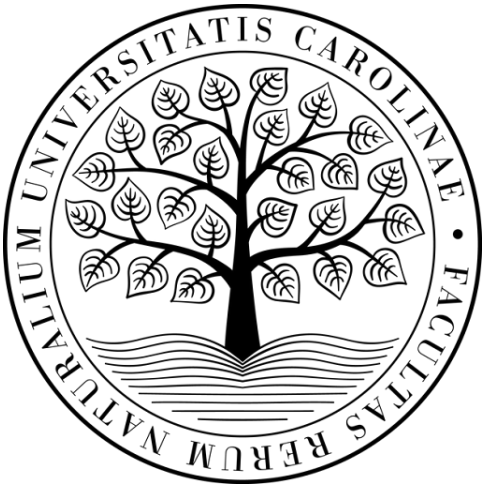
Univerzita Karlova

Prírodovedecká fakulta



Daniel Kalakay

2. ročník študijného programu Sociálna geografia a geoinformatika

Nalezení k-tého největšího prvku v posloupnosti.

Dokumentácia k skúškovému programu na predmet:

Úvod do programování

Akademický rok 2025/2026

Praha 20. 1. 2026

Obsah

Úvod a stručné zadanie

Táto práca sa zaoberá návrhom, vypracovaním a následným zdokumentovaním programu pre nájdenie k-tého najmenšieho prvku v textovom súbore vyplneného používateľom. Celý program bol zostrojený v jazyku Python a v prostredí PyCharm, využíva princípy objektovo orientovaného programovania (OOP).

Cieľom práce je vytvoriť program, ktorý je prehľadný, rozšíriteľný a užívateľsky jednoduchý na používanie. Program by mal podľa zadania okrem hodnoty prvku vypísať aj jeho pôvodnú pozíciu vo vstupnom súbore.

Presné zadanie úlohy

Vstupom programu by mal byť textový súbor obsahujúci neusporiadanú postupnosť reálnych čísel. Užívateľom zadaná hodnota K, určujúca poradie hľadaného prvku v postupnosti, zadávaná ja pomocou vstupu z klávesnice užívateľa. Platí podmienka že $K \leq n$, kde n je počet čísel vo vstupnej postupnosti.

Úlohou programu je nájsť k-tý najmenší prvok v postupnosti a určiť jeho pôvodnú pozíciu v neusporiadanom vstupe. Výsledkom programu by mal byť hľadaný prvok a jeho pozícia v pôvodnom programe.

Analýza úlohy

Úloha by sa dala zjednodušene poňať ako rozdelenie číselného radu a nájdenie pomocou indexu hľadaný prvok. Program by mohol mať nasledovnú štruktúru:

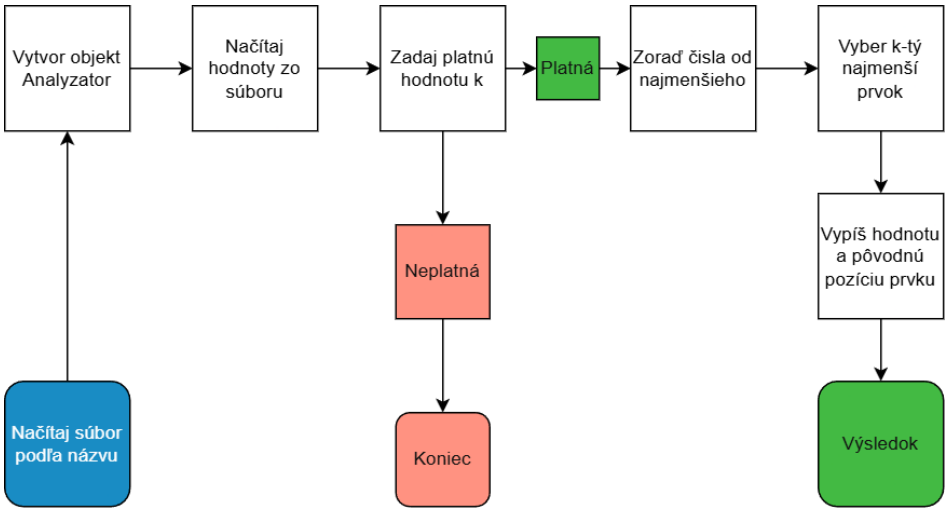
- Načítanie vstupného súboru s neusporiadaným radom reálnych čísel
- Vstup od používateľa
- Uloženie pôvodného radu pomocou enumerácie
- Zoradenie radu a pomocou indexu prvku nájsť prvok súhlasný so vstupom používateľa
- Vyberanie hľadaného prvku
- Vypísať výsledky

Z obecného zadania pre riešenie každého príkladu, používať OOP, je určite najvýhodnejšie zabaliť celý proces do jednej triedy, ktorá následne ponesie zodpovednosť za spravovanie všetkých dát, či už to je vyhľadanie prvku, uchovávanie týchto dát, vytvorenie dočasného listu a následné vypísanie výsledkov. Možno jedinou výhodou tohto prístupu je menšie zaťaženie celkového procesu, jednoduchšia manipulácia s dátami pri prípadnom rozširovaní programu a globalizácia použitých premenných.

Voľba algoritmu

Použité postupy a algoritmy

Náš algoritmus prechádza vybraný súborom s číslami zjednodušene asi nejako takto:



Táto schéma zjednodušene ukazuje ako zhruba náš program pracuje. Načítava všetky hodnoty zo vstupu a následnom zoradení. Pre zachovanie informácii o pôvodných pozíciách je pri spracovaní použitá enumerácia, ktorá ku každej hodnote priradzuje jej index vo vstupnej postupnosti.

Najjednoduchšie je si celú problematiku ukázať na jednoduchom príklade, napríklad máme vstupný súbor : ***test.txt*** s neusporiadanými hodnotami ***1, 3, 2, 5***

1. Používateľ zadá hľadané k, v našom prípade bude *k = 2*
2. Program prevedie hodnoty do zoznamu dvojíc v tvare (*index, hodnota*)

Hodnota	1	3	2	5
Index	0	1	2	3

3. Následne začne postup zoradenia podľa čísel od najmenšieho

Hodnota	(0,1)	(2,2)	(1,3)	(3,5)
Index	0	1	2	3

4. Podľa užívateľa výberného *k* nájde hodnotu na indexe s hodnotou 1 keďže index počítame od číslice 0 a nie 1

Hodnota	(0,1)	(2,2)	(1,3)	(3,5)
Index	0	1	2	3

5. Vypíše výsledok z riadku hodnota, program pozná že pôvodná hodnota 2 mala index 2

Diskusia k výberu algoritmu

Pri ťažkom rozhodovaní, ktorý algoritmus použiť autor uvažoval aj o efektívnejších, no bohužiaľ menej transparentnejších možnostiach, napríklad výberový algoritmus typu QuickSelect. Tento algoritmus by nevyžadoval úplné zoradenie vstupných dát. Vzhľadom na zameranie úlohy a obmedzenému rozsahu dát bol pre istotu zvolený algoritmus ktorý dáta triedi podľa ich hodnoty.

Vybraný algoritmus by mal byť prehľadnejší a ľahšie implementovateľný a umožňuje prehľadnú demonštráciu práce so zoznamom a uchovávaním pôvodných indexov.

Program

Výsledný kód používa triedu Analyzator, ktorá zastrešuje všetku funkcionalitu potrebnú pre prácu s dátami a vyhľadávaním požadovaného prvku.

Trieda obsahuje atribút pre názov vstupného súboru a zoznam, v ktorom sú načítané čísla zo súboru. Dáta sa načítavajú automaticky pri vytvorení triedy používateľom. Hlavná metóda triedy slúži k nájdeniu k-teho najmenšieho prvku a vracia jak jeho hodnotu tak aj pozíciu v pôvodnej postupnosti.

Hlavná časť programu vytvára inštanciu analyzátoru, načítava hodnotu k od užívateľa, volá príslušné metódy a zobrazuje výsledky.

```python
class Analyzator:
    def __init__(self, subor):
        self.subor = subor
        self.sekvencia = []
        self._nacitaj_data()

    def _nacitaj_data(self):
        with open(self.subor, "r") as f:
            self.sekvencia = [float(x) for x in f.read().split()]

    def nacitaj_najmensi_k(self, k):
        if k < 1 or k > len(self.sekvencia):
            raise ValueError("Neplatna hodnota k")

        ind_sekvencia = list(enumerate(self.sekvencia)) # uloženie pôvodných pozícií

        ind_sekvencia.sort(key=lambda x: x[1])  # triedenie podľa hodnoty

        pozicia, hodnota = ind_sekvencia[k-1]
        return hodnota, pozicia

def main():
    subor = "vstupny_subor_ukol_2.txt"
    funkcia = Analyzator(subor)

    k = int(input("Zadajte k: "))

    hodnota, pozicia = funkcia.nacitaj_najmensi_k(k)

    print(f"{k}-ty najmensi prvok ma hodnotu {hodnota}")
    print(f"Povodna pozicia v postupnosti je {pozicia+1}")

if __name__ == "__main__":
    main()
```

Alternatívne prístupy

Možnou, autorom uprednostňovanou, metódou by bolo klasické procedurálne riešenie bez použitia tried, program by používal iba funkcie a globálne premenné. Také riešenie by bolo určite menej prehľadné a ťažko rozšíriteľné ale za to rýchlejšie na naprogramovanie a možno časovo úspornejšie. Ďalšou metódou by mohlo byť vytvorenie separátnych knižníc alebo algoritmov pre výber prvku bez úplného triedenia. V práci by mohli pôsobiť ale menej prehľadne.

Vstupné dáta

Vstupné dáta sú uložené v externom textovom súbore vo forme reálnych čísel oddelených medzerami alebo novým riadkom. Na finálnom rozdelení nezáleží, záleží iba na správnom vyjadrení čísel. Čísla by mali byť oddelené medzerou a namiesto desatinnej čiarky užívateľ použije bodku. (0.01 5.03 10.54)

Výstupné dáta

Výstup programu je vypísaný v konzole. Obsahuje informáciu o hodnote hľadaného k-teho prvku a jeho pozícii v postupnosti. Pozícia je používateľovi prezentovaná s indexovaním od čísla jedna, aby mala viac ľudskú formu.

Priebeh práce

Práca na programe zase prekvapivo začala na papieri, autor nakreslil schému a podľa nej sa riadil po celú čas práce. Autor začal vytvorením triedy a následne tvorbou funkcii, najprv na načítanie súboru a potom na nájdenie hľadaného prvku.

Po dokončení programu bol program testovaný na rôznych sadách vstupných súborov s dátami, iným výberom hodnoty k a boli dokončené úpravy formátu výstupu.

Čo by bolo vhodné pridať

Pri vytváraní autor zhodnotil že by sa do programu určite hodilo pridať viac zabezpečení formátu vstupu, prípadne širšia podpora oddeľovačov vo vstupnom súbore.

Následne prišla možnosť že v súbore sa nachádzajú dva alebo viac zhodných čísel, program stále vyberá číslo s nižším indexom takže ak sa v súbore nachádza viac zhodných čísel program ich nevypíše všetky. Možné rozšírenie a ošetrenie tohto problému by bolo postupné preverenie každého čísla s vybraným najnižším, ak by sa zhodovali zapísal by dané čísla a ich indexy do separátneho listu, ktorý by následne vypísal.

Záverečný povzdych

Pri riešení úlohy autor zistil že znovu písanie dokumentácie nie je až taká zlá vec ale stále sa prikláňa k názoru že by úlohu osobne cez OOP neriešil. Zadanie nemalo príliš vysokú náročnosť ale implementácia OOP znova autora trošku potrápila. Autor si ale stále stojí za svojím presvedčením že dokumentáciu k programu nechce písať.