5. Datové soubory

BAB37ZPR – Základy programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Přehled témat

- Část 1 – Pole

  Příklady – 1D pole

  Dvojrozměrné pole – matice

  Příklad – 2D pole

- Část 2 – Řetězce a seznamy
- Část 3 – Soubory

Část I

Pole

I. Pole

Problém sběratele 1/2

Formulace problému

Kolikrát musíme náhodně s opakováním vybrat z množiny $\mathtt{N}$ prvků, než vybereme každý prvek alespoň jednou?

Matematická analýza

$$\text{počet výběrů} = T(N) = \lceil N\,H_N \rceil \approx N(\log N + \gamma) + 0.5$$
$$T(50) = 225$$

H_N – harmonické číslo n-tého řádu
$\gamma \approx 0,577$ – Euler–Mascheroni konstanta

Problém sběratele 2/2

```
import random
import sys

n=50            # pocet ruznych prvku

collectedCount=0  # pocet jiz vybranych ruznych prvku
isCollected=[False]*n   # jiz jsme videli tento prvek
count=0                 # kolik prvku jsme jiz vybrali

while collectedCount < n:
  r=random.randrange(n)  # nahodny prvek 0..n-1
  count+=1
  if not isCollected[r]: # novy prvek
    collectedCount+=1
    isCollected[r]=True

print("Pro n=%d bylo potreba %d vyberu." % (n,count))
```

Prvočísla 1/2

- Dělitelná jen 1 a sebou samým
- Předmět zájmu matematiků od pradávna, cca od 70. let minulého století i velmi důležité aplikace (kryptografie)
- Problémy s provočísly
  - výpis (počet) prvočísel v zadaném intervalu
  - test prvočíselnosti
  - rozklad na prvočísla
- Přímočarý výpis prvočísel v intervalu

```
for num in range(lower,upper + 1):
   if num > 1:
      for i in range(2,num):
         if (num % i) == 0:
            break
         else:
            print(num)
```

Prvočísla 2/2

Možná vylepšení naivního algoritmu

- dělíme pouze dvojkou a lichými čísly
- dělíme pouze do $\sqrt{n}$
- ...

Eratosthenovo síto

- Opakovaně provádíme
  - označ další neškrtnuté číslo v seznamu jako prvočíslo
  - všechny násobky tohoto čísla vyšktrni

Prvočísla 2/2

Možná vylepšení naivního algoritmu

- dělíme pouze dvojkou a lichými čísly
- dělíme pouze do $\sqrt{n}$
- ...

Eratosthenovo síto

- Opakovaně provádíme
  - označ další neškrtnuté číslo v seznamu jako prvočíslo
  - všechny násobky tohoto čísla vyšktrni

Prvočísla – Eratosthenovo síto

```
def reset_multiples(is_candidate, i):
  k = 0
  while k < len(is_candidate):
    is_candidate[k] = 0
    k += i

def eratosthenes(n):
  is_candidate = [1 for _ in range(n)]
  for i in range(2, n):
    if is_candidate[i]:
      print(i, end=" ")
      reset_multiples(is_candidate, i)
```

Histogram

- Histogram (PDF, Probability Density Function) je statistický ukazatel, popisující četnost sledované diskétní veličiny v zadaném intervalu
- Nechť pole `data` obsahuje celá čísla v intervalu 0 – 9

```
data = [0, 1, 3, 6, 8, 1, 6, 3, 4, 3, 4, 1, 7, 5,
  4, 9, 0, 0, 4, 2, 3, 4]
```

- Pro určení histogramu pak potřebujeme pole o stejné velikosti, jako je rozsah sledované veličiny. Pro každý výskyt hodnoty veličiny je pak inkrementována příslušná položka pole

```
h = [0]*10;
for i in range(len(data)):
  h[data[i]] += 1
print(h)
```

I. Pole

Příklady – 1D pole

Dvojrozměrné pole – matice

Příklad – 2D pole

Vytvoření dvourozměrné matice 1/3

```
def print_2d_matrix(a):
  for i in range(len(a)):
    print(a[i])
a=[[0.]*4]*2
print_2d_matrix(a)
```

```
[0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]
```

```
a[0][1]=1.0
print_2d_matrix(a)
```

```
[0.0, 1.0, 0.0, 0.0]
[0.0, 1.0, 0.0, 0.0]
```

Nefunguje jak chceme, kvůli aliasingu.

Vytvoření dvourozměrné matice 1/3

```
def print_2d_matrix(a):
  for i in range(len(a)):
    print(a[i])
a=[[0.]*4]*2
print_2d_matrix(a)
```

```
[0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]
```

```
a[0][1]=1.0
print_2d_matrix(a)
```

```
[0.0, 1.0, 0.0, 0.0]
[0.0, 1.0, 0.0, 0.0]
```

Nefunguje jak chceme, kvůli aliasingu.

Vytvoření dvourozměrné matice 2/3

```
def create_2d_matrix(n,m,v):
  "Creates  a n-by-m matrix with initial value 'v'"
    a=[]
    for i in range(n):
        a+=[ [v]*m ]
    return a

a=create_2d_matrix(2,4,0.0)
a[0][1]=1.0
print_2d_matrix(a)
```

```
[0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]
```

Vytvoření dvourozměrné matice 3/3

```
def create_2d_matrix(n,m,v):
    "Creates  a n-by-m matrix with initial value 'v'"
    return [ [v]*m for i in range(n) ]

a=create_2d_matrix(2,4,0.0)
a[0][1]=1.0
print_2d_matrix(a)
```

```
[0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]
```

I. Pole

Příklady – 1D pole

Dvojrozměrné pole – matice

Příklad – 2D pole

Binomický koeficient (kombinační číslo)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{for } n \geq k \geq 0$$

- počet k prvkových podmnožin z n

$$\binom{3}{1} = 3, \quad \binom{4}{2} = 6$$

- koeficient v binomické větě

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

$$(x+y)^1 = x + y$$

$$(x+y)^2 = x^2 + 2xy + y^2$$

$$(x+y)^3 = x^3 + 3x^2y + 3y^2x + y^3$$

$$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^3$$

Binomický koeficient (kombinační číslo)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{for } n \geq k \geq 0$$

- počet k prvkových podmnožin z n

$$\binom{3}{1} = 3, \quad \binom{4}{2} = 6$$

- koeficient v binomické větě

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

$$(x+y)^1 = x + y$$

$$(x+y)^2 = x^2 + 2xy + y^2$$

$$(x+y)^3 = x^3 + 3x^2y + 3y^2x + y^3$$

$$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^3$$

Pascalův trojúhelník

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
$n=0$	1				
$n=1$	1	1			
$n=2$	1	2	1		
$n=3$	1	3	3	1	
$n=4$	1	4	6	4	1

Platí rekurze

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Pascalův trojúhelník v Pythonu

Vypočítejte pole `p`, tak aby `p[n][k]` = $\binom{n}{k}$

```
def pascal_triangle(N):
  p=[[1]]
  for n in range(2,N+1):
    prev = p[n-2] # předchozí řada
    p += [[1] +
      [prev[k-1] + prev[k] for k in range(1,n-1)] +
      [1]]
  return p
```

Pascalův trojúhelník v Pythonu

```
print_2d_matrix(pascal_triangle(5))
```

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

Řádky pole nemusí mít stejnou délku.

Část II

Řetězce a seznamy

Seznam z řetězce – split

- Rozdělí řetězec podle zadaného oddělovače, vrátí seznam

```
>>> vowels = "a,e,i,o,u,y"
>>> vowels.split(",")
['a', 'e', 'i', 'o', 'u', 'y']
>>> message = ".-..|---|-..|.-"
>>> message.split("|")
['.-..', '---', '-..', '.-']
```

- Využití pro načítání vstupu od uživatele

```
>>> input_string = input()
3 7
>>> xstring, ystring = input_string.split(" ")
>>> x = int(xstring)
>>> y = int(ystring)
```

Část III

Soubory

Práce se soubory

Proč?

- Vstupní data
- Uložení výstupu programu
- Udržovaní stavu programu mezi jednotlivými běhy
- Větší projekty: databáze

Základní operace

- Otevření souboru
- Práce se souborem (čtení, zápis)
- Zavření souboru

Práce se soubory

Proč?

- Vstupní data
- Uložení výstupu programu
- Udržovaní stavu programu mezi jednotlivými běhy
- Větší projekty: databáze

Základní operace

- Otevření souboru
- Práce se souborem (čtení, zápis)
- Zavření souboru

Práce se soubory – otevření a uzavření souboru

- `f = open(filename, mode)`
- jméno souboru: řetězec
- způsob otevření:
  - čtení – `"r`
  - zápis – `"w"` – přepíše soubor, pokud ještě neexistuje, vytvoří jej
  - přidání na konec – `"a"`
  - další možnosti: čtení i zápis, binární režim
- uzavření: `f.close()`

Práce se soubory – čtení a zápis do souboru

- `f.write(text)` – zapíše řetězec do souboru
  - raw výstup – neukončuje řádky, je třeba explicitně použít `'\n'`
- `f.readline()` – přečte jeden řádek
- `f.readlines()` – přečte všechny řádky, vrací seznam řádků
- `f.read(count)` – přečte daný počet znaků
- `f.read()` – přečte celý soubor, vrací řetězec
- `f.tell()` – aktuální pozice v souboru
- `f.seek(position)` – přesun pozice v souboru

Práce se soubory – iterování po řádcích

- Intuitivní způsob

```
for line in f.readlines():
  print(line)
```

- Alternativní způsob

```
for line in my_file:
  print(line)
```

- Načtení pole ze souboru

```
f=open('line.txt','r')
line = f.readline()
pole = list(map(int, line.split()))
```

Práce se soubory – `with`

Speciální blok `with`

- není třeba soubor zavírat – uzavře se automaticky po ukončení bloku
- souvislost s výjimkami

```
with open("/tmp/my_file", "r") as my_file:
  lines = my_file.readlines()
print(lines)
```