Factory Machines feladatmegoldás

Az alábbi kód célja annak meghatározása, hogy egy adott számú (t) termék legyártásához milyen minimális idő szükséges, ha különböző sebességű gyártó gépek állnak rendelkezésre. A megoldás egy bináris keresés algoritmust alkalmaz, amely hatékony a nagy keresési tartományban történő optimalizálásra. Függvénydefiníció:

```
1   def minimális_idő(n, t, gépek):
```

n: a gépek száma; t: az elkészítendő termékek száma; gépek: egy lista, amelyben minden elem azt az időt jelöli, amely egy adott gép számára szükséges egy termék legyártásához. Bináris keresés határainak meghatározása:

```
3       bal = 1
4       jobb = t * min(gépek)
5       válasz = jobb
```

bal: a minimális idő kezdeti feltételezése. Ez legalább 1, mivel 0 idő alatt nem készülhet termék; jobb: az idő maximális becslése. Ha a leglassabb géppel készítjük az összes terméket (azaz t * min(gépek) idő alatt), az biztosan elegendő lesz; válasz: a lehetséges minimális időt mentjük el ebbe a változóba. Bináris keresés ciklus:

```
7       while bal <= jobb:
8           közép = (bal + jobb) // 2
```

közép: az időintervallum középső pontja. A bináris keresés mindig ezt vizsgálja. Termékek számának kiszámítása adott idő alatt:

```
10          # Számoljuk meg, hány termék készül el adott idő alatt
11          összes_termék = sum(közép // k for k in gépek)
```

közép // k: egy gép adott idő alatt (közép) hány terméket tud legyártani. Például, ha egy gép egy terméket 3 egységnyi idő alatt készít el, akkor 12 időegység alatt 12 // 3 = 4 terméket tud előállítani; sum(...): minden gép hozzájárulását összeadjuk, így megkapjuk, hogy összesen hány termék készül el a közép idő alatt. Termékek száma alapján döntés:

```
        if összes_termék >= t:
            # Ha elég termék készül, csökkentjük az időtartamot
            válasz = közép
            jobb = közép - 1
        else:
            # Ha nem elég, növeljük az időtartamot
            bal = közép + 1
```

Ha az elkészített termékek száma legalább t: frissítjük a válasz-t: Ez az időpont egy lehetséges minimális idő; csökkentjük a felső határt (jobb), hogy még kisebb időt keressünk. Ha az elkészített termékek száma kevesebb, mint t: növeljük az alsó határt (bal), mivel több időre lesz szükség. Eredmény visszaadása:

```
    return válasz
```

A függvény az eddig talált legkisebb időt adja vissza, amely alatt biztosan legyártható a kívánt t termék. Bemenet kezelése:

```
# Bemenet olvasása
import sys
bemenet = sys.stdin.read
adatok = bemenet().splitlines()
```

sys.stdin.read: a program várja a bemeneti adatokat (pl. egy fájlból vagy konzolról); adatok: a beolvasott sorokat egy listába rendezi. Bemeneti adatok feldolgozása:

```
n, t = map(int, adatok[0].split())
gépek = list(map(int, adatok[1].split()))
```

n és t: az első sorból olvassuk ki a gépek számát és az elkészítendő termékek számát; gépek: a második sorban található számokat listává alakítjuk, amely megadja az egyes gépek gyártási sebességét. Függvény meghívása és eredmény kiíratása:

```
# Minimális idő kiszámítása
eredmény = minimális_idő(n, t, gépek)

# Kimenet
print(eredmény)
```

minimális_idő: meghívjuk a függvényt a bemeneti paraméterekkel; print(eredmény): kiírjuk a kiszámított minimális időt. Ez a kód hatékonyan kezeli az optimalizálási problémát bináris keresés segítségével, biztosítva, hogy a megoldás gyorsan megtalálható legyen még nagy bemeneti adatok esetén is.