje za ulazni prirodni broj vraćaju isključivo 0 ili 1 ima neprebrojivo mnogo, dok programa ima samo prebrojivo mnogo. Iz toga direktno sledi da ima neprebrojivo mnogo funkcija koje nisu izračunljive.

Pregled

Literatura

- Slajdovi su pripremljeni na osnovu trećeg poglavlja knjige Filip Marić, Predrag Janičić: Programiranje 1
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je koristiti knjigu!