

Bevezetés az informatikába

8. előadás

Dr. Istenes Zoltán

Eötvös Loránd Tudományegyetem
Informatikai Kar
Programozáselmélet és Szoftvertechnológiai Tanszék

Matematikus BSc - I. félév / 2008 / Budapest



1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Programozási paradigmák

feladat - algoritmus - programozási paradigma - programozási nyelv
programozási paradigmák

- deklaratív (logika, vezérlés leírása nélkül , "*mit* és nem *hogyan*") vs. imperatív
 - funkcionális (pld. Lisp, Scheme, Haskell)
 - logikai (pld. Prolog)
- imperatív (állapot átmenetek, utasítások sorozata) vs. deklaratív
 - objektum orientált
 - procedurális
- iteratív vs. rekurzív
- metaprogramming ("program generálás")
- párhuzamos
- nem determinisztikus (backtrack)
- ...

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- **Programozási nyelvtípusok**
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Programozási nyelvtípusok

Programozási nyelvek "végrehajtásuk", "értelmezésük" szempontjából:

- gépi kód
- assembly
- fordított nyelvek
- interpretált nyelvek
- bájt kód alapú interpretált nyelvek

Gépi kód

a processzor számára közvetlenül értelmezhető "adatsor"
("bitsorozat"), tárgykód

Assembly

a gépi kódnak megfelelő, assembly utasítások, "mnemonik"-ok

- assembler fordító
- rövid, gyors kód, nehéz használat, processzor függő

Z80 mikroprocesszor assembly példa:

```
ld    de, 40000    ; a de regiszterpárba a forráscím
ld    hl, 16384    ; a hl regiszterpárba a célcím
ld    bc, 6912     ; a bc regiszterpárba a blokk hossza
loop  ld    a, (de) ; az a regiszterbe a forrás értéke
      ld    (hl), a ; az a regiszterből a célba
      inc   hl      ; célcím növelése eggyel
      inc   de      ; forráscím növelése eggyel
      dec   bc      ; a hátralévő hossz csökkentése
      ld    a, b    ; bc regiszterpár=0 vizsgálat
      or    c        ;
      jr    nz, loop ; ha nem, ugrás vissza "loop"-ra
      ret           ; visszatérés
```

Fordított nyelvek

Az adott fordított nyelvből, egy fordító program (compiler) állít elő gépi kódú programot.

Példa C programozási nyelv "hello, world" program

http://en.wikibooks.org/wiki/Computer_programming/Hello_world:

```
main()  
{  
    printf("hello, world\n");  
}
```

fordító program:

- lexikai ellenőrzés ("felbontás")
- szintaktikai ellenőrzés ("forrás nyelv")
- szemantikai ellenőrzés ("típusellenőrzés")
- közbenső kód generálás
- optimalizálás
- kódgenerálás

Példák: ALGOL, C, C++, Clipper, Cobol, Fortran, Pascal, PL/1

Interpretált nyelvek

- Az adott interpretált nyelvet, egy értelmező (interpreter), hajtja végre, a kód utasításenkénti értelmezésével.
- A kód futtatásához egy futtató környezete szükséges

PHP példa program:

```
<html>
```

```
<head><title>A hét napjától függő háttérszín</title></head>
```

```
<?
```

```
$today = date("w");
```

```
$bgcolor = array("#FEF0C5", "#FFFFFF", "#FBFFC4", "#FFE0DD", "#E6EDFF",
```

```
?>
```

```
<body bgcolor="<?print("$bgcolor[$today]");?>">
```

```
<br>Ma ilyen színű...
```

```
</body></html>
```

Példák: Awk, BASIC, JavaScript, Logo, PHP

Bájkód alapú interpretált nyelvek

- A fordító program először egy átmeneti kódot (ez a "bájkód") állít elő
- Az interpretált nyelvekhez képest hatékonyabb szintaktikai ellenőrzés, kisebb kódméret, nagyobb sebesség

Perl példa faktoriális számításra:

```
#!/usr/bin/perl
sub fac {
    my ($n) = @_;
    if ($n < 2) {return $n;}
    else {return $n * fac($n-1);}
}
print fac(6), "\n";
```

Példák: Perl, Python, Ruby

Leíró nyelvek

Szöveg és a szövegre vonatkozó extra információk
("leírás", szerkezet, megjelenés, értelmezés)

\LaTeX példa:

```
\begin{frame}[containsverbatim]
\frametitle{Leíró nyelvek}
\begin{block}{}
Szöveg és a szövegre vonatkozó extra információk\\
("leírás", szerkezet, megjelenés, értelmezés)
\end{block}
{\footnotesize
\LaTeX  példa:
\begin{verbatim}
...
```

Példák: \LaTeX , HTML, XML

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

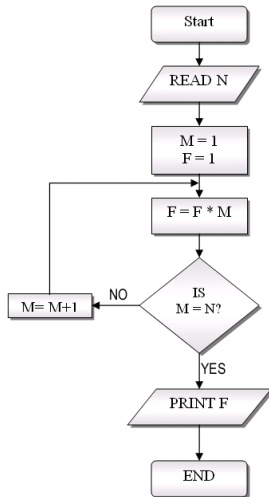
2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Folyamatábra, formális specifikáció



OPERATIONS

$x \leftarrow \text{op_max2}(a, b) =$

PRE

$a:\text{NAT} \ \& \ b:\text{NAT}$

THEN

ANY c

WHERE

$c:\text{NAT} \ \&$

$((c=a \ \& \ a \geq b) \text{ or } (c=b \ \& \ b \geq a))$

THEN

$x := c$

END

END

END

Algoritmusok csoportosítása

- rekurzív / iteratív : visszautalás önmagára, funkcionális programozás, hanoi tornyai
- logikai : logikai következtetés (+ vezérlés)
- soros / párhuzamos : egy vagy több processzor
- determinisztikus / nem determinisztikus : heurisztika
- pontos / közelítő : "nehéz" problémák

Algoritmusok értékelési szempontjai

- komplexitás (futási idő)
- tárigény

jelölés	név	példa
$O(1)$	konstans	páros, páratlan eldöntés
$O(\log n)$	logaritmikus	keresés rendezett listában
$O(n)$	lineáris	keresés rendezetlen listában
$O(n \times \log n)$	loglineáris	rendezés (pld. verem, összefésüléses)
$O(n^2)$	négyzetes	rendezés (pld. beszűrások, buborék)
$O(n^c)$	polinomiális	
$O(c^n)$	exponenciális	utazó ügynök probléma
$O(n!)$	faktoriális	logikai kifejezés megegyezősége

P = NP ? (1.000.000\$) algoritmusok

P : determinisztikus gépen polinomiális idő alatt megoldható

NP : determinisztikus gépen polinomiális idő alatt ellenőrizhető,

vagy nem determinisztikus gépen kiszámítható

Példa : van-e a $\{-2, -3, 15, 14, 7, -10\}$ -nak olyan részhalmaza, aminek az összege 0 ?

Nehezen számolható $O(2^n)$, de könnyen ellenőrizhető $O(n)$.

Megállási probléma

Be fog fejeződni ? Igen, nem, "talán" (eldönthetetlen)...

$k, s := m - 1, 0$
$k \neq n$
$s := s + f(k + 1)$
$k := k + 1$

$i := 1$
$i \neq 0$
$i := i + 1$

$t := 2$
$\exists x, y : \text{prim}(x) \wedge \text{prim}(y) \wedge x + y = t$
$t := t + 2$

Goldbach sejtés (1742) :

minden 2-nél nagyobb páros szám előáll két prímszám összegeként

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Turing gép

Turing gép (1936 Alan Turing) :

- absztrakt automata,
- számítógép modell,
- szimbólum manipuláló eszköz,

matematikai számítások, algoritmusok precíz leírására

Turing gép - klasszikus, informatikai modell

- memória : cellákra osztott, "végtelen", szalag. Minden cellában egy véges ABC egy jele.
- vezérlőegység : gép "programját" tartalmazza. Különféle belső állapotokban lehet.
- író-olvasó fej : jeleket ír vagy olvas a szalag celláira. A szalagon balra vagy jobbra mozdulhat
- átmenettábla : állapot - olvasott jel - jel írás - új állapot - mozgás

Turing gép - matematikai modell

Matematikai modell : $(\Lambda, \Sigma, A, \square, \sigma, \delta, \Phi)$ ahol :

- Λ : szalag ABC
- Σ : belső állapotok halmaza
- A : "mozgás", $A = \{\leftarrow, \downarrow, \rightarrow\}$
- \square : üres jel a szalagon
- σ : kezdőállapot $\sigma \in \Sigma$
- δ : parciális függvény $\delta : (\Lambda_i \times \Sigma) \mapsto (\Lambda_o \times \Sigma \times A)$
- Φ : elfogadható végállapotok $\Phi \subseteq \Sigma$

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- **Turing gép működés**
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Turing gép - működés

- 1 jel beolvasás
- 2 állapottábla (olvasott jel és belső állapot) alapján:
 - új jel írása
 - új belső állapot
 - mozgás a szalagon
- 3 ha nem végállapot akkor újratekdi

Leállítás :

- vagy szabályosan leáll ("STOP állapot")
- vagy sose áll le (sose kerül "STOP állapotba"), végtelen ciklus...

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Turing gép - összeadás az egyes számrendszerben

$3_{10} = 111_1$ 1 : "a" jel * : elválasztó □ : üres

kiindulás : □ 1 1 * 1 1 1 * 1 □

eredmény : □ □ □ 1 1 1 1 1 1 □

áll.	olvas	ír	vált.	mozog	leírás
α	□	□	ω	↓	nincs több szám : ω
α	1	1	α	→	jobbra amíg 1
α	*	*	β	↓	ha *, átvált β -ra
β	□	□	γ	→	visszaért az elejére, γ
β	1	1	β	←	balra amíg 1
β	*	1	β	←	ha *, 1-t ír
γ	1	□	α	→	legelejére 1 írás, újra
ω	□	STOP			leállás

Turing gép - összeadás az egyes számrendszerben

$3_{10} = 111_1$ 1 : "a" jel * : elválasztó □ : üres

kiindulás :

□	1	1	*	1	1	1	*	1	□
---	---	---	---	---	---	---	---	---	---

eredmény :

□	□	□	1	1	1	1	1	1	□
---	---	---	---	---	---	---	---	---	---

áll.	olvas	ír	vált.	mozog	leírás
α	□	□	ω	↓	nincs több szám : ω
α	1	1	α	→	jobbra amíg 1
α	*	*	β	↓	ha *, átvált β -ra
β	□	□	γ	→	visszaért az elejére, γ
β	1	1	β	←	balra amíg 1
β	*	1	β	←	ha *, 1-t ír
γ	1	□	α	→	legelejére 1 írás, újra
ω	□	STOP			leállás

Turing gép - összeadás az egyes számrendszerben

$3_{10} = 111_1$ 1 : "a" jel * : elválasztó □ : üres

kiindulás : □ 1 1 * 1 1 1 * 1 □

eredmény : □ □ □ 1 1 1 1 1 1 □

áll.	olvas	ír	vált.	mozog	leírás
α	□	□	ω	↓	nincs több szám : ω
α	1	1	α	→	jobbra amíg 1
α	*	*	β	↓	ha *, átvált β -ra
β	□	□	γ	→	visszaért az elejére, γ
β	1	1	β	←	balra amíg 1
β	*	1	β	←	ha *, 1-t ír
γ	1	□	α	→	legelejére 1 írás, újra
ω	□	STOP			leállás

Turing gép - összeadás az egyes számrendszerben

$3_{10} = 111_1$ 1 : "a" jel * : elválasztó □ : üres

kiindulás :

□	1	1	*	1	1	1	*	1	□
---	---	---	---	---	---	---	---	---	---

eredmény :

□	□	□	1	1	1	1	1	1	□
---	---	---	---	---	---	---	---	---	---

áll.	olvas	ír	vált.	mozog	leírás
α	□	□	ω	↓	nincs több szám : ω
α	1	1	α	→	jobbra amíg 1
α	*	*	β	↓	ha *, átvált β -ra
β	□	□	γ	→	visszaért az elejére, γ
β	1	1	β	←	balra amíg 1
β	*	1	β	←	ha *, 1-t ír
γ	1	□	α	→	legelejére 1 írás, újra
ω	□	STOP			leállás

Church-Turing tézis

Church-Turing tézis :

Minden algoritmikusan megoldható probléma megoldható Turing-géppel is.

De nem minden probléma oldható meg Turing géppel.

Church-Turing tézis

Church-Turing tézis :

Minden algoritmikusan megoldható probléma megoldható Turing-géppel is.

De nem minden probléma oldható meg Turing géppel.

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

- Hanoi tornyai - rekurzív algoritmus

Tartalom

1 Paradigmák, nyelvek, algoritmusok

- Programozási paradigmák
- Programozási nyelvtípusok
- Algoritmusok megadása, csoportosítása

2 Turing gép

- A Turing gép meghatározása
- Turing gép működés
- Turing gép példa

3 Algoritmus példák

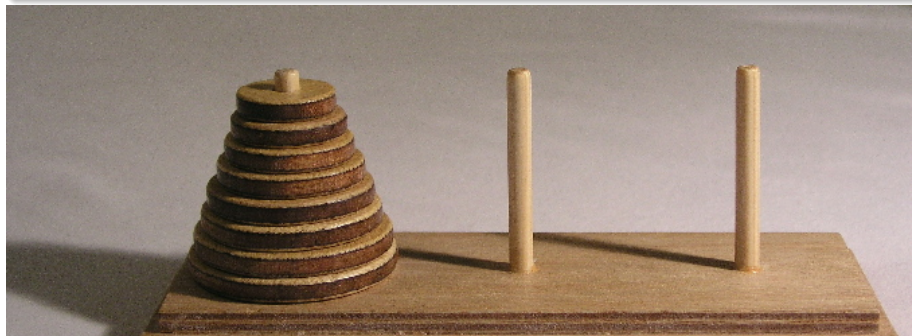
- Hanoi tornyai - rekurzív algoritmus

Hanoi tornyai probléma

Hanoi tornyai probléma

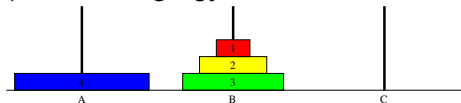
Három rúd (A,B,C), az egyik rúdon (A), adott számú (n darab), eltérő méretű korongok, a legkisebb (#1) legfelül, a legnagyobb (# n) legalul. A korongokat át kell mozgatni egy másik rúdra (C):

- egyszerre csak egy korongot lehet mozgatni
- egy korongot nem lehet kisebb korongra rakni

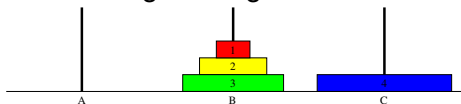


Hanoi tornyai rekurzív algoritmus

- 1 $n-1$ korong átmozgtása A-ról C-re
(a $\#n$ korong egyedül marad az A-n)



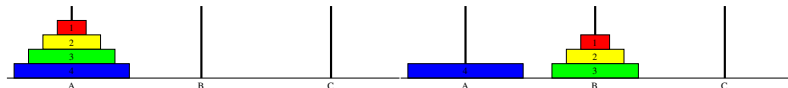
- 2 a $\#n$ korong átmozgatása A-ról B-re



- 3 $n-1$ korong átmozgatása C-ről B-re
(a korongok az $\#n$ -ra kerülnek)



Hanoi tornyai rekurzív algoritmus



Cél: 123: A-B , túl bonyolult...

- ① 12 : A-C
- ② 3 : A-B
- ③ 12 : C-B

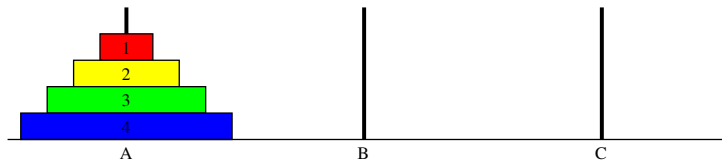
- 12 : A-C
 - 1 : A-B
 - 2 : A-C
 - 1 : B-C
- 3 : A-B
- 12 : C-B
 - 1 : C-A
 - 2 : C-B
 - 1 : A-B

Hanoi tornyai : megoldás Lisp nyelven

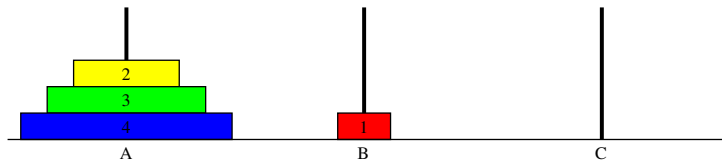
```
(defun dohanoi(n to from u)
  (cond
    ( (> n 0)
      (dohanoi (- n 1) u from to)
      (format t "move ~D --> ~D~&" from to)
      (dohanoi (- n 1) to u from)
    )
  )
)

(defun hanoi(n)
  (dohanoi n 3 1 2)
)
```

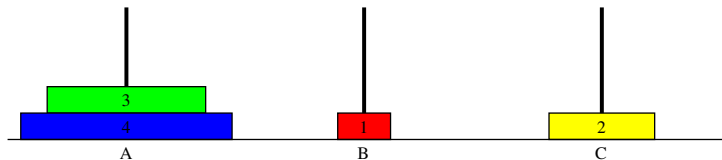
Hanoi...



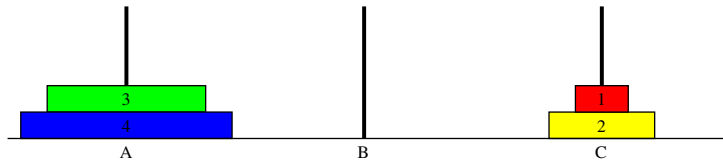
Hanoi...



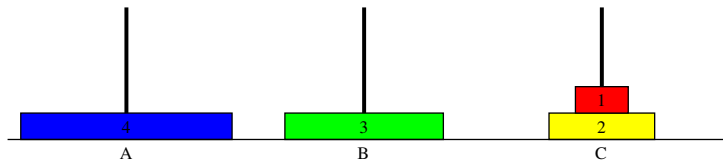
Hanoi...



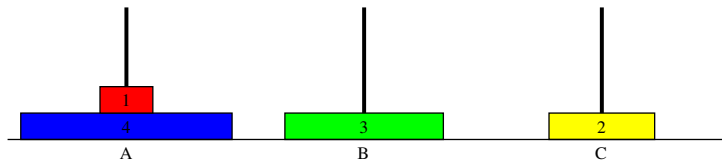
Hanoi...



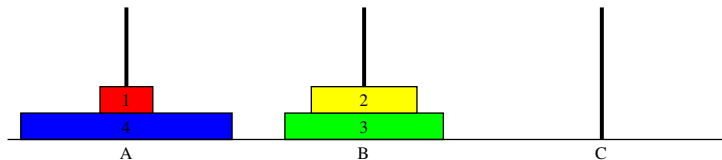
Hanoi...



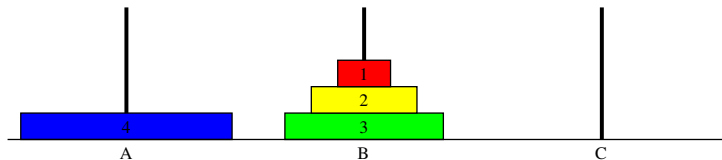
Hanoi...



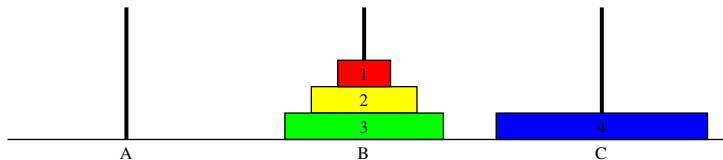
Hanoi...



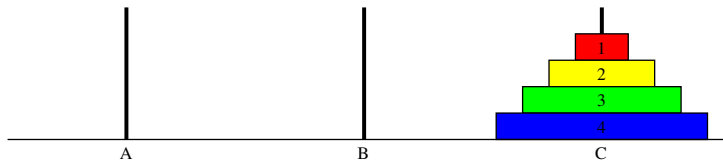
Hanoi...



Hanoi...



Hanoi...



Hanoi tornyai - lépésszám

n korongra lépésszám :

- $T_n = 2 \times T_{n-1} + 1$
- $2^n - 1$

64 korongra lépésszám :

- $2^{64} - 1 = 18,446,744,073,709,551,615$ lépés
- 1 lépés/sec : ~ 583 milliárd év
(az univerzum ~ 13.7 milliárd éves...)

Hanoi tornyai - lépésszám

n korongra lépésszám :

- $T_n = 2 \times T_{n-1} + 1$
- $2^n - 1$

64 korongra lépésszám :

- $2^{64} - 1 = 18,446,744,073,709,551,615$ lépés
- 1 lépés/sec : ~ 583 milliárd év
(az univerzum ~ 13.7 milliárd éves...)

Összefoglalás

