Univerzita Karlova

Přírodovědecká fakulta

Studijní obor: Geografie a kartografie

Veronika Kalová

Úloha č. 94 – Výpočet mediánu pro nesetříděnou posloupnost tvořenou n prvky

Úkoly ke zkoušce – Úvod do programování

Kralupy nad Vltavou, 2026

Rozbor problému

„Na vstupu je neuspořádaná posloupnost n celočíselných prvků, načtená z textového souboru. Spočtěte medián této posloupnosti, pro setřídění množiny nepoužívejte vestavěný třídící algoritmus.“

Mějme vstupní dokument (textový soubor), ve kterém je posloupnost čísel s čísly v náhodném pořadí. Úkolem úlohy č. 94 bylo vytvořit program, který by danou posloupnost čísel (konečná řada čísel) seřadil, našel medián a výsledky vypsal. Pro seřazení množiny čísel nesměl být vestavěný algoritmus pro řazení. Zadání nespecifikuje způsob tisku výsledků nebo formát výstupních dat.

Existující algoritmy

Pro tuto úlohu by mohl být použit vestavěný algoritmus pro setřídění dané posloupnosti (např. *sort* nebo *sorted*), jehož použití však bylo v této úloze zakázáno. V úloze by mohlo být využito jiných algoritmů, například Quick Sort nebo Merge Sort, jejich implementace by byla však složitější než zvolený algoritmus Bubble Sort.

Dále by bylo možné použít funkci median importovanou z vestavěné knihovny statistics či vestavěné funkce QuickSelect, který dokáže najít n-tý nejmenší člen neseřazené množiny čísel. Tyto funkce nebyly v úloze explicitně zakázány, ale předpokládám, že by jejím použitím došlo k velkému ulehčení úlohy a byl by ztracen význam zadání.

Popis zvoleného algoritmu

Jako nejjednodušší algoritmus pro seřazení vstupní posloupnosti celočíselných prvků byl vybrán Bubble Sort. Bubble Sort pracuje na principu porovnání dvou sousedních hodnot a přepsání jejich pořadí od nejmenší k největší. Pro seřazení čísel v množině existují i jiné algoritmy, rychlejší algoritmy – viz kapitola Existující algoritmy. V tomto úkolu byl však Bubble Sort použit pro svoji jednoduchost a předpokladu nízkého počtu celočíselných prvků ve vstupním dokumentu. Princip je pro lepší pochopení uživatelem zapsán v „pseudokódu“:

Pro každé číslo i v rozpětí (délka):

Pro každý číslo j v rozpětí (0 až délka – znak – 1):

Pokud je číslo na pozici n větší, než číslo na pozici (n + 1):

Prohoď pořadí čísel

Dalším algoritmem využitým v tomto programu je výpočet mediánu. To je prováděno pomocí zjištění prostředního prvku (u lichých čísel) nebo výpočtem průměru dvou středních čísel (u sudých čísel) a následné vypsání daného členu.

Pokud je počet číslic v souboru dělitelný dvěma beze zbytku:

Vyber číslo na pozici (délka posloupnosti dělená celočíselně dvěma a posun se o pozici zpátky)

Vyber číslo na pozici (délka posloupnosti dělená celočíselně dvěma)

Z těchto dvou čísel udělej průměr

Jinak:

Vyber číslo na pozici (délka posloupnosti dělená celočíselně dvěma)

Vypiš číslo na dané pozici

V tomto algoritmu je důležité pro výpočet mediánu ze sudého počtu prvků vzít prvky na místech [i] a [i-1], protože v programátorských jazycích jsou prvky indexované od nuly (na prvním místě je číslo s indexem [0], na druhém místě číslo s indexem [1], atd.).

Struktura programu

Samotný program je napsán v objektově orientovaném prostředí. Program má vytvořenou třídu *Median*, která se skládá z několika metod, jimiž jsou *otevri_soubor*, *serad*, *najdi_median* a *tisk_vysledku*.

V první metodě (*otevri_soubor*) probíhá otevření souboru, jeho čtení a přeformátování. Pomocí funkce *.strip()* jsou z konců řádků odstraněny případné mezery. Dále jsou jednotlivá čísla oddělena středníkem. V tomto bloků kódu dochází i k ošetření chyb (*FileNotFoundError*, *ValueError*, a *Exception*), přičemž při výskytu FileNotFoundError (nenalezení souboru) by mělo pomocí příkazu *exit()* dojít k ukončení programu. Data jsou následně dočasně uložena do listu s názvem „self.serazeni“ ve formátu integer pro další práci – seřazení a přepsání do výstupního textového souboru.

Dále program volá metodu *serad*, která nejdříve přečte délku posloupnosti (resp. počet čísel v ní) a následně pomocí algoritmu bubble sort seřadí celou posloupnost čísel od nejmenšího po největší. Metoda na konci zavolá metodu *najdi_median*, kde probíhá samotný výpočet mediánu celočíselné posloupnosti. Nakonec je programem zavolána metoda *tisk_vysledku*, která v konzoli informuje uživatele o tvorbě textového souboru s názvem „vysledky94.txt“. Zadání nespecifikovalo formát výstupu (zdali má vzniknout nový dokument se statistikou či zdali se mají výsledky vytisknout do konzole uživatele), já se pro vyšší přehlednost rozhodla pro použití této metody. Tato metoda zároveň uživateli umožňuje další práci s daty.

Popis vstupních a výstupních dat

Vstupními daty pro tento úkol je textový soubor pojmenovaný „uloha_94.txt“. Tento soubor obsahuje řadu náhodných celých čísel oddělených od sebe středníkem („;“) v tomto formátu (číslice obsažené v tomto příkladu nejsou součástí vstupních dat):

```
31;33;25;11;19;23;5;37;1;17;3;29;13;9;7;15;27;39;21;35
```

Zadání sice specifikuje, že se jedná o celá čísla, ale někoho, kdo k programu přijde bez ponětí, co program dělá, by mohl soubor s číselnou posloupností být matoucí – proto nejsou čísla oddělena čárkou, jak je tomu obvykle. Pro ulehčení čtení znaků mezi číslicemi nejsou mezery.

Výstupními daty tohoto úkolu je soubor „*výsledky94.txt*“ vytvořený přímo v programu (přesněji v metodě *tisk_vysledku*), do kterého byla přepsána seřazená posloupnost. Čísla jsou pro větší přehlednost rozdělena do řádků, takže každý řádek obsahuj právě jedno číslo posloupnosti. Pod vypsanými čísly je statistika – zjištěný medián pro danou posloupnost n prvků ve formátu: „Medián nesetříděné posloupnosti je X."

Problematická místa

Mezi jedno z problematických míst tohoto programu patřilo vymyšlení algoritmu pro čtení a přepis. Zvolila jsem postup, při kterém jsou následně seřazená čísla přepsána do jednotlivých řádků výstupního programu. Sice se tím výstupní soubor stává delším, než byl původně, ale více přehledným. Úpravy pro výstup jsou dále popsány v kapitole Možná vylepšení.

Možná vylepšení

Mezi vylepšení programu by určitě patřilo rozšíření pro desetinná čísla. V tomto případě by bylo potřeba změnit výstup čísel z typu integer na float.

Dalším možným vylepšením programu by byl lepší přepis – seřazená čísla by mohla ve výstupním dokumentu být vypsána v řádku. Tím by došlo k více přirozenému výstupu, na druhou stranu se výrazně snižuje přehlednost (kdyby posloupnost obsahovala více čísel, než by se po seřazení dalo vytisknout na jeden řádek, uživatel by pravděpodobně měl větší problém najít prostřední číslo, které by odpovídalo mediánu).

V neposlední řadě by program uvítal vyšší flexibilitu v podobě vstupních dat – například přidáním možnosti výběru pro uživatele, jakým znakem jsou čísla v neseřazené posloupnosti dělena (toto by pomohlo hlavně v kombinaci s vylepšením programu na desetinná místa) či vytvoření příkazu pro identifikaci jednotlivých čísel, ať už jsou rozdělena jakýmkoli znakem (např. čárka, tečka, středník, mezera, pomlčka, atd.)

V případě, kdy by bylo předpokládáno, že bude vstupní soubor obsahovat vysoký počet celočíselných prvků, bylo by efektivnější změnit třídící algoritmus na jiný, protože Bubble Sort se vyšším počtem vstupních čísel v množině stává časově neefektivním. Jeho časová složitost se dá vypočítat pomocí vzorce $O(n^2)$, zatímco časová složitost jiných třídících algoritmů (např. QuickSort) je počítán pomocí vzorce $O(n \log n)$.