Mentor Klub

Python tanfolyam

12. alkalom

Algoritmus

- Egy tetszőleges feladat megoldásához szükséges lépéseket leiró utasitássorozatot (receptet) algoritmusnak nevezünk.
- Egy algoritmus nem feltétlenül képletekből áll, néha szöveges megfogalmazással is megadható. (Az iparban is)
- A program vagy forráskód az nem algoritmus, hanem egy algoritmus adott nyelven történő megvalósítása.
- Az algoritmus kigondolását, tervezését algoritmizálásnak nevezzük, ami tervezői munka.

Algoritmizálás

- Minden esetben szükséges?
  - Léteznek nevezetes feladatok, amire mások már megalkották az algoritmust. Ilyenkor ne álljunk neki job algoritmust keresni, hacsak nem az a feladat.
    - Például ha az a feladat, hogy adjunk össze számokat egy listában, arra tudjuk az algoritmust, nem kell rajta gondolkozni, csak le kell kódolni.
- De az algoritmus megirása nem azonos a feladat megértésével.
- Lépések:
  1. Feladat vagy probléma megértése
  2. Visszavezethető-e a feladat egy már ismert algoritmusra?
  3. Ha nem, akkor algoritmus megtervezése (papír, ceruza)
  4. Algoritmus tesztelése (papír, ceruza)
  5. Algoritmus adott programnyelven történő megvalósítása (számítógép)
  6. Program tesztelése

Algoritmusok fajtái

- Az algoritmusokat, melyekben valamilyen ismétődő műveletet kell végrehajtani (általában ilyenek), két nagy csoportba szoktuk sorolni, attól függően, hogy az ismétlődést hogyan valósitjuk meg:
- Iteratív algoritmus:
  - Az ismétlődő lépéseket ciklusok segitségével oldjuk meg.

- Rekurzív algoritmus:
  - Az ismétlődést egy olyan függvény segitségével valósitjuk meg, mely saját magát hívja.
    - Igen, ezt egy kicsit bonyolult elképzelni…
    - Igen, ezektől mindenki fél egy kicsit…

Faktoriális számítás

- Feladat: Számoljuk ki egy természetes szám faktoriálisát.
- Feladat megértése:
  - 0! = 1
  - 1! = 1
  - 2! = 1 * 2
  - 3! = 1 * 2 * 3
  - n! = 1 * 2 * …. * n
- Ránézésre, ciklussal könnyen meg lehet oldani a feladatot, mert a számokat 1-től össze kell szorozni egymással. Ez olyan, mint az összegzés, csak szorzással.

Faktoriális számítás – iteratív algoritmus

- Az összegzés algoritmusát ciklussal ismerjük:
  - Legyen egy akkumulátor, kezdetben 0 → szorzásnál ez 1 legyen
  - Vegyük sorban a számokat, és adjuk őket össze → szorozzuk őket össze.

```
def faktorialis_iterativ(n):
    result = 1
    for i in range(1,n+1):
        result *= i

    return result
```

Faktoriális számítás – rekurzív algoritmus

Mentor Klub

- Ez bonyolultabb, mert másképpen kell elképzelni a helyzetet.
- Ha “elég sokáig nézzük” a feladatot, akkor feltűnhet, hogy
  - faktorialis(5) = 5 * faktorialis(4)
  - faktorialis(4) = 4 * faktorialis(3)
  - faktorialis(1) = 1

- Tehát találtunk egy olyan mintát, ami ugyanannak a függvénynek a hívásán alapszik, csak más paraméterekkel.

Faktoriális számítás – rekurzív algoritmus

- Egy olyan függvényt kell készíteni, ami saját magát hívja, csak mindig egyel kissebb paraméterrel, és összeszorozza a bemeneti paramétert az eredménnyel.
- Rekurzív leszállásnak is szokták hivni.

```
def faktorialis_rekurzivan(n):
    if n <= 1:
        return 1
    return n * faktorialis_rekurzivan(n-1)
```

Hogyan képzeljük ezt el?

Mentor Klub

- Nincs végtelen rekurzió.
  - Egy rekurzív függvény kötelező eleme a **kilépési feltétel**, azaz egy olyan ág, ami nem fog újabb függvényhivást eredményezni.
  - Mivel minden függvényhivás egy újabb elem a végrehajtási veremben, ezért ha nem lenne kilépési feltétel, elfogyna a memória.
    - Ezt úgy hivják, hogy Stack Overflow (nem a weboldal ☺)
- A kilépési feltétel teljesülésekor a legbelső függvény visszatér, ami miatt az azt hivó függvény is vissza tud térni, és igy tovább...
- Tehát először “leszáll” az algoritmus a legelemibb részig, ahonnan már nincs tovább, majd visszafele kiszámolódik az egész.

Mi a nehéz a rekurzióban? A rekurzió

Fibonacci sorozat

- Feladat: Számítsuk ki a Fibonacci sorozat n-edik elemét.
- Fibonacci sorozat: 0,1,1,2,3,5,8,13,21,…..
- A Fibonacci sorozat első 3 eleme: 0,1,1
- A Fibonacci sorozat n-edik eleme (n > 3): A sorozat előző két elemének az összege
  - 0
  - 1
  - 1
  - 1+1 = 2
  - 1+2 = 3
  - 2+3 = 5

Fibonacci sorozat – iteratív algoritmus

```
def fibonacci_iterativ(n):
    sorozat = [0, 1, 1]

    if n > 3:
        for i in range(3, n):
            kovetkezo = sorozat[i-1] + sorozat[i-2]
            sorozat.append(kovetkezo)
     return sorozat[n-1]
```

Fibonacci sorozat – rekurzív algoritmus

- Az elgondolás hasonló a faktoriális számításhoz.
- Az n-edik Fibonacci szám nem más, mint az n-1 és az n-2 Fibonacci szám összege.
- Kilépési feltétel?
  - Az első 3 számot nem kell számolnunk, tehát azokat csak egyszerűen visszaadhatja a függvény.
- Probléma: Nem hatékony, hiszen kétszer számol…

```
def fibonacci_rekurziv(n):
    if n <= 1:
        return n
    else:
        return(fibonacci_rekurziv(n-1) + fibonacci_rekurziv(n-2))
```

Mentor Klub

KÖSZÖNÖM SZÉPEN A FIGYELMET ÉS AZ AKTIV RÉSZVÉTELT!