Seminarium 02

Experimentering

Gustav Sörnäs

14 september 2021

Seminarieformen

Mer fokus på eget kodskrivande än ”Ett steg i taget”, men mer konkret än ”Diskussion och analys”.

Fritt fram att byta spår under terminen.

Innan seminariet: läs förberedelsematerialet och försök er på uppgifterna.

Skicka in lösningar så vi kan diskutera i helklass: `seminarium.sörnäs.se`. Anonymt, sålänge du inte skriver ditt namn i koden :)

Dagens seminarium

- Datatyper
- Uppgift: iteration
- ~~Fysisk rekursion~~
- Analys: tupler och rekursion
- Uppgift: palindrom
- Analys: dictionaries
- Diskussion

Datatyper – Enkla datatyper

- Strängar (`str`)
- Tal (`int`/`float`)
  Tänk på att flyttal har en maxgräns ($1{,}8 \cdot 10^{308}$)
  och riskerar avrundningsfel:

```
>>> (0.1 + 0.2) * 10 == 3.0
False
```

```
>>> 0.1 + 0.2
0.30000000000000004
```

- Sanningsvärden (`bool`)
- Null-värdet (`None`)

Datatyper – Sammansatta datatyper

- Listor
  Sparar en hög med element (inte nödvändigtvis av samma datatyp).
- Tupler
  Som listor, men de går inte att modifiera.
- Dictionaries
  Associerar ett värde till en nyckel. Typ som en lista som du ”indexerar” med något arbiträrt.

```
>>> fruit_colors = {"banan": "gul", "äpple": "grön"}
>>> fruit_colors["banan"]
"gul"
```

Uppgift: `find_uncut`

Om `True` är en klippt gräsruta och `False` är en oklippt gräsruta, returnera en lista över alla oklippta rutors koordinater.

```
>>> find_uncut([
...     [True,  False, True,  False],
...     [False, False, True,  True],
...     [True,  False, False, True],
...     [False, True,  True,  True]
... ])
...
[(1, 0), (3, 0), (0, 1), (1, 1), (1, 2), (2, 2), (0, 3)]
```

seminarium.sörnäs.se

min_max

```
1   def min_max(seq):
2       if not seq:
3           return (None, None)
4       if len(seq) == 1:
5           return (seq[0], seq[0])
6
7       (l_min, l_max) = min_max(seq[1:])
8
9       if seq[0] < l_min:
10          l_min = seq[0]
11      if seq[0] > l_max:
12          l_max = seq[0]
13
14      print("min: ", l_min)
15      print("max: ", l_max)
16
17      return (l_min, l_max)
```

1. Funktionen körs med `[1, 2, 3]` som indata. Vad skrivs ut? Vad returneras?
2. Fungerar `[1, 2, 'abc']` som indata?
3. Fungerar `(1, 2, 3)` som indata?

Hur en rekursiv funktion kan analyseras

- Vad är basfallen?
- Hur delas problemet upp i samma problem i mindre delar?
- Hur sätts de mindre delarna ihop till grundproblemet?

Exempel på analys av rekursiv funktion: `min_max`

```
1   def min_max(seq):
2       if not seq:
3           return (None, None)
4       if len(seq) == 1:
5           return (seq[0], seq[0])
6
7       (l_min, l_max) = min_max(seq[1:])
8
9       if seq[0] < l_min:
10          l_min = seq[0]
11      if seq[0] > l_max:
12          l_max = seq[0]
13
14      print("min: ", l_min)
15      print("max: ", l_max)
16
17      return (l_min, l_max)
```

Vad är basfallen?Hur delas problemet upp i samma problem i mindre delar?Hur sätts de mindre delarna ihop till grundproblemet?

Paus

Uppgift: palindrom

Skriv två funktioner (en iterativ och en rekursiv) som tar en sträng och returnerar huruvida strängen är ett palindrom eller inte.

Ett palindrom är ett ord som skrivs likadant framlänges som baklänges. Exempel: ABBA, kajak.

`seminarium.sörnäs.se`

Hur en rekursiv funktion kan designas

- Vad är basfallen?
- Hur kan problemet delas upp i samma problem i mindre delar?
- Hur sätts de mindre delarna ihop till grundproblemet?

Exempel på en rekursiv funktion: `palin_rec`

```
def palin_rec(s):
    if len(s) <= 1:
        return True
    return palin_rec(s[1:-1]) \
            and s[0] == s[-1]
```

ABBA

kajak

Iterativt blir det att vi beskriver *hur* snarare än *vad*.

```
def palin_iter(s):
    mid = len(s) // 2
    for i in range(mid):
        if s[i] != s[-(i + 1)]:
            return False
    return True
```

Allmänt om rekursion

Väldigt bra på att lösa vissa typer av problem, främst problem som innehåller rekursiva strukturer. Exempel: trädstrukturer och uttrycksparsing (båda i labb 4).

Föredrog ni `palin_rec` eller `palin_iter`? Var den ena lättare än den andra att skriva / förstå? (Inte nödvändigtvis samma sak.)

dict_func

Vad gör den här funktionen?

```
1  def dict_func(d):
2      result = {}
3
4      for key in d:
5          if d[key] in result:
6              result[d[key]].append(key)
7          else:
8              result[d[key]] = [key]
9      return result
```

Seminariematerialet på nätet

`github.com/sornas/tdde23-seminars.git`

PDF efter seminariet.

Talbaser – decimala tal

$$1234 = \underset{\times 10^3}{1}\ \underset{\times 10^2}{2}\ \underset{\times 10^1}{3}\ \underset{\times 10^0}{4} = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

Decimalt talsystem: 10 siffror (0-9)

Talbaser – binära tal

Binärt talsystem: 2 siffror (0-1)

$$101010_2 = \underset{\times 2^5}{1}\ \underset{\times 2^4}{0}\ \underset{\times 2^3}{1}\ \underset{\times 2^2}{0}\ \underset{\times 2^1}{1}\ \underset{\times 2^0}{0}$$

$$= 2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42_{10}$$

Talbaser – Decimala decimaltal

Fortsätter åt höger med negativa exponenter

$$12{,}34 = \underset{\times 10^{1}}{1}\ \underset{\times 10^{0}}{2}\ ,\ \underset{\times 10^{-1}}{3}\ \underset{\times 10^{-2}}{4} = 1 \cdot 10 + 2 \cdot 1 + 3 \cdot 0{,}1 + 4 \cdot 0{,}01$$

Talbaser – Binära decimaltal

$$101{,}010_2 = \underset{\times 2^2}{1}\ \underset{\times 2^1}{0}\ \underset{\times 2^0}{1}\ ,\ \underset{\times 2^{-1}}{0}\ \underset{\times 2^{-2}}{1}\ \underset{\times 2^{-3}}{0}$$

$$= 2^2 + 2^0 + 2^{-2} = 4 + 1 + 0{,}25 = 5{,}25$$

Vissa tal $\in]0, 1[$ går inte att representera exakt med ett binärt talsystem, på samma sätt som $\frac{1}{3}$ inte går att representera exakt med ett decimalt talsystem ($0{,}333\ldots$).

Flyttal i datorsystem är lite annorlunda (IEEE 754) men poängen kvarstår.