Why did the Italian chef fail his web coding class? ...
He wrote spaghetti from scratch.

ROK CAPUDER, ZAVOD 404

MATIJA LOKAR, UL FAKULTETA ZA MATEMATIKO IN FIZIKO

Danes na meniju:

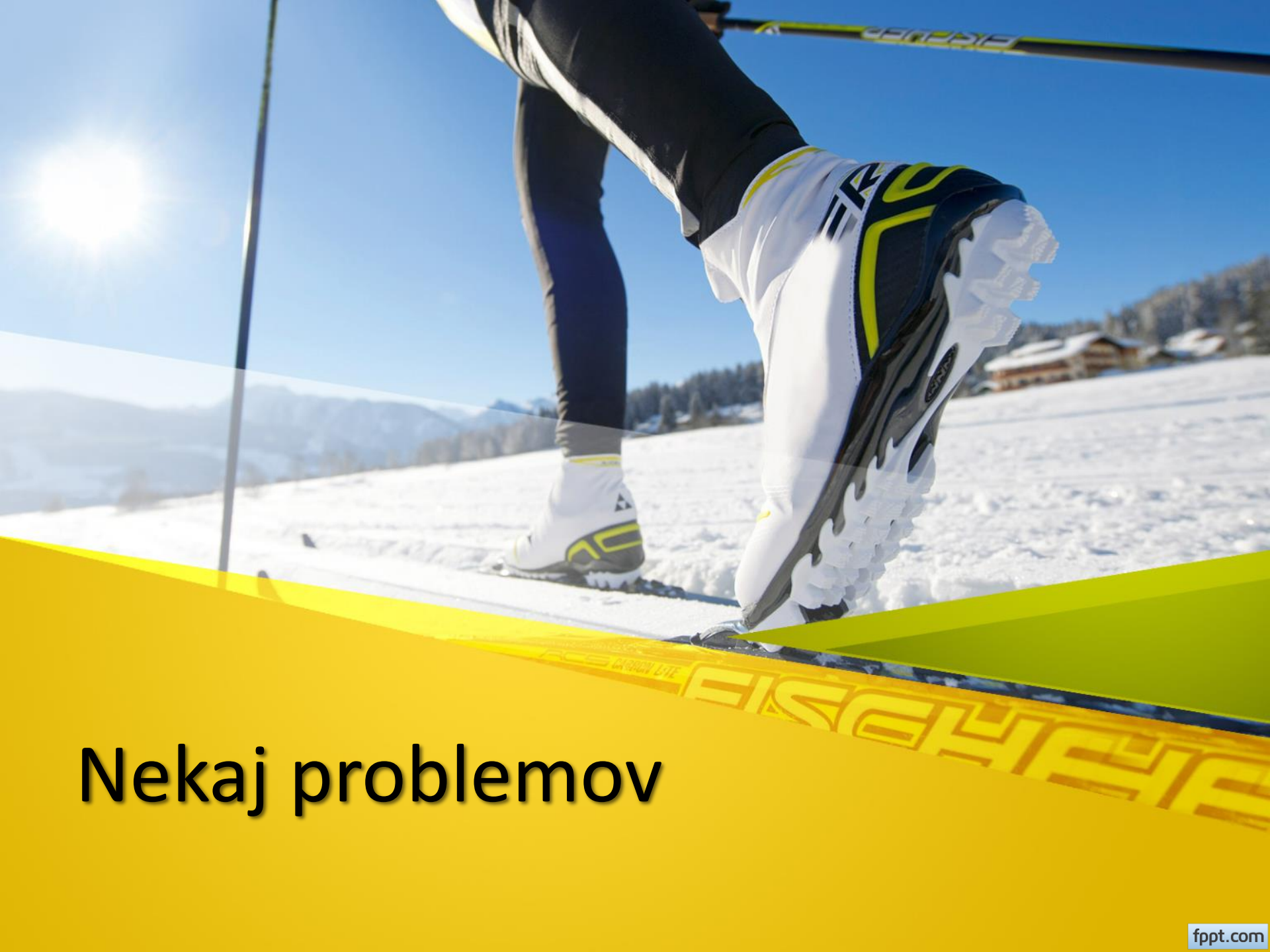
- Projekt NAPOJ
- Skupnost učiteljev računalniških predmetov
- Fizično programiranje
- Dinamično programiranje

NAPOJ (MATIJA L.)

SKUPNOST UČITELJEV (MATIJA L.)

FIZIČNO RAČUNALNIŠTVO (ROK C.)

DINAMIČNO PROGRAMIRANJE (MATIJA L.)



Nekaj problemov

Mošnjički z denarjem

Kapitan Kljuka se je odločil nagraditi svojega zvestega krmarja Enookega. Poklical ga je v kabino, kjer je na mizi v vrsti čakalo 10 mošnjičkov z zlatniki. Na vsakem mošnjičku je pisalo, koliko kovancev je v njem.

Enooki lahko pobere kolikor hoče mošnjičkov, le nikoli ne sme vzeti obeh sosednjih. Katere naj pobere, da si bo lahko kupil novo stekleno oko in mu bo ostalo še za rum!

Skupni podnizi

- Podana imamo niza X in Y, dolžin n in m.
- Poiščimo najdaljši, ne nujno strnjeni, skupni podniz.
  - Podniz je poljubno zaporedje črk iz nekega niza, ki so v istem relativnem vrstnem redu, kot so bile podane v prvotnem nizu
- **KONKURENCA, KOMUTATIVNO**
- **Rezultat:**
  - **KOUA**

Učiteljeva dilema

Imamo 480 minut časa in 6 opravil (popravljanje domačih nalog, vpis manjkajočih …). Za vsako poznamo čas trajanja in pomembnost, da jo opravimo. Vsako opravilo, ki ga začnemo, moramo opraviti do konca.

Katera izbrati?

Množenje matrik

- $M_1 \times M_2 \times M_3 \ldots \times M_n$
- Dimenzije so seveda ustrezne
- V kakšnem vrstnem redu množiti, da bomo opravili kar se da malo množenj realnih števil

Množenje 5 matrik

A1*A2*A3*A4*A5

A1: 30 ❊ 35

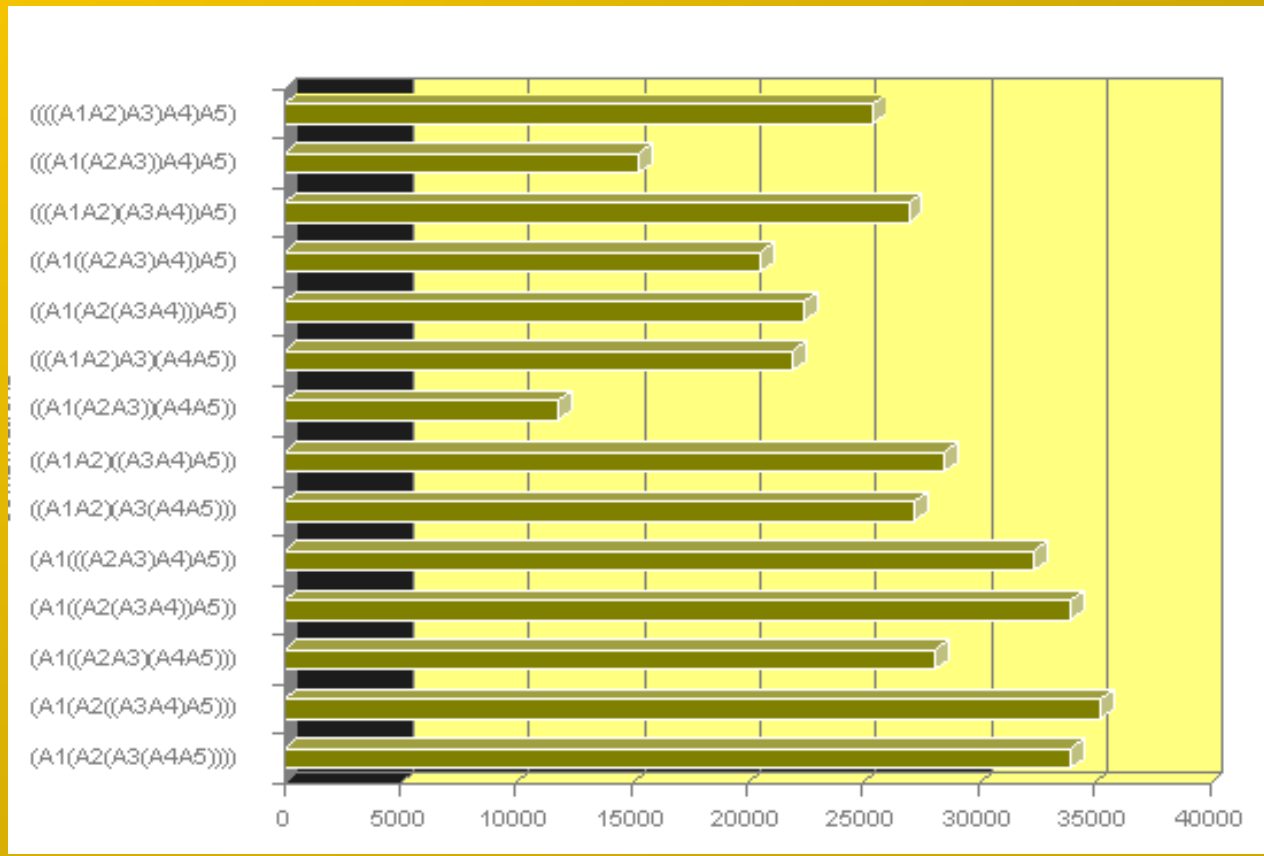
A2: 35 ❊ 15

A3: 15 ❊ 10

A4: 10 ❊ 5

A5: 5 ❊ 10

14 načinov!



Nedeljski planinec

Vsak nedeljski planinec se prej ali slej sooči z dejstvom, da bi rad v gore odnesel neverjetne količine hrane in pijače, teža, ki jo lahko nese, pa je zelo omejena - denimo 10 kilogramov.

Da bo lažje ocenil kaj naj vzame s seboj, je naš nedeljski planinec naredil seznam stvari, ki jih ima na razpolago, in jih ocenil glede na pomembnost. Kako naj jih izbere (seveda mora vzeti bodisi cel predmet, bodisi nič), da njihova skupna teža ne bo presegala 10 kilogramov in da bo njihova skupna pomembnost kar največja?

- Cviček, 4 kg, pomembnost 8
- Radenska, 3 kg, pomembnost 5
- Gosti sok, 1 kg, pomembnost 1.5
- Pršut, 5 kg, pomembnost 4
- Čelada, 0.5 kg, pomembnost 2
- Prva pomoč, 1 kg, pomembnost 1
- sir Jošt, 2 kg, pomembnost 6
- Sladkarije, 7 kg, pomembnost 10
- Komplet za varovanje, 3.5 kg, pomembnost 4
- Rženi kruh, 1.5 kg, pomembnost 5

Najkrajše poti

Učenci bi radi za celotno šolo izračunali, kako lahko najhitreje pridemo iz ene učilnice v drugo.

OPTIMIZACIJSKI PROBLEMI

DINAMIČNO PROGRAMIRANJE

Dinamično programiranje

- Osnovna lastnost dinamičnega programiranja = temelji na pravilu optimalnosti:
  - *Rešitev je optimalna, če so rešitve podproblemov zase optimalne*.
- Bellmanova enačba (rekurzivna izražava rešitve)

ENOSTAVNA REKURZIVNA FORMULACIJA

Primera

MOŠNJIČKI, POTI NA ŠOLI

SPROGRAMIRAMO

- In "ne gre"
  - Predlogo časa …
  - Razen, če je problem "majčken"
- Prepletanje podproblemov
- Rešitev:
  - Memoizacija (pomnjenje rešitev podproblemov)

REKURZIVNA DEFINICIJA NE POMENI NUJNO, DA MORAMO ALGORITEM ZAPISATI REKURZIVNO

Fibbonacijevo zaporedje

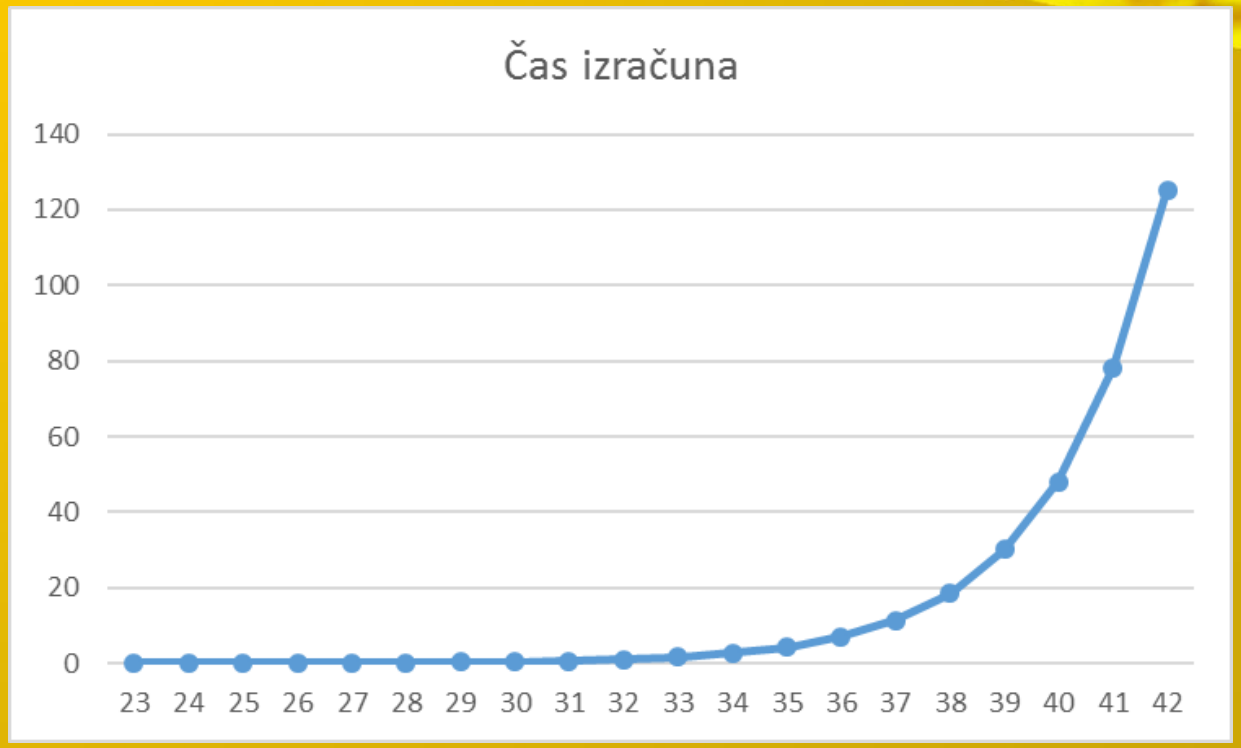
- $F_0 = 1\ F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$

```
def FibRek(n):
    '''Fibonacci s klasično rekurzijo
'''
    if n == 0: return 1
    if n == 1: return 1
    return FibRek(n-2) + FibRek(n-1)
```

```
for n in range(10, 100):
    # merimo čas
    zacRek = time.process_time()
    rezRek = FibRek(n)
    konRek = time.process_time()
    print(str(n) + ' : ', rezRek, '\t\t\t', konRek - zacRek)
```

Fibbonacijevo zaporedje

23 : 46368 0.015625
24 : 75025 0.03125
25 : 121393 0.03125
26 : 196418 0.0625
27 : 317811 0.078125
28 : 514229 0.15625
29 : 832040 0.234375
30 : 1346269 0.390625
31 : 2178309 0.625
32 : 3524578 1.046875
33 : 5702887 1.640625
34 : 9227465 2.625
35 : 14930352 4.265625
36 : 24157817 6.921875
37 : 39088169 11.34375
38 : 63245986 18.375
39 : 102334155 30.078125
40 : 165580141 47.96875
41 : 267914296 78.109375
42 : 433494437 125.203125
43 : 701408733 202.296875
44 : 1134903170 325.84375
45 : 1836311903 529.5625



Zakaj tako naraščanje

- Koliko klicev rekurzije imamo?
- T(0) = 1
- T(1) = 1
- T(n) = T(n-1) + T(n-2) + 1

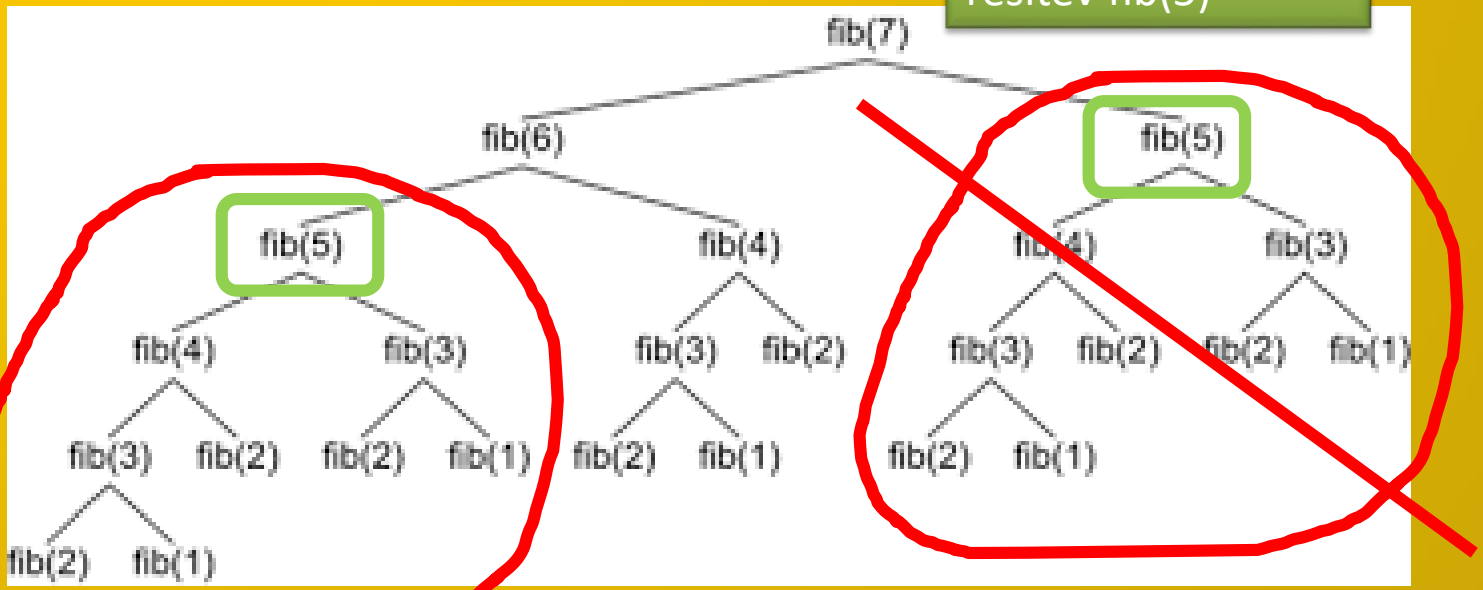
- $T(n) = 2F_{n+1} - 1$
- Izračun F_n zahteva 2x toliko korakov kot če bi kar šteli do F_n
- In ker "vemo", da $\lim_{n\to\infty}\left(\frac{F_{n+1}}{F_n}\right) = \varphi$, kjer je $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.618034$

Zakaj?

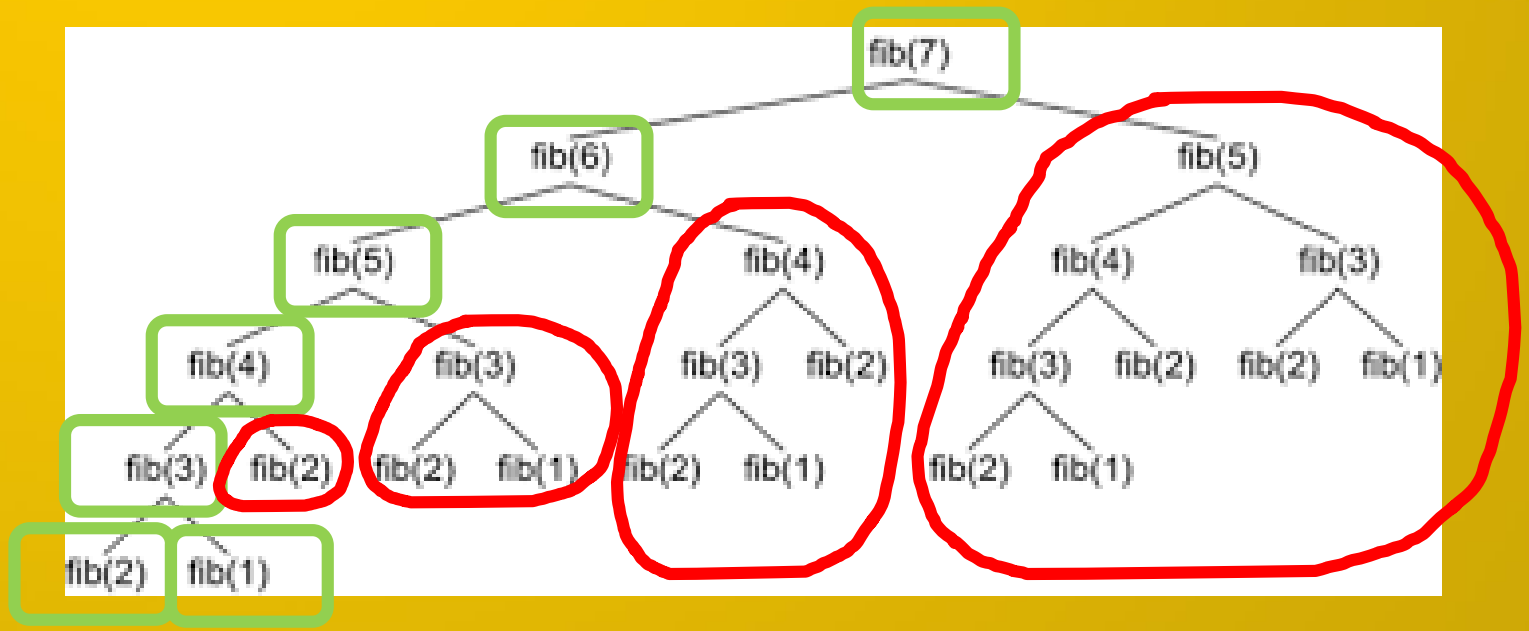
- Rekurzija je "neumna"
- Veliko dela opravimo prevečkrat!

Drevo klicev

Če smo shranili rešitev fib(5)



Drevo klicev



Kako "spametovati" rekurzijo

- "Memoizacija"

```
glCache = dict()

def FibMem(n):
    ''' s pomočjo globalnega slovarja glCache'''
    if n == 0:
        return 1
    if n == 1:
        return 1
    if n in glCache:
        return glCache[n]
    # ni ga v slovarju
    rez = FibMem(n-2) + FibMem(n-1)
    glCache[n] = rez # si ga zapomnimo
    return rez
```

Bolj programersko

```
def Fibonnaci(n):
    '''Hitrejša različica Fibonaccijja'''
    def FibMem(n): # lokalna funkcija
        ''' s pomočjo slovarja glCache iz "nosilne funkcije"'''
        if n == 0 or n == 1:
            return 1
        if n in glCache:
            return glCache[n]
        # ni ga v slovarju
        rez = FibMem(n-2) + FibMem(n-1)
        glCache[n] = rez # si ga zapomnimo
        return rez
    # gl. fun
    glCache = dict()
    return FibMem(n)
```

In rezultati …

23 : 46368	0.015625	* memoizacija * 1.078125e-05
24 : 75025	0.015625	* memoizacija * 1.125e-05
25 : 121393	0.046875	* memoizacija * 1.171875e-05
26 : 196418	0.046875	* memoizacija * 1.234375e-05
27 : 317811	0.09375	* memoizacija * 1.28125e-05
28 : 514229	0.140625	* memoizacija * 1.3125e-05
29 : 832040	0.234375	* memoizacija * 1.359375e-05
30 : 1346269	0.390625	* memoizacija * 1.40625e-05
31 : 2178309	0.640625	* memoizacija * 1.46875e-05
32 : 3524578	1.015625	* memoizacija * 1.546875e-05
33 : 5702887	1.671875	* memoizacija * 1.609375e-05
34 : 9227465	2.734375	* memoizacija * 1.625e-05
35 : 14930352	4.390625	* memoizacija * 1.65625e-05
36 : 24157817	6.921875	* memoizacija * 1.6875e-05

In še nekaj …

```
Fibonacci: 42
Rekurzija:                                                  433494437          124.0625
memoizacija - izvedemo 10000 ponovitev, a "goljufamo":      433494437          0.0
memoizacija - izvedemo 10000 ponovitev in "pošteno":        433494437          2.1875e-05
memoizacija - izvedemo 10000 ponovitev in "brez packanja":  433494437          2.03125e-05
>>> %Run Fibonacci.py
Fibonacci: 43
Rekurzija:                                                  701408733          203.46875
memoizacija - izvedemo 10000 ponovitev, a "goljufamo":      701408733          1.5625e-06
memoizacija - izvedemo 10000 ponovitev in "pošteno":        701408733          2.1875e-05
memoizacija - izvedemo 10000 ponovitev in "brez packanja":  701408733          2.1875e-05
>>> %Run Fibonacci.py
Fibonacci: 44
Rekurzija:                                                  1134903170         326.421875
memoizacija - izvedemo 10000 ponovitev, a "goljufamo":      1134903170         0.0
memoizacija - izvedemo 10000 ponovitev in "pošteno":        1134903170         2.34375e-05
memoizacija - izvedemo 10000 ponovitev in "brez packanja":  1134903170         2.1875e-05
>>> %Run Fibonacci.py
Fibonacci: 45
Rekurzija:                                                  1836311903         527.0625
memoizacija - izvedemo 10000 ponovitev, a "goljufamo":      1836311903         0.0
memoizacija - izvedemo 10000 ponovitev in "pošteno":        1836311903         2.34375e-05
memoizacija - izvedemo 10000 ponovitev in "brez packanja":  1836311903         2.34375e-05
```

A z iteracijo gre načeloma še hitreje …

```
def Fibonnaci(n):
    '''Iterativna različica Fibonaccijja'''
    nti, nasl = 1,1
    for _ in range(n):
        nti, nasl = nasl, nti + nasl
    return nti
```

Memoizacija

- CSharp
  - CSharp
  - CSharp 2
  - CSharp (napredno):
    - osnova iz tega članka
- Python
  - s slovarjem ...
  - uradno
  - kratko ...
  - dobra razlaga
  - slovar, potem ...