Dokumentace k programu *Bubble sort* – Filip Raasch

Zadání

Na vstupu posloupnost čísel uložená v textovém souboru. Výsledkem setříděná posloupnost uložená do textového souboru.

Úvod

Metoda *Bubble sort* patří mezi třídící algoritmy. Třídícími algoritmy jsou myšleny postupy, které jsou schopny setřídit sadu datových záznamů do požadovaného pořadí. Metoda *Bubble sort* je algoritmem, který je především používán na setřídění posloupnosti čísel dle jejich velikosti, a to buď vzestupně, či sestupně.

Bubble sort se řadí k nejjednodušším třídícím algoritmům. Jednoduchost tohoto algoritmu je však vykoupena neefektivitou pro velké datové řady. Používá se proto především pro nenáročné aplikace.

Princip

Při metodě *Bubble sort* procházíme posloupnost vždy po dvojicích čísel. U každé dvojice čísel se porovná jejich velikost a v případě nevyhovujícího pořadí se dvojice prohodí. Seznam procházíme do té doby, než v něm není dvojice čísel, u které je jejich pořadí nevyhovující.

Příklad:

Máme následující nesetříděný seznam čísel a chceme ho setřídit od nejvyššího po nejnižší:

3 10 2 9 1

Krok 1:
V prvním kroku se prohodí čísla 3 a 10 a také čísla 2 a 9, jelikož jsou ve špatném pořadí.
10 3 9 2 1

Krok 2:
V následujícím kroku se už pouze prohodí čísla 3 a 9.
10 9 3 2 1

Pro tuto posloupnost čísel tedy stačilo seznam projít dvakrát, abychom dostali správně setříděnou posloupnost. Dle počátečního seřazení čísel však může narůst složitost až na n^2 (kdy n je počet prvků) prohození čísel. Tento exponenciální nárůst složitosti je důvod, proč se v praxi tato metoda často nepoužívá.

Existující algoritmy:

Algoritmů pro metodu *Bubble sort* je pro většinu programovacích jazyků napsáno již větší množství.

Dle stránky GeeksforGeeks.org je nejjednodušší varianta algoritmu napsána následovně, pro seřazení od nejmenšího čísla po nevětší:

```
def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

Nevýhoda této implementace algoritmu je, že ho může procházet nadále, i když už je posloupnost seřazená. V některých případech je metoda *Bubble sort* proto optimalizována tak, aby procházela seznam pouze, dokud v ní jsou neseřazená čísla:

```
def bubble_sort(our_list):
    # We want to stop passing through the list
    # as soon as we pass through without swapping any elements
    has_swapped = True

    while(has_swapped):
        has_swapped = False
        for i in range(len(our_list) - 1):
            if our_list[i] > our_list[i+1]:
                # Swap
                our_list[i],our_list[i+1]=our_list[i+1], our_list[i]
                has_swapped = True
```

Metoda *Bubble sort* se dále vyznačuje tím, že už při první iteraci skončí na posledním místě seznamu buďto největší nebo nejmenší položka (dle toho, jak seřazujeme). Můžeme proto při každém dalším procházení seznamu vynechat o jedno porovnání čísel více. Toho využívá další optimalizace:

```
def bubble_sort(our_list):
    has_swapped = True

    num_of_iterations = 0

    while(has_swapped):
        has_swapped = False
        for i in range(len(our_list) - num_of_iterations - 1):
            if our_list[i] > our_list[i+1]:
                # Swap
                our_list[i],our_list[i+1] = our_list[i+1], our_list[i]
                has_swapped = True
        num_of_iterations += 1
```

Popis zvoleného algoritmu

Mnou zvolený algoritmus pro seřazení čísel sestupně je v rámci optimalizace na druhé úrovni – nepokračuje tedy, když už jsou čísla seřazená. Proto je základem metoda while - cyklus, který skončí, když jsou čísla seřazena. Pro procházení seznamu je využit for – cyklus:

```
# setrideni cisel metodou bubble sort
p=0
while p == 0:
    p = 1
    for i in range(len(cisla)-1):
        if cisla[i]<cisla[i+1]:
            a=cisla[i]
            cisla[i]=cisla[i+1]
            cisla[i+1]=a
            p=0
```

Struktura programu

Program se v kostce dělí na dvě části – na definování použitých funkcí a hlavní část programu.

Definované funkce:

Název: `zda_cislo`

Použití: Funkce zjistí, zda načtená hodnota je číslo tím, že se ho pokusí převést na typ float. Je použita pro odfiltrování textových dat ze vstupního souboru.

Název: `nacist_cisla`

Použití: Funkce načte data z textového souboru vstup.txt. Zároveň odstraní případné textové řetězce. Funkce vrací seznam neseřazených čísel.

Název: `uloz_cisla`

Použití: Funkce uloží čísla do textového souboru vystup.txt. Jsou uložena na jednom řádku, oddělené mezerami.

Hlavní část programu

V hlavní části programu se pomocí funkce nacist_data načtou všechna čísla (typu integer nebo float) z textového souboru vstup.txt. Následně je seřadí metodou *Bubble sort* od nejvyššího po nejnižší. Algoritmus pro tento krok je zmíněn výše. V poslední části uloží setříděnou posloupnost čísel funkcí uloz_cisla.

Vstupní a výstupní data

Vstup:

Ve složce, ze které je spouštěn .py program, je nutné mít soubor vstup.txt, ve kterém jsou uložena data. Čísla by měla být oddělena mezerou nebo čárkou. Jiný oddělovat čísel způsobí, že program vyhodnotí číslo jako text. V souboru mohou být také uložena další data nebo text, měl by však být oddělen od čísel mezerou nebo čárkou. V případě, že jsou používána čísla desetinná, musí být desetinným oddělovačem tečka. Použití desetinné čárky způsobí, že je číslo rozděleno na dvě celá čísla.

Výstup:

Do složky, ze které je spouštěn .py program, je uložen soubor vystup.txt. V tomto souboru je seřazena posloupnost čísel sestupně na jednom řádku. Pokud soubor ve složce neexistuje, tak je vytvořen. V případě, že již existuje, tak je přepsán.

Problematická místa a možná vylepšení

Možným problémem programu může být pomalost metody *Bubble sort.* Pokud je na vstupu velké množství dat, je možné, že program bude pomalý, vzhledem k exponenciálnímu se zvyšování časové náročnosti při vyšším objemu dat. Problémy mohou nastat také, když na vstupu je nečitelný soubor, nebo v něm nejsou uložena čísla správně. Program funguje pouze pro reálná čísla.

Efektivita programu by šla vylepšit mnoha způsoby. Algoritmus samotného třídění by šel vylepšit snižováním o jedno porovnání, jak je zmíněno v pasáži o existujících algoritmech. Dále by mohl být program vylepšen uživatelsky. Uživatel by mohl být schopen vybrat, odkud číst data a kam je ukládat. Volba, zda mají být čísla seřazena sestupně či vzestupně, by také mohla být přínosná.

Shrnutí

Přestože je metoda *Bubble sort* časově náročná na nevhodná pro velké objemy dat, má své výhody. Především je snadno implementovatelná a pro rychlé seřazení malého množství dat může být dobře využitelná.