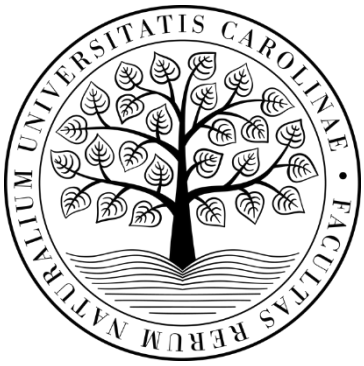
Univerzita Karlova
Přírodovědecká fakulta

Studijní program: Geografie a kartografie



Šimon Kohout

3. ročník studijního programu Geografie a kartografie

Výpočet mediánu pro nesetříděnou posloupnost tvořenou *n* prvky

Úkol ke zkoušce

Předmět „Úvod do programování“

Akademický rok 2024/2025

Praha, 12. 2. 2025

Zadání:

Naprogramovat program, který je schopný vypočítat medián posloupnosti *n* čísel, které nemusí být seřazeny. Uživatel zadá neseřazenou číselnou posloupnost, ze které bude vypočítán medián. Nesmí být použito vestavěné funkce pro výpočet mediánu (např. *statistics.median*) ani vestavěné třídění (*sort*).

Rozbor:

Výpočet mediánu je běžnou úlohou v oblasti statistiky a datové analýzy. Medián představuje střední hodnotu číselné posloupnosti a slouží jako míra centrální tendence, která je méně citlivá na extrémní hodnoty než aritmetický průměr. Hlavním cílem tohoto problému je nalézt medián v libovolné neseřazené posloupnosti obsahující *n* prvků. Předpoklad pro řešení této problematiky je, že *n* je kladné číslo (tedy n > 0)

Medián je definován tak, že pokud je počet prvků lichý, pak je medián rovný prostřední hodnotě setříděné posloupnosti. Pokud je počet prvků sudý, medián je aritmetický průměr dvou prostředních hodnot. To znamená, že před samotným určením mediánu je nutné posloupnost nejprve setřídit. Prvky posloupnosti mohou být celá čísla (kladná i záporná) a také desetinná čísla. Účelem algoritmu by mělo být tedy i správné pracování s desetinnými čísly. Výstupem je jedna číselná hodnota odpovídající mediánu této posloupnosti.

Vhodné algoritmy:

Pro použití v tomto zadání byl zvolen jeden z třídících algoritmů a to tzv. **Bubble Select**. Fungování programu záleželo zejména na výběru vhodného třídícího algoritmu. Vybraný algoritmus má velkou výhodu v jednoduché implementaci. Naopak jeho nevýhoda je nízká efektivita pro velká data. Jeho časová složitost je **$O(n^2)$** (v nejlepším případě O(n)). Fungování tohoto algoritmu spočívá v posouvání největšího prvku seznamu na konec v každé iteraci. Algoritmus musí projít a porovnat velké množství čísel.

Opačně funguje algoritmus **Selection Sort**. Ten funguje na stejném principu, jen nejnižší číselné hodnoty přesune na začátek. Časová složitost je tedy stejná jako pro **Bubble Select**. Dalším třídícím algoritmem je například **Insertion Sort**, který je vhodný pro částečně seřazené posloupnosti. Tento algoritmus porovnává číselné hodnoty a vkládá je na správná místa v seznamu. Výše zmíněné algoritmy jsou jednoduché na implementaci a vhodné pro menší seznamy.

Pro velké seznamy se hodí například algoritmus **Quickselect**, který netřídí množinu, ale hledá k-tý nejmenší prvek. Náhodně vybere tzv. *pivot* a porovnává ho s ostatními čísly v seznamu. Ten je tedy rozdělen na prvky větší a menší než vybraná hodnota. Algoritmus rekurzivně pokračuje dál, dokud nenajde medián.

Použitý algoritmus:

1. Inicializace a přečtení vstupních dat zvolené uživatelem
2. Seřazení neseřazené posloupnosti pomocí algoritmu
3. Výpočet mediánu (odlišný přístup pro liché a sudé číslo)
4. Kontrola, zda jsou vstupní data validní
5. Výpis vypočítaného mediánu

Popis programu

1. **Třídy**
   Třída *Median* slouží k výpočtu mediánu z nesetříděného seznamu čísel a obsahuje metody pro vlastní výpočet mediánu a pro Bubble Sort.
2. **Atributy**
   - *self.numbers* - instanční atribut, který ukládá seznam čísel zadaných uživatelem
3. **Metody**
   - *__init__(self, numbers)* – inicializátor objektu a ukládání seznamu
   - *bubble_sort(self)* – seřazení seznamu pomocí Bubble Sort
   - *calculate_median(self)* – výpočet mediánu

Popis vstupních a výstupních dat

Vstupní data jsou neseřazený číselný seznam obsahující i záporná, a i desetinná čísla. Pokud by se v seznamu objevilo něco jiného, program vypíše chybovou hlášku a uživatel bude vyzván vypsat seznam znova. Výstupní data jsou jednoduchý text s vypsaným vypočteným mediánem.

Příklad vstupního textu:

1.25 22 16 -12.5 69

Příklad výstupního textu:

Your median is: 16.0

Možná vylepšení

Program funguje spolehlivě a rychle pro malé posloupnosti. Ke zrychlení může dojít při vybrání vhodnějších třídících algoritmů (např. Quickselect). Dalším vhodným vylepšením může být možnost načtení souboru z počítače. Výsledkem by pak mohl být nový soubor. Dále by bylo vhodné pro uživatele umožnit opakované výpočty, aby uživatel nemusel posloupnost vypisovat znovu po dokončení výpočtu.

Závěr

Výsledný program spolehlivě vypočítá medián z neseřazené posloupnosti o n prvcích. Rychlost výpočtu závisí na zvoleném třídícím algoritmu a velikosti posloupnosti. V programu je stále prosto k zdokonalování (např. rychlosti a uživatelské přívětivosti)