Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Razvoj algoritmov



Jurij Mihelič, UniLj, FRI

Definicija problema

logika množice funkcije



Razvoj algoritma

naravın jezik diagram poteka psevdokoda



Analiza algoritma

pravilnost

učinkovitost preprostost izvedljivost uporabnost

Programiranje algoritma

programski jeziki C, C++, rust java, ...



Izvajanje algoritma

strojna koda x86, ARM, RISC-V

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

THE CLASSIC WORK NEWLY UPDATED AND REVISED

The Art of Computer Programming

VOLUME 1

Fundamental Algorithms Third Edition

DONALD E. KNUTH

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

Kako razvítí algorítem za nek problem?

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

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

- 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)

Ostale metode

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

• . . .

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.

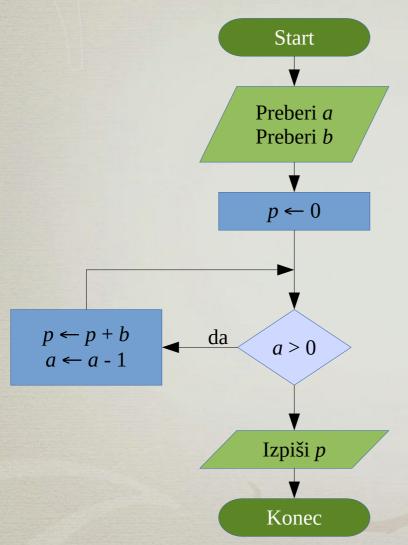
Iskanje elementa v seznamu elementov

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.

Diagram poteka



Grafíční príkaz, kí omogoča šírší pogled, vendar okoren opís podrobností

Psevdokoda

Preberi a in b $p \leftarrow 0$ while a > 0 do $p \leftarrow p + b$ $a \leftarrow a - 1$ endwhile
Izpiši p

Prílagodljíva natančnost ín jasnost opísa, uporaba matematíčníh formul.

	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
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Ustvari seznam celih števil od 2 do *N*Ponavljaj, dokler seznam ni prazen

Izpiši prvi element seznama

Iz seznama odstrani vse večkratnike prvega elementa

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

Programski jezik

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

Programerska realnost, algorítem lahko dejansko izvedemo, ogromna izbira jezíkov.

```
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

0000280 0e 00 00 00 20 00 00 00 0c 00 00 2f 64 2f 6c 69 79 6c 64 00 00 Opisovan 2a 28 fa ed fe 07 00 00 01 03 00 00 80 02 00 00 00 00 38 00 00 00 18 00 00 00 00 01 00 2f 75 01 00 00 00 00 00 00 6c 69 62 53 79 74 65 6d 2e 42 73 0000050 00 00 00 00 00 00 00 00 00 00 00 69 62 00 00 00 00 00 00 26 00 00 00 10 38 01 00 00 00 00 .text 00 00 00 .global pgcd Strojna koda je namenjena računalniku, 00 00 00 00 zato se hitro izvaja. 00 pgcd: 00 0 %ebp push 00 00 00 f8 89 45 mov %esp, %ebp 00 00 00 00 00000d 8b 45 f8 8(%ebp), %eax 00 00 00 00 00000e mov ff ff 89 45 fc 8b d6 e8 c9 48 83 c4 00000f 00 00 00 66 2e Of 84 66 66 66 mov 12(%ebp), %ecx 000010 00 00 00 48 83 ec 00 89 push %edx 00 00 00 0000f80 00 00 e8 99 000011 .loop: 00 00 00 000012 48 83 10 5d c3 90 01 00 00 00 000013 \$0, %ecx Θ 00 00 00 cmp 0000fa0 00 00 00 00 1c 00 00 00 00 00 00 00 1c 00 00 00 000014 00 00 00 00 20 of 00 00 34 00. 00 00 34 00 00 00 jе .end 000015 00 00 00 00 00 00 00 00 00 34 %edx, %edx xor 000016 00 00 00 00 10 00 01 00 00 00 00 00 01 00 div %ecx 00 00 00 00 000017 00 00 00 00 00 00 01 7a 52 000018 mov %ecx, %eax 00 00 00 00 10 0c 07 08 90 01 00 00 00 00 00 00 000019 00 00 00 00 00 05 00 03 5f 6d 68 %edx, %ecx mov 4b 45 44 00001a 4e 65 5f 68 65 61 64 65 72 00 jmp .loop 00001b 01 00 00 00 02 00 00 03 00 2f .end: 00 00 00 00001c 00 00 00 00 00 00 a0 1e 50 1e 00001d %edx 01 00 00 00 00 0001040 fa de 0c 05 00 00 00 14 00 00 00 01 pop 00001e 00 00 80 30 00 00 00 00 00 00 00 00 02 $\Theta \Theta = \Theta \Theta$ leave 00 00 00 00 00 00 00 00001f 0001060 00 00 00 01 00 00 00 16 00 00 00 ret 01 00 00 00 1b 00 0000210 00 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 5f 00 00 00 65 5f 68 65 75 74 00 00 00 03 00 00 00 03 00010b0 63 64 00 5f 6d 61 69 6e 00 64 6C 00010c0 75 62 5f 62 69 6e 64 65 72 00 00 00 APS 1, Jurij Mihelič

Pravilnost algoritma

Specifikacija želenega obnašanja

- tisto, kar mislimo, da algoritem dela
- izhaja iz definicije problema

Iskanje minimuma

- naloga: seznam $L = [x_1, x_2, ..., x_n]$ števil
- rešitev: število $m \in L$, kjer $m \le x$ za vsak $x \in L$

Alí algorítem res dela tísto, kar míslimo, da dela?

Dejansko obnašanje algoritma

- tisto, kar algoritem res dela
- sledi iz opisa algoritma

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

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);
}
```

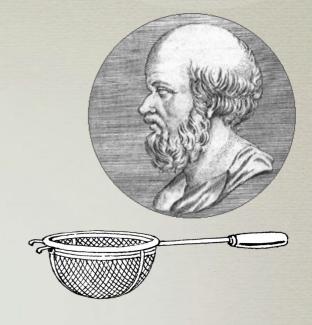
```
gcd(264, 72)

# a b q r
0 264 72 3 48 gcd(264, 72) =
1 72 48 1 24 gcd(72, 48) =
2 48 24 2 0 gcd(48, 24) =
3 24 0 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

#	S	seznam																											
	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
		3		5		7		9		11		13		15		17		19		21		23		25		27		29	
				5		7				11		13				17		19				23		25				29	
						7				11		13				17		19				23						29	
										11		13				17		19				23						29	
												13				17		19				23						29	
7																17		19				23						29	
																		19				23						29	
																						23						29	
																												29	