CSES Stick Lengths - rekurzív feladatmegoldás

A calculate_cost függvény

```
1   def calculate_cost(p, target):
2       """Kiszámítja az összköltséget egy adott célhosszhoz."""
3       return sum(abs(x - target) for x in p)
```

Ez a függvény kiszámítja, hogy mennyi költségbe kerül az összes bot hosszát egy adott értékre (target) módosítani. Bemenete: p: A botok jelenlegi hosszúságait tartalmazó lista; target: Az a hossz, amelyre a botokat igazítani szeretnénk. Logikája: Az abs(x - target) kiszámítja az egyes botok xxx hosszának és a célhossz (target) közötti abszolút különbséget. Ezeket a különbségeket összeadjuk a sum() függvénnyel, amely megadja az összköltséget. Visszatérési érték: Az összes bot hosszának módosítási költsége az adott célhosszhoz.

A minimal_cost_recursive függvény

```
5   def minimal_cost_recursive(p):
6       # Alapeset: Ha csak egy bot van, nincs költség
7       if len(p) == 1:
8           return 0
9
10      # Rendezzük a botok hosszát
11      p.sort()
12
13      # Medián meghatározása
14      median = p[len(p) // 2]
15
16      # Költség kiszámítása a mediánnal
17      total_cost = calculate_cost(p, median)
18
19      return total_cost
```

Ez a függvény kiszámítja a botok hosszának módosítási költségét úgy, hogy a költség minimális legyen. Ehhez a medián használata az optimális megoldás.

```
7       if len(p) == 1:
8           return 0
```

Alapeset: Ha csak egy bot van a listában, akkor nincs szükség módosításra, hiszen minden bot azonos hosszúságú lesz.

Botok rendezése:
A botok hosszát növekvő sorrendbe rendezzük. Ez azért

```
11      p.sort()
```

fontos, mert a mediánt a rendezett lista alapján számítjuk ki.

```
14      median = p[len(p) // 2]
```

Medián meghatározása: A medián az a középső elem a rendezett listában. Ha a botok száma páratlan, a középső elem a medián. Ha a botok száma páros, akkor a bal oldali középső elemet vesszük mediánként (Python indexelés alapján). A medián használata azért optimális, mert az abszolút eltérések összegét minimalizálja.

```
17      total_cost = calculate_cost(p, median)
```

Összköltség kiszámítása: Meghívjuk a korábban definiált calculate_cost függvényt, hogy kiszámítsuk az összes bot hosszának módosítási költségét a mediánhoz viszonyítva.

```
19      return total_cost
```

Visszatérési érték: Visszaadja az összköltséget.

A főprogram

```
21  # Bemenet olvasása
22  n = int(input())
23  p = list(map(int, input().split()))
24
25  # Minimális összköltség kiszámítása
26  result = minimal_cost_recursive(p)
27
28  # Eredmény kiírása
29  print(result)
```

```
22  n = int(input())
23  p = list(map(int, input().split()))
```

Bemenet olvasása: Az első sorban beolvasott n az elemek (botok) száma. A második sor a botok hosszát tartalmazza.

Megoldás meghívása: Meghívjuk a minimal_cost_recursive függvényt, amely kiszámítja a minimális összköltséget a botok mediánra igazításával.

```
29  print(result)
```

Eredmény kiírása: Kiírjuk a minimális költséget. Ez a megoldás hatékonyan számítja ki a minimális költséget. A medián használata garantálja az optimális eredményt az abszolút eltérések összegének minimalizálása érdekében. A kód könnyen érthető és hatékonyan fut nagy bemeneti méret mellett is.