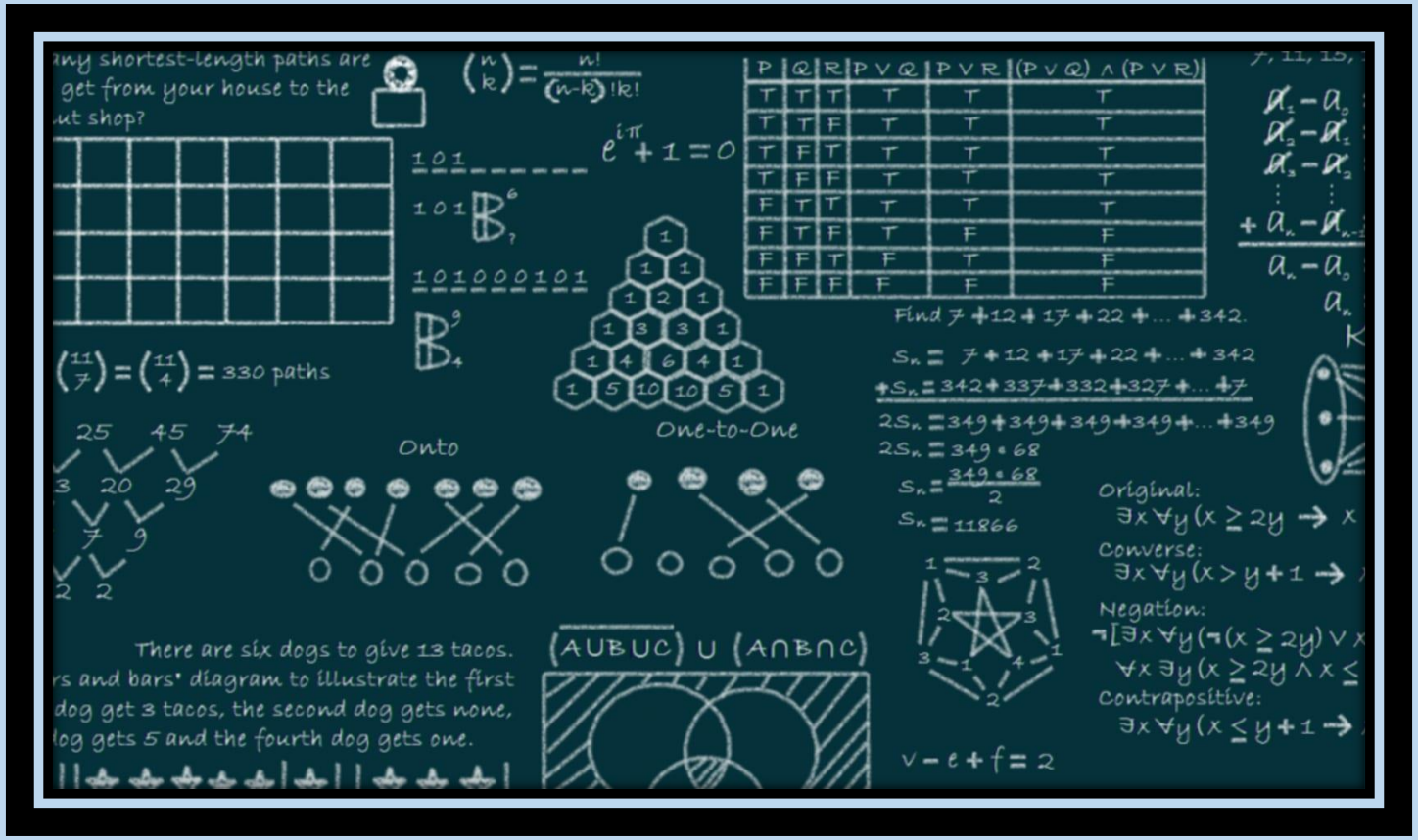


FÖRELÄSNING 6

Datastrukturer och algoritmer
KYH – 2022 HT

Andreas Nilsson Ström

Agenda

- Rekursion

Repetition

Uppvärmning!

Algoritm: Minsta multipel?

- 2520 är det minsta talet som är jämt delbart med alla nummer 1-10
- Vad är det minsta positiva tal som är jämt delbart med alla nummer 1-20?

- Källa: ProjectEuler.net problem 5

Vad är rekursion?

- En metod för att lösa problem genom att:
  - Dela in problemet i **mindre problem** **av samma form**
  - Tills de små problemen är lätta att lösa

- Inom programmering: Betyder vanligtvis att en funktion anropar sig själv
- Låter oss skriva eleganta lösningar på vissa svåra problem

Mönster (Python)

```
def recursive_function(input):
    if (input är så enkelt det kan bli):
        Räkna ut resultatet utan rekursion
    else:
        Dela upp problemet i delproblem
        Anropa recursive_function() på varje delproblem
        Samla ihop resultaten av delproblemen
```

Två fall

Varje rekursiv algoritm behandlar åtminstone två fall:

- Basfallet: Det enkla fallet. Något som kan svaras på direkt. Fallet som rekursionen reduceras till.
- Rekursiva fallet: Ett mer komplext problem som inte kan svaras på direkt, men som kan beskrivas i mindre nedbrytningar av samma problem

Demo

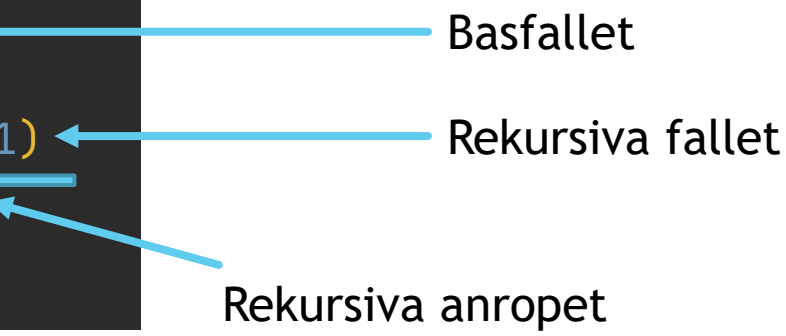
- Uträkning av fakultet (n!)

Demo

- Uträkning av fakultet (n!)

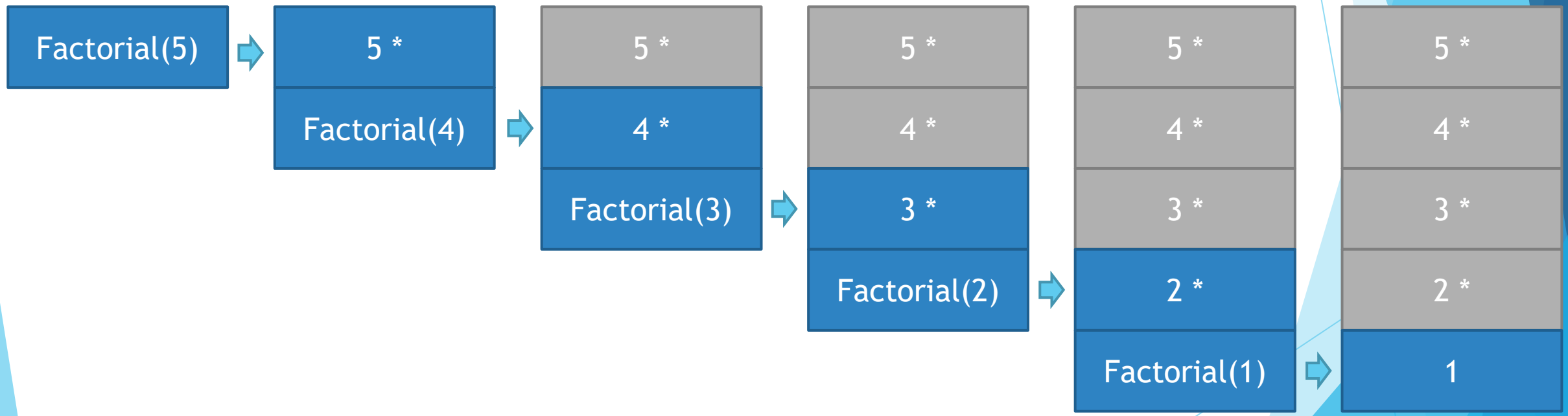
```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)


print(factorial(10))  # 3628800
```



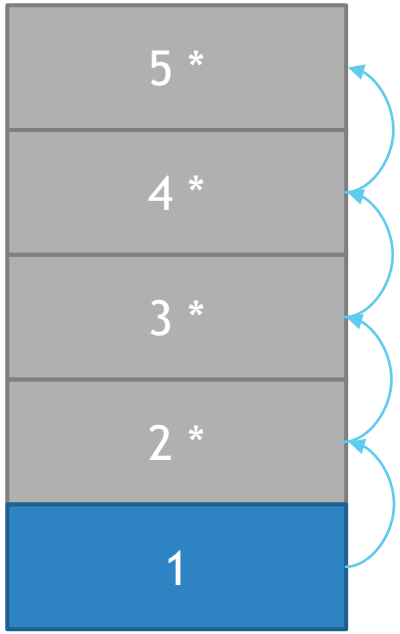
Grafisk representation

- factorial(5) = 5 * 4 * 3 * 2 * 1



Grafisk representation

- factorial(5) = 5 * factorial(4) = 5 * 4 * factorial(3) = 5 * 4 * 3 * factorial(2) …



De tre nycklarna till rekursion

1) Er kod måste ha ett fall för alla giltiga inputs
2) Du måste ha ett basfall som inte gör några rekursiva anrop
3) När du gör rekursiva anrop ska det vara till en förenkling som tar steg mot en lösning

Övning

- Skriv en rekursiv funktion som summerar alla tal i en lista
  - sum_list([5, 4, 2, 7, 1]) # Ska bli 19

- Skriv en rekursiv funktion som räknar ut "x upphöjt till n" (x ^ n) (power)
  - Alltså: 5^3 == 5 * 5 * 5
  - 4^6 = 4 * 4 * 4 * 4 * 4 * 4
  - power(4, 6) # Ska bli 4096

- Identifiera ett basfall
- Hur tar ni ert problem ett steg närmare basfallet?

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):
    if n < 10:
        return n
    else:
        a = n // 10
        b = n % 10
        return mystery(a + b)
```

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):  # n = 648
    if n < 10:
        return n
    else:
        a = n // 10  # a = 64
        b = n % 10   # b = 8
        return mystery(a + b) # mystery(72)
```

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):  # n = 72
    if n < 10:
        return n
    else:
        a = n // 10  # a = 7
        b = n % 10   # b = 2
        return mystery(a + b) # mystery(9)
```

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):  # n = 9
    if n < 10:  # return 9
        return n
    else:
        a = n // 10
        b = n % 10
        return mystery(a + b)
```

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):  # n = 72
    if n < 10:
        return n
    else:
        a = n // 10  # a = 7
        b = n % 10   # b = 2
        return mystery(a + b) # return 9
```

Övning

- Spåra funktionen: Vad blir svaret av mystery(648)?
  - A: 8
  - B: 9 ←
  - C: 54
  - D: 72
  - E: 648

```
def mystery(n):  # n = 648
    if n < 10:
        return n
    else:
        a = n // 10  # a = 64
        b = n % 10   # b = 8
        return mystery(a + b) # return 9
```

Tail-rekursion

- Ta ett exempel: Räkna ut summan av alla tal från N till 1.
  - Ex: N = 10
  - 10 + 9 + … + 2 + 1 = 55
- Istället för att göra som vi gjorde förut, vad händer om vi skickar med en räknare i rekursionen?
- Demo: Tailrekursion av summa

Tail-rekursion

- mysum(0, 10)
- mysum(10, 9)
- mysum(19, 8)
- mysum(27, 7)
- mysum(34, 6)
- mysum(40, 5)
- mysum(45, 4)
- mysum(49, 3)
- mysum(52, 2)
- mysum(54, 1) = 55

```
def mysum(the_sum, number):
    if number == 1:
        return the_sum + 1
    else:
        new_sum = the_sum + number
        return mysum(new_sum, number - 1)


print(mysum(0, 10))
```

Tailrekursion

- Gör anropet till rekursion helt fristående från andra rader -> Tail-rekursion
- Ett sätt att få sista rekursionen att skicka tillbaka svaret direkt
- Istället för att gå hela vägen upp igen

Övningar

- Skriv en rekursiv funktion som räknar ut fakultet med tail-rekursion
  - tail_factorial(product=1, num=5) ska bli 120
- Vad är maximala antal rekursioner som python tillåter?

Extra Övningar

- Skriv en rekursiv funktion som vänder på en lista
  - reverse_list [5, 4, 3, 2, 1] -> [1, 2, 3, 4, 5]

- Skriv en rekursiv funktion som
  - Givet en länkad lista
  - Skriver ut listan i ordningen från slutet till början
  - Ex: 1 -> 2 -> 3 -> 4 -> 5
    - Skriver ut 5, 4, 3, 2, 1
    - (Kan skriva ut en siffra per rad)

Inlämning 3

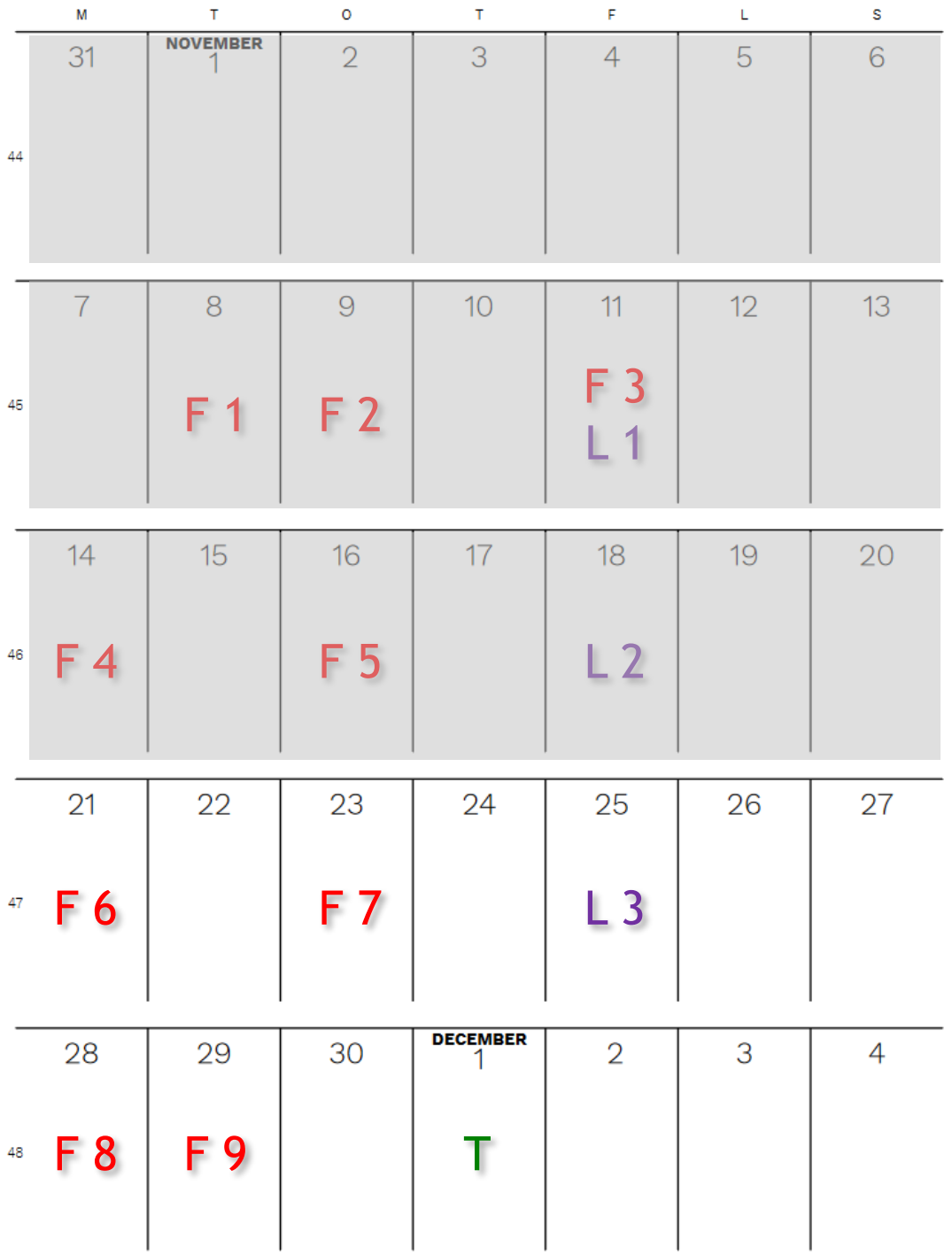
- Sökning i filer
- Se detaljer på Omniway
- Deadline på fredag kl 23:59
- Länk till testfiler på GitHub

Avslutning

- Reglerna
  1) Er kod måste ha ett fall för alla giltiga inputs
  2) Du måste ha ett basfall som inte gör några rekursiva anrop
  3) När du gör rekursiva anrop ska det vara till en förenkling som tar steg mot en lösning
- Rekursion kan ersätta iteration i vissa fall

Översikt

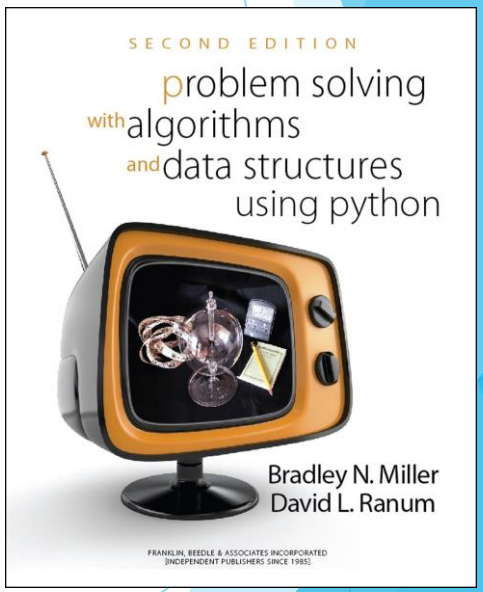
- F6: Rekursion
- F7: Träd och Grafer
- F8-F9: Repetition



Mer läsning

Bok

- **Problem Solving with Algorithms and Data Structures Using Python**
- Länk



Dagens material: Kapitel 4

Nästa gång: Träd och Grafer

- Om man vill läsa i förväg:
  - Kapitel 6 – Trees
  - Kapitel 7 - Graphs