Od linearne funkcije do hanojskega stolpa

Andrej Brodnik
Univerza na Primorskem, DIST
Univerza v Ljubljani, FRI

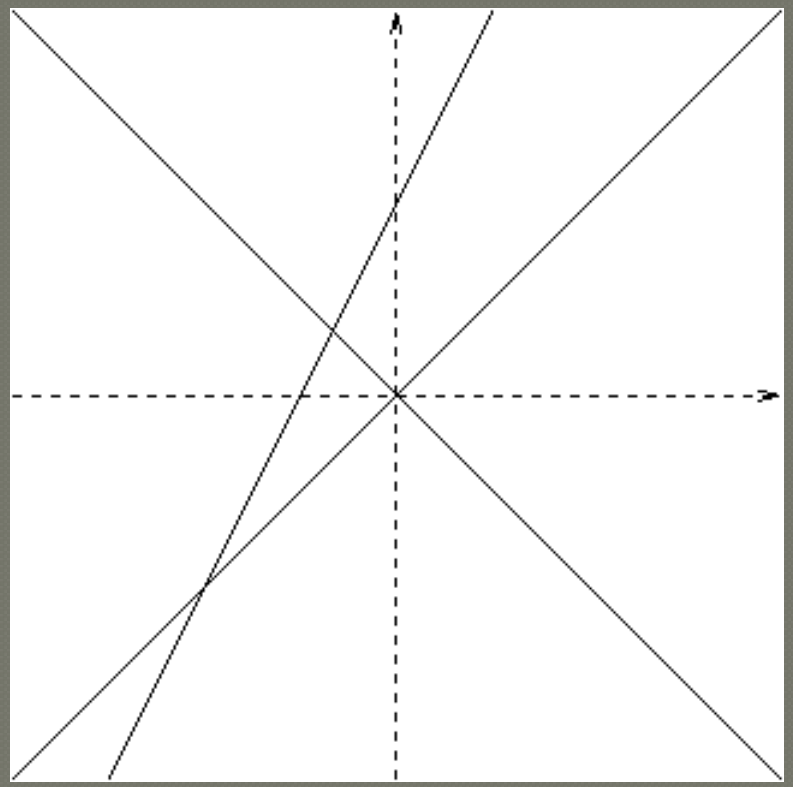
Povzetek

Vsi poznamo linearno funkcijo. Njen graf v ravninskem kartezičnem koordinatnem sistemu je premica. Po drugi strani morda še nismo vsi slišali za hanojski stolp. Hanojski stolp imenujemo tudi bramanski stolp. Sestoji iz niza 64 vedno manjših zlatih diskov z luknjami v sredini. Diski so nameščeni na palčki in bramani jih morajo prestaviti po posebnem pravilu na drugo palčko. Ko bodo prestavili zadnji disk, pravi zgodba, bo konec sveta.

In kaj imata skupnega linearna funkcija ter hanojski stolp?

Linearna funkcija

- $y = k\,x + n$
  - $y = x$; $y = -x$; $y = 2x + 1$
- n – odmik in k – naklon
- lahko zapišemo tudi kot: $f(x) = k\,x + n$
  - $f_1(x) = x$; $f_2(x) = -x$; $f_3(x) = 2x + 1$



Linearna funkcija malo drugače

- $f_1(x) = x$:
  - *Opis:* funkcija $f_1(x)$ izračuna vrednost koordinate y za katerokoli vrednost x za linerano funkcijo $y = x$
- $f_2(x) = -x$:
  - *Opis:* funkcija $f_1(x)$ izračuna vrednost koordinate y za katerokoli vrednost x za linerano funkcijo $y = -x$
- $f_3(x) = 2x + 1$:
  - *Opis:* funkcija $f_1(x)$ izračuna vrednost koordinate y za katerokoli vrednost x za linerano funkcijo $y = 2x+1$

- Zakrivanje ali enkapsulacija.

Zakrivanje

- funkcija $f_1()$ na (magičen) način izračuna vrednost y koordinate pri danem x
- o $f_1()$ moramo vedeti kaj dela in ne kako; potrebujemo njen podpis (signature):

```
function f1(x):
Opis: izračuna y po obrazcu y= x
Parametri: celo/realno število x
Rezultat: celo/realno število y
```

Linearna funkcija še malo drugače

- Imamo funkcije
  - $f_1(x) = x$
  - $f_2(x) = -x$
  - $f_3(x) = 2x + 1$
- ali bi lahko kako posplošili funkcije, da bi nam ne bi bilo potrebno definirati toliko funkcij?
  - pouporaba (*reuse*)

- Posplošitev ali abstrakcija.

Posplošitev

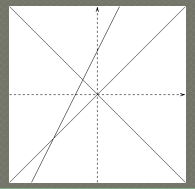
- vse tri funkcije bomo posplošili v eno funkcijo:

```
function f(x, k, n):
Opis: izračuna y po obrazcu y= k*x+n
Parametri: celo/realno število x,
  naklon k in odmik n
Rezultat: celo/realno število y
```

- naše funkcije lahko sedaj zapišemo kot:
  - $f_1(x) = f(x, 1, 0)$
  - $f_2(x) = f(x, -1, 0)$
  - $f_3(x) = f(x, 2, 1)$

Na poti

- pričeli smo pri matematičnih funkcijah
- spoznali smo pojma zakrivanja in posplošitve
- zakrivanje in posplošitev predstavljata osnovna prijema/tehniki, ki ju uporabljamo v računalništvu in informatiki



Risanje

- vzamemo ravnilo, naredimo dve točki ter narišemo premico skozi točki
- kje je palček v računalniku, ki je uporabil ravnilo?
- v računalniku lahko narišemo samo eno točko na enkrat
  - primer: program *octave* in v njem funkcija `plot`
  - `plot([1, 2, 3], [1, 2, 3], "@11")`
- predno se lotimo risanja črte, si poglejmo `plot`

plot

- ..., ki ni ograja, ampak pomeni (na)riši☺
- plot() izgleda skoraj kot naša funkcija f()

```
function plot(x, y, fmt):
Opis: nariše točke z x koordinato in y
  koordinato, pri čemer upošteva format izrisa
  fmt
Parametri: seznam x koordinat točk; seznam y
  koordinat točk; navodilo za obliko izrisa
  fmt ...
Rezultat: ??
```

- funkcija lahko samo nekaj naredi in nič ne vrne
  - stranski učinek (*side-effect*)

Podprogram

- funkciji, ki ne vrne nič (*void*), rečemo včasih podprogram (*procedure*, *subroutine*)

- nazaj k plot in risanju črte

Risanje črte

- vemo:
  - plot zna narisati točke, ki jih podamo kot seznama x in y koordinat
  - če bi funkciji podali toliko mnogo točk, da jih ne bi mogli razločiti, bi dobili učinek črte
- v računalništvu svet ni zvezen, ampak diskreten
- kako do seznama točk?

Izračun točk

- *Če imamo kladivo, potem vsaka stvar izgleda kot žebelj – tudi vijak.*
- Ideja:
  - napišimo funkcijo, ki bo naračunala točke
  - *opomba:* za potrebe tega predavanja bomo točke samo izpisovali na zaslon, kar sicer otežuje njihovo uporabo za risanje
- Kako izgleda funkcija? Kako jo začrtati?

Izračun točk

- Točke bomo izpisovali od nekega začetnega x do končnega xKonc in s korakom delta:

```
tocke(x, delta, xKonc, k, n)
Opis: izpiše točke funkcije kx+n od
  danega x do končnega xKonc s korakom
  delta.
```

- To je kaj, sedaj kako?

Izračun točk

- **Kaj:**

  ```
  tocke(x, delta, xKonc, k, n)
  Opis: izpiše točke funkcije kx+n od
    danega x do končnega xKonc s korakom
    delta.
  ```

- **Kako:**
  - če je x > xKonc: ne naredimo nič
  - če je x <= xKonc:
    - izpišemo x in k*x + n
    - in potem???
- **Deli in vladaj.**

Izračun točk ...

- Potem pa izpišemo še točke od x+delta do končnega xKonc in s korakom delta
- Kako?
  - pokličemo funkcijo točke, z malce spremenjenimi parametri (argumenti)

- Funkcija tocke kliče samo sebe, da opravi delo – rekurzija

Izračun točk

- Kaj (podpis):

```
tocke(x, delta, xKonc, k, n)
Opis: izpiše točke funkcije kx+n od
  danega x do končnega xKonc s korakom
  delta.
```

- Kako (psevdokoda):
  - če je x > xKonc: ne naredimo nič
  - če je x <= xKonc:
    - izpišemo x in k*x + n
    - izpišemo preostale točke z uporabo funkcije tocke

Izračun točk...

```
tocke(x, delta, xKonc, k, n)
Opis: izpiše točke funkcije kx+n od
 danega x do končnega xKonc s korakom
 delta.
 if (x > xKonc) endif          # (1)
 else # if (x <= xKonc)        # (2)
    printf("%f %f\n", x, k*x + n)
    tocke(x+delta, delta, xKonc, k, n)
  endif
endfunction
```

Na poti

- pričeli smo pri matematičnih funkcijah
- spoznali smo pojma zakrivanja in posplošitve

- pojmi podprograma, podpis in psevdokoda
- pri načrtovanju funkcije smo uporabili metodo deli in vladja ter metodo rekurzije

Hanojski stolp

- Problem pravi:

  Hanojski stolp imenujemo tudi bramanski stolp. Sestoji iz niza 64 vedno manjših zlatih diskov z luknjami v sredini. Diski so nameščeni na palčki in bramani jih morajo prestaviti po posebnem pravilu na drugo palčko. Pravilo pravi, da mora biti vedno manjši disk na večjem disku.

- Kako najti pravilno zaporedje prelaganja
  - uporabimo, kar imamo: (i) funkcijo, (ii) deli in vladaj, (iii) zakrivanje, (iv) rekurzija

Izračun točk

- **Kaj:**

```
tocke(x, delta, xKonc, k, n)
Opis: izpiše točke funkcije kx+n od
  danega x do končnega xKonc s korakom
  delta.
```

- **Kako:**
  - če je x > xKonc: ne naredimo nič
  - če je x <= xKonc:
    - izpišemo x in k*x + n
    - izpišemo preostale točke z uporabo funkcije tocke

Hanojski stolp – *kaj*

- Kaj:

```
hanoi(zacetna, koncna, vmesna, n)
Opis: prestavi n diskov iz zacetne
  palčke na končno palčko, pri čemer
  lahko nekaj odloži na vmesno palčko.
```

- Primer:
  - prestavi(leva, desna, srednja, 5)
  - prestavi(leva, desna, srednja, 64)

Hanojski stolp – *kako*

- Kako:
  - če je n = 0: ne naredimo nič
  - če je n = 1: prestavimo disk z začetne na končno palčko
  - če je x > 1:
    - (i) prestavimo (umaknemo) n-1 disk z začetne palčke na vmesno palčko, pri čemer lahko končno palčko sedaj uporabimo za vmesno palčko
    - prestavimo najbolj spodnji (n-ti) disk z začetne na končno palčko
    - (ii) prestavimo še n-1 disk z vmesne palčke na končno palčko, pri čemer lahko začetno palčko sedaj uporabimo za vmesno palčko

Hanojski stolp – *kako*

- Koraka (i) in (ii) sta dejansko enaka kot:
  - (i) prestavimo (umaknemo) n-1 disk z začetne palčke na vmesno palčko, pri čemer lahko končno palčko sedaj uporabimo za vmesno palčko

    hanoi(zacetna, vmesna, koncna, n-1)

  - (ii) prestavimo še n-1 disk z vmesne palčke na končno palčko, pri čemer lahko začetno palčko sedaj uporabimo za vmesno palčko

    hanoi(vmesna, koncna, zacetna, n-1)

Hanojski stolp – *kako*

```
function hanoi(z, k, v, n)
Opis: prestavi n diskov iz zacetne palčke
  na končno palčko, pri čemer lahko nekaj
  odloži na vmesno palčko.
   if (n == 0)
   elseif (n == 1)
     printf("z %s na %s\n", z, k)
   else # if (n > 1)
     hanoi(z, v, k, n-1)
     printf("z %s na %s\n", z, k)
     hanoi(v, k, z, n-1)
   endif
endfunction
```

Hanojski stolp – celotna zgodbica

Hanojski stolp imenujemo tudi bramanski stolp. Sestoji iz niza 64 vedno manjših zlatih diskov z luknjami v sredini. Diski so nameščeni na palčki in bramani jih morajo prestaviti po posebnem pravilu na drugo palčko. Ko bodo prestavili zadnji disk, pravi zgodba, bo konec sveta.

- in kdaj bo konec sveta?

Hanojski stolp in konec sveta

- premikanje diska je v našem programu izpis opisa premika
- premik traja:
  - 10 sek – ročni premik
  - 5 sek – robotski premik
  - 1 msek – miselni premik ☺
- koliko premikov moramo narediti?

```
function hanoi(z, k, v, n)
Opis: prestavi n diskov iz zacetne
   palčke na končno palčko, pri čemer
   lahko nekaj odloži na vmesno
   palčko.
  if (n == 0)
  elseif (n == 1)
    printf("z %s na %s\n", z, k)
  else # if (n > 1)
    hanoi(z, v, k, n-1)
    printf("z %s na %s\n", z, k)
    hanoi(v, k, z, n-1)
  endif
endfunction
```

Hanojski stolp in konec sveta

- recimo, da moramo narediti T(n)= 1.234.567 premikov, potem bo to trajalo:
  - 12.345.670 sek = 142 dni 21 ur 21 min 10 sek pri ročnih premikih;
  - 6.172.835 sek = 71 dni 10 ur 40 min 35 sek pri robotskih premikih; in
  - 1.234,567 sek = 20 min 34 sek 567 msek pri miselnih premikih
- en premik naj torej traja c
- ključno je število T(n)

Hanojski stolp in T(n)

- Koliko je T(n) – štejmo:
  - n = 1: T(1) = 1c
  - n = 2: T(2) = 3c
  - ...
- in za nek poljuben k?
  - T(k) = T(k-1) + c + T(k-1) = 2 T(k-1) + c
  - T(1)= c
  - T(2)= 2 T(1) + c = 3c
  - ...
  - T(n)= 2T(n-1) + c = 2(2T(n-2) + c) + c = 4T(n-2)+3c

```
function hanoi(z, k, v, n)
Opis: prestavi n diskov iz zacetne
   palčke na končno palčko, pri čemer
   lahko nekaj odloži na vmesno
   palčko.
  if (n == 0)
  elseif (n == 1)
    printf("z %s na %s\n", z, k)
  else # if (n > 1)
    hanoi(z, v, k, n-1)
    printf("z %s na %s\n", z, k)
    hanoi(v, k, z, n-1)
  endif
endfunction
```

Hanojski stolp in T(n) ...

$$T(n) = 2T(n-1) + c = 2(2T(n-2) + c) + c = 4T(n-2)+3c$$
$$= 2^2 (2\,T(n-3) + c) + 3c = 2^3\,T(n-3) + 2^2\,c + 3c$$
$$= 2^3\,T(n-3) + 2^2\,c + (2^1 + 2^0)c$$
$$= 2^3\,T(n-3) + (2^2 + 2^1 + 2^0)c$$
$$= \ldots$$
$$= 2^{n-1}\,T(n - (n-1)) + (2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0)c$$
$$= 2^{n-1}\,T(1) + (2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0)c$$
$$= 2^{n-1}\,c + (2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0)c$$
$$= (2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0)c$$

Hanojski stolp in T(n) ...

$T(n) = (2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0)c$

$= (2 - 1)*(2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^2 + 2^1 + 2^0) / (2-1)$ $*c$

$= (2^n - 1) *c$

$= \mathbf{O(2^n)}$

In kdaj bo konec sveta?

- izračunajmo T(n):

```
function rezultat= stevPremikov(n)
  if (n == 1)
    rezultat= 1
  else
    rezultat= 2 * stevPremikov(n-1) + 1
  endif
  printf("%d: %d\n", n, rezultat)
endfunction
```

In kdaj bo konec sveta?

- in koliko je T(n):
  - v enem letu je približno 31.536.000 sek
  - T(35) = 17.179.869.184 -> 5.000 let ročnih premikov
  - za 36 1.000 let
  - ...
  - za 64 je to 2^{64-36} tisoč let, kar je 268 milijard let

Hvala za pozornost!

andrej.brodnik@upr.si

- vabljeni na slovensko državno tekmovanje iz znanja računalništva in informatike: rtk.acm.si