Függvénykönyvtár

A függvénykönyvtár különböző rendezési algoritmusokat valósít meg. Az alábbi funkciókkal kell rendelkeznie:

1. **Alapvető rendezési algoritmusok** (referenciaként):
   - Például: **Gyorsrendezés (Quick Sort)**, **Halmazrendezés (Merge Sort)**.
2. **Kevésbé ismert rendezési algoritmusok**:
   - Például: **Comb Sort**, **Binary Insertion Sort** vagy más ritkább módszerek.
3. **Felhasználó által definiált kulcs szerinti rendezés**:
   - Lehetővé teszi, hogy a felhasználó egyedi kulcsokat adjon meg a rendezéshez (pl. `lambda`-függvény).

Rendezési algoritmusok jellemzői:

- Az algoritmusok általános listákra alkalmazhatók, amelyek lehetnek számok, szövegek, vagy egyéb adattípusok.
- A rendezési függvények visszatérési értéke egy új lista, amely a bemeneti lista rendezett változata.
- Az algoritmusok kompatibilisek Python 3.x verzióval, és dokumentáltak.

Alkalmazás

Az alkalmazás egy konkrét gyakorlati példát biztosít a függvénykönyvtár használatára:

1. Egy bemeneti szövegfájlt (`input.txt`) dolgoz fel, amely soronként neveket tartalmaz.
2. Az adatokat ábécé sorrendbe rendezi.
3. A rendezett adatokat egy új szövegfájlba (`output.txt`) írja ki.

Az alkalmazás paraméterezhető, így a felhasználó megadhatja:

- A bemeneti és kimeneti fájl nevét.
- A rendezési algoritmust, amelyet használni szeretne.

A program használata

Bemenetek

1. **Függvénykönyvtár**:
    - Egy lista, amely rendezendő.
    - (Opcionális) Kulcs a rendezéshez, ha nem alapértelmezett sorrendet szeretnénk.

2. **Alkalmazás**:
    - Egy bemeneti szövegfájl, amely soronként tartalmazza a rendezendő adatokat.
    - Egy kimeneti szövegfájl neve, ahová az eredmény mentésre kerül.
    - A rendezéshez használt algoritmus neve.

Kimenetek

1. **Függvénykönyvtár**:
    - Egy rendezett lista, amely az algoritmus eredményét tartalmazza.

2. **Alkalmazás**:
    - Egy kimeneti szövegfájl, amely a rendezett adatokat tartalmazza soronként.

Fájlkezelés

- A bemeneti szövegfájlnak `.txt` kiterjesztéssel kell rendelkeznie, és soronként tárolja az adatokat.
- A kimeneti szövegfájlt a program automatikusan létrehozza, vagy felülírja, ha már létezik.

Funkcionalitás leírása

(user ezeket fogja használni, így csak azok vannak feltüntetve)

Rendezési függvények

1. **`comb_sort(data: list) -> list`**
    - Beágyazott fésűs rendezést valósít meg, amely csökkenti a nagyobb elemek "buborék-effektusát".
2. **`binary_insertion_sort(data: list) -> list`**
    - Bináris kereséssel optimalizált beszúrásos rendezés.
3. **`quick_sort(data: list) -> list`**
    - A gyorsrendezés klasszikus megvalósítása.
4. **`merge_sort(data: list) -> list`**
    - Halmazrendezés (oszd meg és uralkodj technika).

Szövegfájl rendezés

1. Az `input.txt` fájl beolvasása.
2. A fájl adatait soronként egy listába olvassa be.
3. A kiválasztott algoritmussal rendezi a listát.
4. A rendezett lista tartalmát a `output.txt` fájlba írja ki.

A program kezelése

1. **Könyvtár használata**:
   - Importálás:

     ```
     from sort_library import comb_sort, sort_file
     ```

   - Függvényhívás:

     ```
     data = [34, 7, 23, 32, 5, 62]
     sorted_data = comb_sort(data)
     print(sorted_data)
     ```

2. **Alkalmazás futtatása**:
   - A `main.py` futtatása parancssorból:

     ```
     python main.py --input input.txt --output output.txt --
     algorithm comb_sort
     ```

Specifikáció megjegyzések

- A program megvalósítása során nem számítunk arra, hogy a bemeneti fájlok mérete meghaladja a normál memóriaigényt.

- Hibakezelés biztosított a hiányzó fájlok, üres bemenetek, és nem megfelelő adattípusok esetére.

- Az alkalmazás könnyen bővíthető további rendezési algoritmusokkal és testreszabott funkciókkal.