1 | Algoritmusok

Mit értünk algoritmus alatt?

Az algoritmus fogalmára létezik pontos, formális definíció, de ebben a tárgyban beérjük a következő, a gyakorlatban jól használható meghatározással. Az algoritmus egy

- mechanikusan végrehajtható utasítássorozat, ami
- adott formátumú inputból adott formátumú elvárt kimenetet állít elő, és
- minden megfelelő formátumú inputon véges sok lépésben véget ér.

Nézzünk néhány egyszerű példát olyan helyzetekre, amikor algoritmusra van szükségünk.

> **1.1. példa** (Legkisebb elem)**.**
> Adott véges hosszúságú, pozitív egész számokat tartalmazó sorozatban melyik a legkisebb elem?

> **1.2. példa** (Van-e 7-es szám?)**.**
> Adott véges hosszúságú, pozitív egész számokat tartalmazó sorozatnak eleme-e a 7-es szám?

Amikor egy algoritmust megadunk, akkor három dolgot kell megtennünk:

1. leírni az algoritmust (precízen szövegesen vagy pszeudokóddal),
2. megindokolni, hogy az algoritmus mindig véget ér és helyes kimenetet ad,
3. meghatározni, hogy hány lépést végez az algoritmus.

A lépésszám megbecslésekor a következő szempontokra fogunk figyelni:

- a lépésszám függ az input méretétől,
- minden feladatnál előre megmondjuk, hogy milyen típusú lépések lényegesek,
- legtöbbször elég egy „jó” felső becslést adni.

Tömbök

Egy n-méretű tömb alatt azt értjük, hogy van n darab cella, 0-tól $n-1$-ig megszámozva (megindexelve), és minden cellában egy érték szerepel. Az értékeket az indexük alapján egy lépésben ki tudjuk

olvasni. Az n-méretű T tömbre a $T[0:n-1]$ jelölést használjuk, az i-edik indexű cella értékére pedig $T[i]$ jelöléssel hivatkozunk.

Pszeudokód

A pszeudokód egy egyszerű, ember számára könnyen olvasható leírása egy algoritmusnak. Nem kötődik konkrét programozási nyelvhez, ugyanakkor követi a programozási logika alapvető szabályait. Az alábbiakban a pszeudokódok néhány alapvető elemét tekintjük át.

Algoritmus arra, hogy van-e 7-es szám egy tömbben

A következőkben az 1.2. példában látott feladatra adunk két algoritmust.

Szöveges algoritmus annak eldöntésére, hogy egy tömb elemei között van-e 7-es.

Végigmegyünk a tömbbön, és sorra minden cellában megnézzük, hogy 7-es van-e ott:

- ha látunk valahol 7-est, akkor megállunk és azt mondjuk, hogy `van`,
- ha végigérünk a tömbön anélkül, hogy látnánk 7-est, akkor azt mondjuk a végén, hogy `nincs`.

Pszeudokóddal leírt algoritmus annak eldöntésére, hogy egy tömb elemei között van-e 7-es.

1. algoritmus: Van-e 7-es a tömbben?

Input: egy $n \geq 1$ darab számot tartalmazó T tömb
Output: `van` vagy `nincs`

ciklus: i **fut** 0**-től** $(n-1)$**-ig**
  ha $T[i] == 7$**:**
    return "`van`"
ciklus vége
return "`nincs`"

Helyesség. Az algoritmus jó kimenetet ad, mert minden cellát egyesével megnézünk, és akkor és csak akkor adunk `van` kimenetet, ha találunk 7-es értéket. Az algoritmus biztosan véget ér, legkésőbb a tömb végén.

Lépésszám. Tekintsük lépésnek az összehasonlításokat, valamint az eredmény visszaadását. Legfeljebb n darab összehasonlítás történik (ha korábban találunk 7-est, akkor nem olvassuk végig a

tömböt) és a végén további egy lépés az eredmény közlése. Azaz az algoritmus legfeljebb $n+1$ lépést végez.

Fontos, hogy a lépésszám megbecslésekor nem a „legkönnyebben kezelhető” inputokon elvégzett lépések számát akarjuk meghatározni, hanem épp ellenkezőleg: egy olyan felső becslést keresünk, ami minden n-méretű inputra igaz.

Ennek a felső becslésnek nem feltétlenül muszáj „élesnek” lennie, azaz nem muszáj léteznie olyan inputnak, amin pontosan ennyi lépést tesz az algoritmus. Az 1. algoritmusra adott lépésszámbecslésünk éles, hiszen ha a tömbben nincsen 7-es vagy csak az utolsó eleme 7-es, akkor az algoritmusunk pontosan $n+1$ lépést fog tenni. De nyugodtan mondhattuk volna azt, hogy az 1. algoritmus legfeljebb $2n$ lépést tesz, hiszen tetszőleges $n \geq 1$ egész szám esetén $n + 1 \leq 2n$.

Érdekes kérdés lehet az alsó becslés is, azaz ebben a feladatban az, hogy létezik-e olyan algoritmus, ami minden $n \geq 2$ méretű tömbön mindig kevesebb, mint $n + 1$ lépést tesz. A válasz az, hogy nem, és ezt nem is lenne nehéz belátni. Viszont ilyen jellegű kérdésekkel ennek a tárgynak a keretén belül nem foglalkozunk részletesebben.

Algoritmus egy tömb legkisebb elemének megkeresésére

A következőkben az 1.2. példában látott feladatra adunk algoritmust.

2. algoritmus: Legkisebb elem keresése

Input: egy $n \geq 1$ darab különböző számot tartalmazó T tömb
Output: T legkisebb eleme

```
min := T[0]
ciklus: i fut 1-től (n − 1)-ig
    ha T[i] < min:
        min := T[i]
ciklus vége
return min
```

Helyesség. Az algoritmus jó kimenetet ad, mert minden cellát megnézünk, és amikor elérünk a legkisebb elemhez, akkor az bekerül a `min` változóba, később pedig már nem írjuk át ezt a változót. Az algoritmus biztosan véget ér, legkésőbb a tömb végén.

Lépésszám. Tekintsük lépésnek az összehasonlításokat, az értékadásokat, valamint az eredmény visszaadását. Az elején egy értékadás történik, majd $n - 1$ darab összehasonlítás és legfeljebb ennyi

értékadás, majd a végén további egy lépés az eredmény közlése. Azaz az algoritmus tehát legfeljebb $2n$ lépést tesz.

Feladat. Módosítsuk a 2. algoritmust úgy, hogy a legkisebb elem értékén kívül annak az indexét is kiírja.

Kérdés. Könnyebb lenne-e az 1.1. és az 1.2. példákban leírt feladat, ha egy növekvően rendezett tömböt kapnánk inputként?