Minimum/maximum kiválasztás

Ionescu Klára

clara@cs.ubbcluj.ro

Bevezetés

- A feladatok ***feladatosztályok***ba sorolhatók a jellegük szerint.
- E feladatosztályokhoz készítünk a teljes feladatosztályt megoldó ***algoritmusosztály***t.
- Ezeket az algoritmusosztályokat ***programozási tétel***eknek nevezzük (lásd 1. konzultáció).
- Bebizonyítható, hogy a megoldások a feladat ***garantáltan helyes és optimális*** megoldásai.
- A bemenet és a kimenet szerint négy csoportra oszthatók:
  - sorozathoz érték rendelése (**1 sorozat – 1 érték**)
  - sorozathoz sorozat rendelése (**1 sorozat – 1 sorozat**)
  - sorozatokhoz sorozat rendelése (**több sorozat – 1 sorozat**)
  - sorozathoz sorozatok rendelése (**1 sorozat – több sorozat**)

Egyszerű feladatok

1. Egy kórházban megmérik minden reggel a betegek hőmérsékletét. Határozzuk meg a ***leglázasabb*** beteg hőmérsékletét!
2. Egy táblázatban feltüntették az elsőéves informatika szakos egyetemisták nevét a felvételi után, a vizsgaátlaguk sorrendjében. Határozzuk meg annak az egyetemistának a nevét, akivel ***az ábécé sorrendű táblázat kezdődni fog***!
3. A szociális ösztöndíj kiosztása céljából összegyűjtötték az egyetemisták családjainak jövedelmére vonatkozó adatokat. Határozzuk meg a ***legkisebb*** jövedelmet!

Elemzés

- A megoldások hasonlóak lesznek, mivel mindegyikben meg kell határoznunk az adatok között található ***legnagyobb***, illetve ***legkisebb*** értéket.
- Előfordulhat, hogy több ilyen legnagyobb, illetve legkisebb elem létezik.
- A **Kiválasztás(n, X, sorszám)** programozási tétel sajátos esetével állunk szemben, amikor a keresett elem az adott sorozat legkisebb/legnagyobb értékű eleme.
- Az algoritmus mindig **Minden** típusú struktúrával dolgozik, hiszen minden adatot meg kell vizsgálnunk.

Maximumkiválasztás programozási tétel

- Adott az ***N*** elemű ***X*** sorozat. Határozzuk meg a sorozat ***legnagyobb*** (vagy legkisebb) értékét (vagy sorszámát)!

Elemzés

- Előfordulhat, hogy ***több*** legnagyobb elem létezik, de nekünk most az értéket (vagy az első előfordulás helyét) kell meghatároznunk.
- A megoldásban ***minden*** adatot meg kell vizsgálnunk, ezért az algoritmus, egy **Minden** típusú struktúrával dolgozik.
- A max segédváltozó a sorozat első elemétől kap kezdőértéket.

Algoritmus Maximumkiválasztás(N, X, max):

{ *Bemeneti adatok: az* N *elemű* X *sorozat* }

{ *Kimeneti adat:* max, *a legnagyobb elem értéke* }

{ *Funkció: meghatározza az* N *elemű* X *sorozat legnagyobb értékét* }

max ← X_1

Minden i = 2, n **végezd el**:

Ha max < X_i **akkor**

max ← X_i

vége(ha)

vége(minden)

Vége(algoritmus)

```
Algortmus maximumKiválasztásRek(n, X):
                                                    { rekurzív implementáció }
                                    { Bemeneti adatok: az N elemű X sorozat }
                                    Kimeneti adat: max, a legnagyobb elem értéke }
                    { Funkció: meghatározza az N elemű X sorozat legnagyobb értékét }
   Ha n = 1 akkor
        térítsd X1
   különben
        Ha Xn > maximumKiválasztásRek(n - 1, X) akkor
              térítsd Xn
        különben
              térítsd maximumKiválasztásRek(n - 1, X)
                                    { vizsgálni fogjuk a sorozat következő elemét }
        vége(ha)
   vége(ha)
Vége(algoritmus)
```

Az algoritmus hívása: **Ki:** maximumKiválasztásRek(n, X)

Észrevételek

- A maximumot/minimumot tartalmazó segédváltozónak ***az adatok közül választunk kezdőértéket***, mivel így nem áll fenn a veszély, hogy az algoritmus eredménye egy, az adataink között nem létező érték legyen.

- Ha az adatokat, amelyekből a maximumkiválasztást végezzük valamilyen számolás eredményeként kapjuk, akkor (ritkán) kereshetünk egy olyan értéket, amelynél biztos lesz nagyobb a kiszámolt értékek között.

Változat a maximumkiválasztásra

- Az előbbi algoritmus a maximum ***értékét*** határozza meg.
- Ha nem az érték, hanem a ***hely*** érdekel (index), ahol ***elsőként*** fedeztük fel a legnagyobb értéket, akkor az algoritmus a következőképpen alakul:

Algoritmus Maximum_helye(N, X, hely):

{ *Bemeneti adatok: az* N *elemű* X *sorozat* }

{ *Kimeneti adat:* hely, *a legnagyobb elem pozíciója* }

{ *Funkció: meghatározza a maximumérték első előfordulásnak helyét* }

hely ← 1 { hely *az első elem pozíciója* }

Minden i = 2, n **végezd el:**

Ha $X_{hely} < X_i$ **akkor**

hely ← i { *a maximális elem helye (pozíciója)* }

vége(ha)

vége(minden)

Vége(algoritmus)

```
Algoritmus maximumHelyeRek(n, X):
                                        { Bemeneti adatok: az N elemű X sorozat }
                                        { Kimeneti adat: hely, a legnagyobb elem pozíciója }
                                        { Funkció: meghatározza a maximumérték első előfordulásnak helyét }
                                        { rekurzív implementáció }
    Ha n = 1 akkor
        térítsd 1
    különben
        Ha Xn > XmaximumHelyeRek(n - 1, X) akkor
            térítsd n
        különben
            térítsd maximumHelyeRek(n - 1, X)
        vége(ha)
    vége(ha)
Vége(algoritmus)
```

Az algoritmus hívása: **Ki:** maximumHelyeRek(n, X)

Észrevétel

- Ha ***nem az első***, maximumot tároló helyet kell meghatároznunk, hanem az ***utolsót***, bejárhatjuk a ***sorozatot fordított*** irányban (az előző algoritmust használva).
- Alkalmazhatjuk az előbbi algoritmust úgy is, hogy a **„<”** relációs műveletet **„≤”**-re cseréljük.

Minden maximum helye

Határozzuk meg az ***összes*** olyan indexet, amely indexű elemek egyenlők a legnagyobb elemmel!

Első ötlet:

1. ***Megállapítjuk*** ***a maximum értékét***.
2. *Bejárjuk* ***újra*** *a sorozatot*, abból a célból, hogy kiírhassunk minden indexet, amelyhez ez a legnagyobb érték tartozik.

Algoritmus Minden_max(N, X, db, indexek):

{ *Bemeneti adatok: az* N *elemű* X *sorozat* }

{ *Kimeneti adat: a* db *elemű* indexek *sorozat* }

{ *Funkció: meghatározza minden legnagyobb elem helyét* }

max ← X_1

Minden i = 2, N **végezd el:**

Ha max < X_i **akkor** max ← X_i

vége(ha)

vége(minden)

db ← 0

Minden i = 1, N **végezd el:**

Ha max = X_i **akkor**

db ← db + 1

$indexek_{db}$ ← i

vége(ha)

vége(minden)

Vége(algoritmus)

2. ötlet

Ha ez a megoldás nem megfelelő, mivel

- az adott tömböt kétszer kell bejárnunk, vagy
- olyan feladatunk van, ahol ***a maximumhoz tartozó adatok egy másik (esetleg bonyolult) algoritmus végrehajtásának eredményei***,

$\Rightarrow$ írhatunk olyan algoritmust, amely ***csak egyszer járja be a sorozatot***:

Algoritmus Minden_max_2(N, X, db, indexek):

{ *Bemeneti adatok: az* N *elemű* X *sorozat* }

{ *Kimeneti adat: a* db *elemű* indexek *sorozat* }

{ *Funkció: meghatározza minden legnagyobb elem helyét* }

max ← X_1 db ← 1, $indexek_1$ ← 1

Minden i = 2, N **végezd el:**

Ha max < X_i **akkor**

max ← X_i

db ← 1

$indexek_{db}$ ← i

különben

Ha max = X_i **akkor**

db ← db + 1

$indexek_{db}$ ← i

vége(ha)

vége(ha)

vége(minden)

Vége(algoritmus)

```
Algoritmus maximumokHelyeRek(n, X, poz, db, max):
        Ha n ≥ 1 akkor
                Ha max < X_n akkor
                        poz_1 ← n
                        db ← 1
                        maximumokHelyeRek(n - 1, X, poz, db, X_n)
                különben
                        Ha max = X_n akkor
                                db ← db + 1
                                poz_db ← n
                                maximumokHelyeRek(n - 1, X, poz, db, max)
                        különben
                                maximumokHelyeRek(n - 1, X, poz, db, max)
                        vége(ha)
        vége(ha)
Vége(algoritmus)
```

Az algoritmus hívása: maximumokHelyeRek(n, X, poz, db, X_n), ahol előbb db ← 0