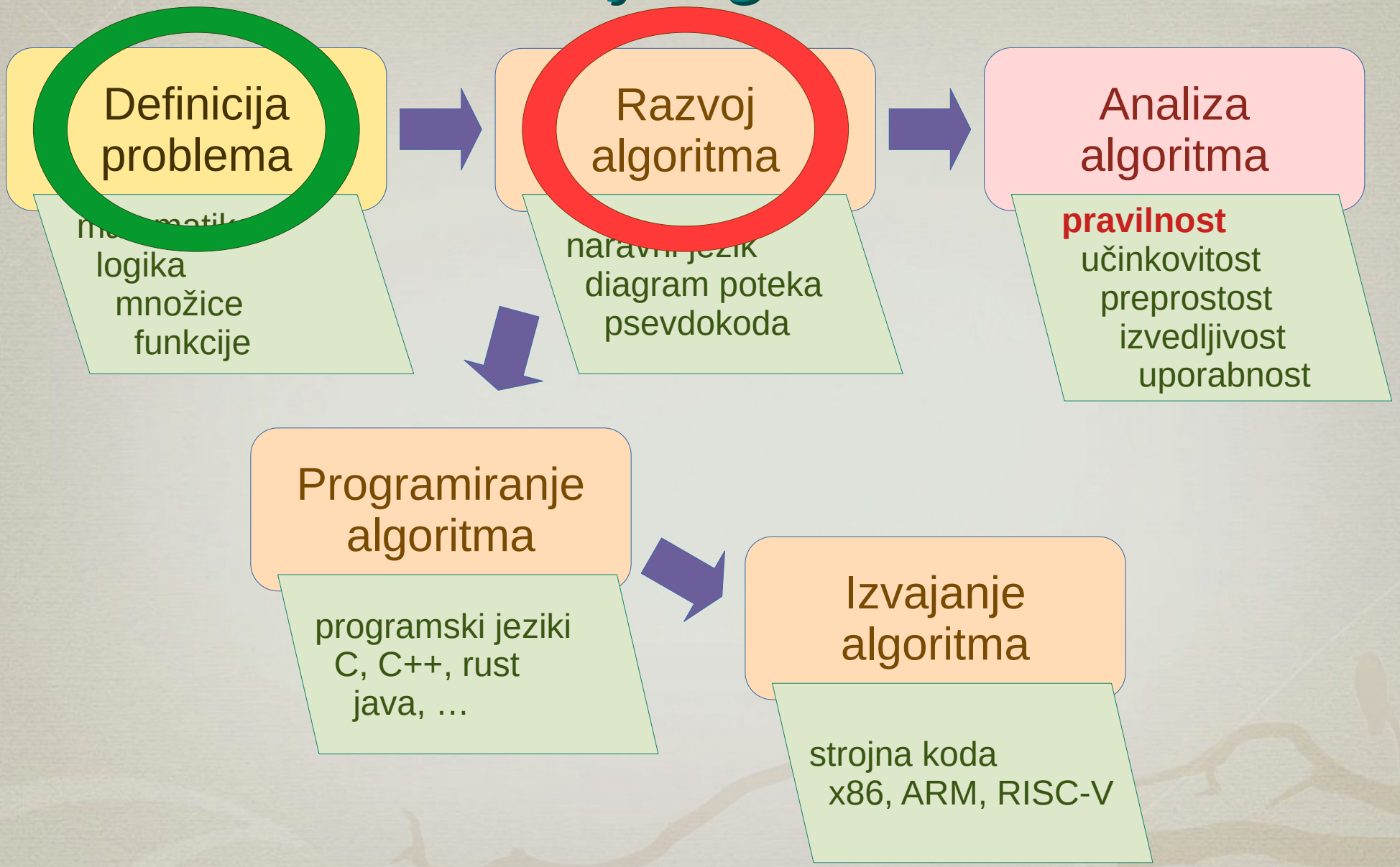
Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Razvoj algoritmov

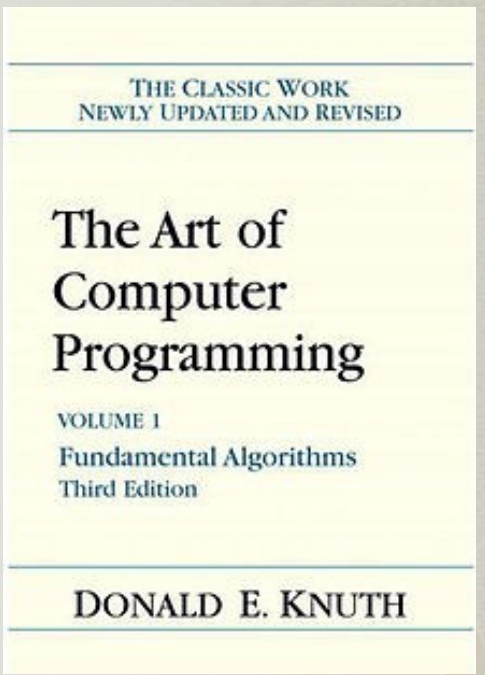
Jurij Mihelič, UniLj, FRI

Razvoj algoritmov



Razvoj algoritmov

- Kako razviti algoritem za dani problem?
- Predpogoj
  - razumevanje problema
  - dobra **definicija problema**
- Cilj
  - **opis algoritma**
  - **pravilnost algoritma**



Razvoj algoritmov

- **Metoda razvoja** algoritmov
  - **sistematičen pristop** k **razvoju** algoritma za reševanje danega problema

Razvoj algoritmov

- Neposredno reševanje
  - preprosti pristopi k problemu
  - pregled (skoraj) vseh možnosti

- groba sila (brute force)
- izčrpno preiskovanje (exhaustive search)
- sestopanje (backtracking)
- razveji & omeji (branch & bound)

Razvoj algoritmov

- Dekompozicija problema
  - delitev problema na manjše probleme
  - podproblemi so iste vrste

- dinamično programiranje (dynamic programing)
- deli & vladaj (divide & conquer)
- zmanjšaj & vladaj (reduce & conquer)
- požrešna metoda (greedy)

Razvoj algoritmov

- Ostale metode
  - prevedi & vladaj (transform & conquer)
  - linearno programiranje (linear programming)
  - iterativne izboljšave (iterative improvement)
  - metahevristike (metaheuristics)
  - evolucijski algoritmi (evolutionary algorithms)
  - randomizacija (randomization)
  - …

Opisovanje algoritmov

- **Naravni jezik**

Iskanje najmanjšega števila v seznamu števil (minimum)
Vsak element seznama primerjaš s prvim in če je element manjši ju zamenjaš.
Prvi element tako postane najmanjši.

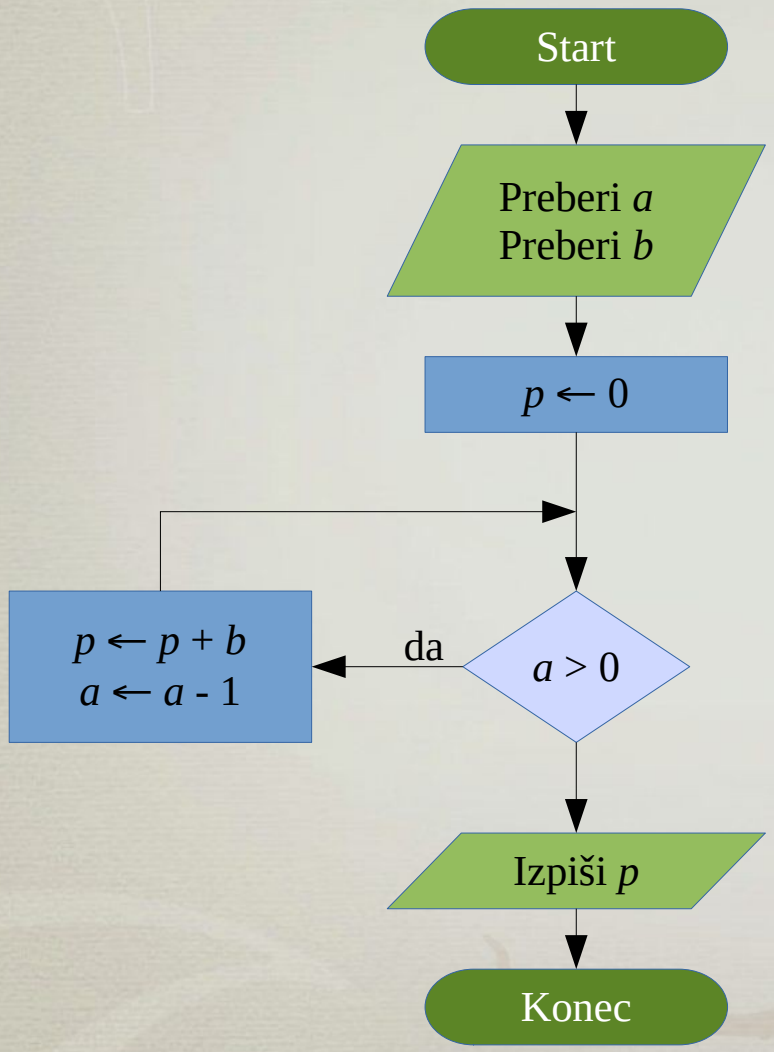
Iskanje elementa v seznamu elementov
Vsak element seznama primerjamo z iskanim in če sta enaka, potem je odgovor DA.
Če noben ni enak, potem je odgovor NE.

Dvojiško iskanje elementa v urejenem seznamu elementov
Iskani element primerjamo s sredinskim elementom seznama.
S tem ugotovimo, ali je iskani element v levi ali desni polovici.
Nato gremo nadaljujem z enakim postopkom v ustrezni levi/desni del seznama.

Nejasnost in dvoumnost jezika, dobro za opis ideje reševanja.

Opisovanje algoritmov

- **Diagram poteka**



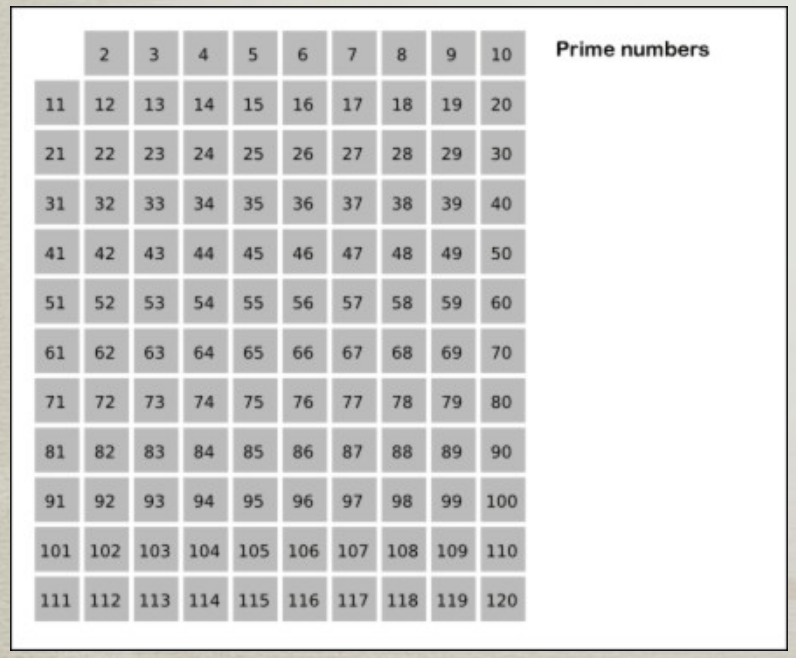
Grafični prikaz, ki omogoča širši pogled, vendar okoren opis podrobnosti

Opisovanje algoritmov

- **Psevdokoda**

Preberi *a* in *b*
p ← 0
while a > 0 **do**
    p ← *p* + *b*
    a ← *a* – *1*
endwhile
Izpiši p

Prilagodljiva natančnost in jasnost opisa, uporaba matematičnih formul.



Glej tudi: http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

Ustvari seznam celih števil od 2 do *N*
Ponavljaj, dokler seznam ni prazen
    Izpiši prvi element seznama
    Iz seznama odstrani vse večkratnike prvega elementa

Opisovanje algoritmov

- **Programski jezik**

```
int gcd(int a, int b) {
   if (b == 0) return a;
   return gcd(b, a % b);
}
```

Programerska realnost,
algoritem lahko dejansko izvedemo,
ogromna izbira jezikov.

```
gcd a 0 = a
gcd a b = gcd b (a `rem` b)
```

```
: gcd ( a b -- c )
    [ abs ] [ [ nip ] [ mod ] 2bi gcd ] if-zero ;
```

Vir programov: http://rosettacode.org/wiki/Greatest_common_divisor

Opisovanje algoritmov

- **Zbirnik in strojna koda**

```
.text
.global pgcd

pgcd:
    push    %ebp
    mov     %esp, %ebp
    mov     8(%ebp), %eax
    mov     12(%ebp), %ecx
    push    %edx
.loop:
    cmp     $0, %ecx
    je      .end
    xor     %edx, %edx
    div     %ecx
    mov     %ecx, %eax
    mov     %edx, %ecx
    jmp     .loop
.end:
    pop     %edx
    leave
    ret
```

Strojna koda je namenjena računalniku, zato se hitro izvaja.

Pravilnost algoritma

- **Specifikacija želenega obnašanja**
  - tisto, kar mislimo, da algoritem dela
  - izhaja iz definicije problema

Iskanje minimuma
- naloga: seznam $L = [x_1, x_2, \ldots, x_n]$ števil
- rešitev: število $m \in L$, kjer $m \leq x$ za vsak $x \in L$

Ali algoritem res dela tisto, kar mislimo, da dela?

- **Dejansko obnašanje algoritma**
  - tisto, kar algoritem res dela
  - sledi iz opisa algoritma

```
m = a[0]
for i = 1 to n-1 do
    if a[i] < m then
        m = a[i]
return m
```

```
for i = 0 to n-1 do
    if a[i] < a[0] then
        swap(a, i, 0)
return a[0]
```

Pravilnost algoritma

- Preverjanje pravilnosti s **testnimi primeri**
  - množica (dobrih) testnih primerov
    - primeri, ki pokrijejo vse veje izvajanja algoritma
    - robni primeri, ki pokrijejo posebne vrednosti
    - za vsak primer pokažemo **pravilnost** ali **nepravilnost**
  - preverjanje algoritma oz. njegove implementacije
    - pogosto lahko avtomatiziramo
    - popolno preverjanje je nepraktično
      - možnih vhodov v algoritem je ogromno
      - št. testnih primerov << št. možnih vhodov
    - v praksi lahko dokažemo le nepravilnost algoritma, pravilnosti pa ne

Pravilnost algoritma

- **Formalni dokaz** pravilnosti algoritma
  - matematični dokaz, da obnašanje algoritma sledi specifikaciji
  - iz specifikacije problema razberemo
    - **lastnosti nalog**, ki jih algoritem lahko obdeluje
    - **lastnosti rešitev**, ki jih vrača algoritem
  - dokaz pravilnost
    - za vsako lastnost rešitve nato dokažemo, da jo algoritem zagotavlja
    - za zanke uporabimo indukcijo in zančne invariante
    - dokaze lahko tudi avtomatiziramo v posebnih programskih jezikih: Coq, Isabelle/HOL, Lean, Agda, …

Sled algoritma

- **Sled algoritma**
  - izpis zanimivih podatkov tekom izvajanja, npr.:
    - spremenljivke, podatkovne strukture
    - št. korakov, globina rekurzije, itd.

- Simulacija izvajanja
  - na papir zapisujemo vrednosti
- Izvajanje z računalnikom
  - vrednosti izpisujemo na zaslon, `printf` metoda

Sled algoritma

- Sled: Evklidov algoritem

```
int gcd(int a, int b) {
   if (b == 0) return a;
   return gcd(b, a % b);
}
```

gcd(264, 72)

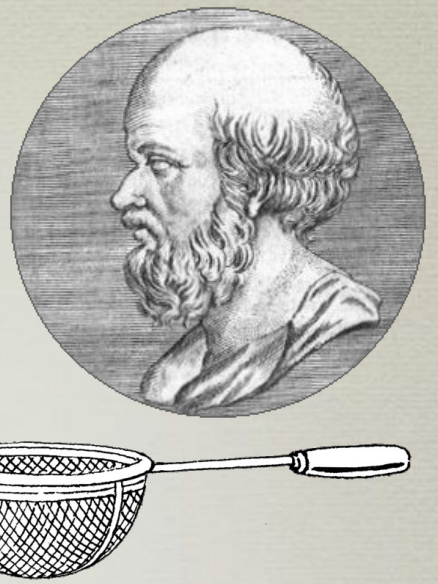
#	a	b	q	r
0	264	72	3	48
1	72	48	1	24
2	48	24	2	0
3	24	0		

```
gcd(264, 72) =
gcd(72, 48) =
gcd(48, 24) =
gcd(24, 0) = 24
```

Sled algoritma

- Sled: Eratostenovo sito

Naj bo dan seznam celih števil od 2 do N
Ponavljaj, dokler seznam ni prazen
    Izpiši prvi element seznama
    Iz seznama odstrani vse večkratnike prvega elementa



Eratostenovo sito za N = 30

#	seznam																									
1	**2**	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2		**3**		5		7		9		11		13		15		17		19		21		23		25		27		29	
3				**5**		7				11		13				17		19				23		25				29	
4						**7**				11		13				17		19				23						29	
5										**11**		13				17		19				23						29	
6												**13**				17		19				23						29	
7																**17**		19				23						29	
8																		**19**				23						29	
9																						**23**						29	
10																												**29**	