Pseudokódy – Časová zložitosť v najhoršom prípade

Bc. Katarína Olejková



KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Príklad 1 - JeKladne(n)

- 1. JeKladne(n)
- 2. **if** n > 0
- 3. print("áno")
- 4. else
- 5. print("nie")

```
1x porovnanie

1x print – buď "áno" alebo "nie"

(podľa toho ako dopadne podmienka if)
```

Príklad 2 – SudeNeboLiche(n)

- SudeNeboLiche(n)
- 2. **if** n <mark>mod</mark> 2 <mark>=</mark> 0
- 3. print("sudé")
- 4. else
- 5. print("liché")

```
1x porovnanie

1x operácia modulo

1x print – buď "sudé" alebo "liché"

(podľa toho ako dopadne podmienka if)
```

Príklad 3 – Signum(n)

- 1. Signum(n)
- 2. **if** n > 0
- 3. return 1
- 4. **else if** n <mark>=</mark> 0
- 5. return 0
- 6. else
- 7. return -1

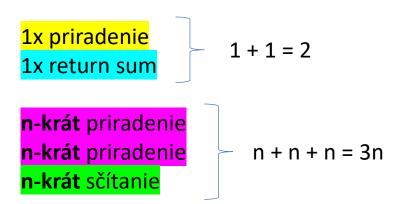
Pozn. ak by na vstupe bolo kladné číslo a platila by podmienka if n > 0 tak by šlo o **nejlepší prípad** lebo by sa vykonali iba dve operácie – porovnanie a return 1 (T(n) = 2)

My ale riešime časovú zložitosť v najhoršom prípade

V oboch prípadoch ale môžeme povedať, že sa jedná o konštantnú časovú zložitosť

Príklad 4 – Sucet(n)

- 1. Sucet(n)
- 2. sum <mark>←</mark> 0
- 4. $sum \leftarrow sum + i$
- 5. return sum



$$T(n) = 3n + 2$$

Lineárna čas. slož.

n-krát

n-krát

n-krát

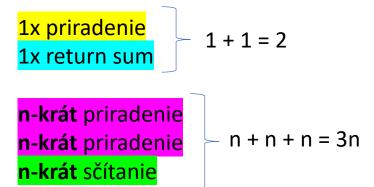
Príklad 5 – Nasobilka(n)

```
1. Nasobilka(n)
```

- 4. soucin <mark>←</mark> i <mark>*</mark> j
- 5. print</mark>(i '*' j '=' soucin)

Príklad 6 – SucetPole(A[0..n-1], n)

- SucetPole(A[0..n-1], n)
- 2. sum \leftarrow 0
- 4. $sum \leftarrow sum + A[i]$
- 5. return sum



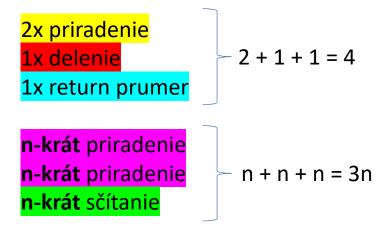
$$T(n) = 3n + 2$$

Lineárna čas. slož.

n-krát

Príklad 7 – AritmetickyPrumer(A[0..n-1], n)

- 1. AritmetickyPrumer(A[0..n-1], n)
- 2. sum \leftarrow 0
- 4. sum <mark>←</mark> sum <mark>+</mark> A[i]
- 5. prumer \leftarrow sum / n
- 6. return prumer



$$T(n) = 3n + 4$$

Lineárna čas. slož.

n-krát

Dobrovolný 1 – Znamka(n)

```
Znamka(n)
       if n ≥ 90 and n ≤ 100
          print('A')
      else if n ≥ 80 and n ≤ 89
          print('B')
       else if n \ge 70 and n \le 79
          print('C')
      else if n ≥ 60 and n ≤ 69
8.
          print('D')
9.
       else if n \ge 50 and n \le 59
10.
          print('E')
11.
12.
       else
          print('F')
13.
```

```
10x porovnanie

5x logicke and

1x print – bud' 'E' alebo 'F'

(podľa toho ako dopadne podmienka else if n \ge 50 and n \le 59)
```

T(n) = 16 Konštantná čas. slož.

Dobrovolný 2 – PocetLichychASudych(A[0..n-1], n)

PocetLichychASudych(A[0..n-1], n) liche \leftarrow 0 sude \leftarrow 0 if $A[i] \mod 2 = 0$ sude \leftarrow sude + 1 6. n-krát else liche — liche + 1 8. print("pocet sudych: " sude)

print("pocet lichych: " liche)

10.

- 2x priradenie 2x print
- n-krát priradenie
 n-krát modulo
 n-krát porovnanie
 n + n + n = 3n

Podľa toho ako dopadne podmienka tak inkrementujeme buď sude alebo liche, nikdy nie oboje.

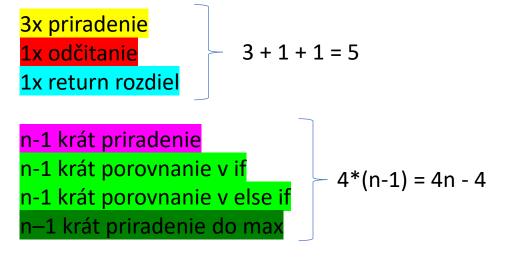
Riadok 6 a 8 sa dokopy vykoná n-krát

$$T(n) = 5n + 4$$

Lineárna čas. slož.

Dobrovolný 3 – MinMaxRozdiel(A[0..n-1], n)

```
1. MinMaxRozdiel(A[0..n-1], n)
      min \leftarrow A[0]
      \max \leftarrow A[0]
      if A[i] < min
            min \leftarrow A[i]
6.
                               n – 1 krát
         else if A[i] > max
            max
      rozdiel ← max ■ min
       return rozdiel
10.
```



Pozn. v najhoršom prípade if podmienka nie je splnená a kontroluje sa ďalšia podmienka v else if, ktorá bude pravdivá a prebehne

T(n) = 4n + 1

Lineárna čas. slož.

priradenie do max

Keby if podmienka platila tak by sme mali o n – 1 operácii (porovnanie v else-if) menej