Automata - A megoldás magyarázata

A megoldást szolgáltató kód az `automata.py` szkript fájlban található meg.

Első lépésben beolvassuk a sorozat elemeit az alábbi módon:

```
# beolvassuk a sorozat elemeit
sorozat = list(map(int, input().split(" ")))
```

Ezután létrehozunk egy `list` típusú (üres) változót, amelyben a feldolgozás során a sorozat elemeit fogjuk tárolni:

```
# ebben a listaban fogjuk tarolni a sorozat elemeit
sor = []
```

Mindezek mellett létrehozunk egy másik `list` típusú (szintén üres) változót, amelyben a kiírt elemeket fogjuk eltárolni a kiírás sorrendjében:

```
# ebben a listaban fogjuk tarolni a kiirt szamokat
# a kiiras sorrendjeben
kimenet = []
```

Mivel minden elem érkezése után meg kell adnunk a sorban lévő elemeket és a kiírt értékeket, így létrehozunk egy format sztringet, amely megkönnyíti majd a kiíratást:

```
# format sztring a kiiras megkonnyitesehez
aktualis_allapot = "{}. sor = {}, kimenet = {}"
```

Szeretnénk követni, hogy éppen hányadik elem feldolgozása történik, így létrehozunk egy segédváltozót is, amellyel meg tudjuk ezt is tenni:

```
# ciklusszamlalo
szamlalo = 0
```

Ezután elkezdjük feldolgozni a sorozat elemeit a beérkezés sorrendjében (vö. `for szam in sorozat`). Elsőként növeljük a ciklusszámláló értékét (vö. `szamlalo += 1`). Ezek után rendre végrehajtjuk azokat a lépéseket, amelyeket a feladatkiírás előírt számunkra. Tehát legelőszöris ha a sorozat üres (vö. `if not sor`), akkor a sor végére tesszük az aktuális számot (vö. `sor.append(szam)`). Különben ha a sor utolsójánál nagyobb az aktuális szám (vö. `elif sor[-1] < szam`), akkor végére tesszük (vö.

`sor.append(szam)`). Különben ha az utolsó kiírtnál kisebb az aktuális szám (vö. `elif kimenet and szam < kimenet[-1]`), akkor hibajelzéssel megáll (vö. `print("HIBA: Utolso kiirtnal kisebb szam.")` és `exit()`). Különben ha a sor elsőjénél kisebb az aktuális szám (vö. `elif szam < sor[0]`), akkor kiírjuk (vö. `kimenet.append(szam)`). Végül különben amíg a sor elsőjénél nagyobb az aktuális szám (vö. `while sor and sor[0] < szam`), addig kiírjuk a sor elsőjét (és a kiírt elemet ki is vesszük a sorból) (vö. `kimenet.append(sor[0])` és `del sor[0]`); ilyenkor, ha a sor üressé válik (vö. `if not sor`), akkor a bemeneti értéket a sorba tesszük (vö. `sor.append(szam)`), különben kiírjuk (vö. `kimenet.append(szam)`). Továbbá minden elem érkezése után megadjuk a sorban levő elemeket és a kiírt értékeket is (vö. `print(aktualis_allapot.format(szamlalo, sor, kimenet))`). A teljes vonatkozó kódrészlet így néz ki:

```
for szam in sorozat:
    szamlalo += 1

    # ha a sor ures, a vegere tesszuk az aktualis szamot
    if not sor:
        sor.append(szam)

    # ha a sor utolsojanal nagyobb az aktualis szam, akkor
    # az aktualis szamot a sor vegere tesszuk
    elif sor[-1] < szam:
        sor.append(szam)

    # ha az utolso kiirtnál kisebb az aktualis szam, akkor
    # hibajelzessel megallunk
    elif kimenet and szam < kimenet[-1]:
        print("HIBA: Utolso kiirtnal kisebb szam.")
        exit()

    # ha a sor elsojenel kisebb az aktualis szam, akkor
    # kiirjuk az aktualis szamot
    elif szam < sor[0]:
        kimenet.append(szam)
    else:
        # amig a sor elsojenel nagyobb az aktualis szam, addig
        # kiirjuk a sor elsojet (es a kiirt elemet ki is vesszuk a sorbol);
        # ilyenkor, ha a sor uresse valik, akkor a bemeneti erteket
        # a sorba tesszuk, kulonben kiirjuk
        while sor and sor[0] < szam:
            kimenet.append(sor[0])
            del sor[0]
        if not sor:
            sor.append(szam)
        else:
            kimenet.append(szam)

    # minden elem erkezése utan megadjuk a sorban levo elemeket
    # es a kiirt ertekeket
    print(aktualis_allapot.format(szamlalo, sor, kimenet))
```

Végül az utolsó érték feldolgozása után a sorban lévő elemeket kiírjuk az alábbiak szerint:

```
# az utolso ertek feldolgozasa utan a sorban levo elemeket kiirjuk
while sor:
    kimenet.append(sor[0])
    del sor[0]
```

1. feladat - megoldás

Ha lefuttatjuk a megadott bemenetre (vö. `feladatlap.md` és `test/input0.txt`) az `automata.py` szkriptet, akkor a következő kimenetet (vö. `test/output0.txt`) kapjuk, amelyet itt a magyarázó fájlban kiegészítettünk kommentbe helyezett magyarázatokkal is, hogy hol melyik szabály került alkalmazásra:

```
1. sor = [3], kimenet = []
# a sor ures volt, igy a vegere tettuk az aktualis szamot (3)
2. sor = [3, 4], kimenet = []
# a sor utolsojanal nagyobb az aktualis szam (4), igy az aktualis szamot a
sor vegere tettuk
3. sor = [3, 4, 6], kimenet = []
# a sor utolsojanal nagyobb az aktualis szam (6), igy az aktualis szamot a
sor vegere tettuk
4. sor = [3, 4, 6], kimenet = [1]
# a sor elsojenel kisebb az aktualis szam (1), igy kiirtuk az aktualis
szamot
5. sor = [3, 4, 6, 8], kimenet = [1]
# a sor utolsojanal nagyobb az aktualis szam (8), igy az aktualis szamot a
sor vegere tettuk
6. sor = [3, 4, 6, 8], kimenet = [1, 2]
# a sor elsojenel kisebb az aktualis szam (2), igy kiirtuk az aktualis
szamot
7. sor = [6, 8], kimenet = [1, 2, 3, 4, 5]
# amig a sor elsojenel nagyobb volt az aktualis szam (5), addig kiirtuk a
sor elsojet (es a kiirt elemet ki is vettuk a sorbol) (ld. 3 es 4); mivel a
sor nem valt uresse, igy az aktualis szamot kiirtuk
8. sor = [6, 8, 9], kimenet = [1, 2, 3, 4, 5]
# a sor utolsojanal nagyobb az aktualis szam (9), igy az aktualis szamot a
sor vegere tettuk
9. sor = [8, 9], kimenet = [1, 2, 3, 4, 5, 6, 7]
# amig a sor elsojenel nagyobb volt az aktualis szam (7), addig kiirtuk a
sor elsojet (es a kiirt elemet ki is vettuk a sorbol) (ld. 6); mivel a sor
nem valt uresse, igy az aktualis szamot kiirtuk
10. sor = [], kimenet = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# az utolso ertek feldolgozasa utan a sorban levo elemeket kiirtuk
```

2. feladat - megoldás

A megadott szabályok alapján jól látható, hogy az algoritmus akkor áll meg hibajelzéssel, ha a sorozat tartalmaz olyan `A_i`, `A_j`, `A_k` részsorozatot, ahol `i < j < k` és `A_k < A_j < A_i`, pl. 3 2 1 (vö.

`test/input1.txt`).