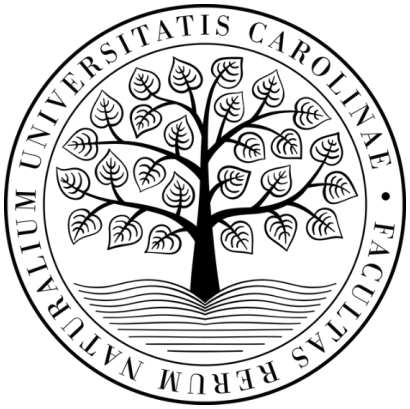
Univerzita Karlova

Přírodovědecká fakulta



ÚVOD DO PROGRAMOVÁNÍ

Dány dvě posloupnosti čísel, nalezení jejich průniku

Petra Krsková

3. ročník Geografie a kartografie

Solnice 2022

Zadání

Na vstupu jsou dvě nesetříděné posloupnosti celých čísel. Nalezněte prvky, které se nacházejí v obou posloupnostech. Pokuste se tuto operaci učinit dostatečně rychle, aby byla metoda použitelná i pro dlouhé posloupnosti. Vstupní data načtěte z textového souboru, výstup uložte také do textového souboru.

Součástí odevzdané úlohy bude zdrojový kód aplikace, vstupní/výstupní data a dokumentace se zadáním v rozsahu 4–5 stran ve formátu PDF obsahující následující:

- rozbor problému
- existující algoritmy
- popis zvoleného algoritmu
- struktura programu (datové struktury, metody,…)
- popis vstupních/výstupních dat
- problematická místa
- možná vylepšení

Aplikace bude považována za nefunkční, pokud:

- při zpracování dat dojde k pádu (runtime chyby, …)
- vrací špatné výsledky
- neřeší možné singulární případy

Rozbor problému

Posloupnost v matematice představuje sadu objektů, v níž se mohou jednotlivé objekty opakovat a záleží na jejich pořadí. Počet objektů, který může být konečný i nekonečný, je vyjádřen délkou posloupnosti. Posloupnost lze definovat jako funkční vztah mezi množinou přirozených čísel (pozice člena posloupnosti) a jejich obrazem (hodnota členu posloupnosti na dané pozici). Posloupnost bývá označována písmeny ve formě a_n, kde dolní index *n* označuje n-tý člen posloupnosti (Wikipedia 2022b).

S posloupnostmi lze provádět základní množinové operace, mezi které patří i průnik. Jako průnik dvou množin *A* a *B*, označujeme množinu obsahující všechny prvky z *A*, které zároveň náleží i množině *B*. Průnik těchto dvou množin pak zapisujeme jako $A \cap B$ (Wikipedia 2022b).

Existující algoritmy

Nejjednodušší metodou pro vytvoření průniku dvou posloupností je využití cyklu, který prochází prvky jednoho seznamu představující první posloupnost a pomocí podmínky kontroluje, zda se nachází i ve druhém seznamu, který odpovídá druhé posloupnosti. Pokud ano, tento prvek je zapsán do třetího seznamu, který po skončení cyklu obsahuje hodnoty odpovídající průniku obou posloupností. Další možností je použití operace *set()* na každý ze seznamů, čímž dojde k odstranění duplicitních prvků. Následně je na oba seznamy aplikován bitový součin a výsledek převeden do podoby seznamu. Jinou variantou je aplikace funkce *set()* na delší množinu a následné použití vestavěné funkce *intersection()* (GeeksforGeeks 2021).

Použitý algoritmus je však založen na prvotním setřídění posloupnosti, čehož lze dosáhnout několika způsoby. Algoritmus *Bubble Sort* je založen na porovnávání sousedních prvků, u kterých dochází k prohození, pokud nejsou ve správném pořadí. Metoda *Merge Sort* nejprve rozdělí posloupnost na dvě poloviny, které pak vzájemně porovnává a ve správném pořadí opět slučuje dohromady. *Insertion Sort* vyhledává správnou pozici prvku ve tříděné posloupnosti. Na začátku porovná první dva prvky a seřadí je. Následně vezme třetí prvek posloupnosti, porovná s dvěma předchozími a zařadí na správnou pozici. Toto je zopakováno pro všechny zbylé prvky posloupnosti. Jednou z možností je také *Shell Sort* založený na rozdělení posloupnosti na menší části a jejich setřídění. Další variantou je *Selection Sort* použitý v tomto případě, který je podrobněji popsán v následující kapitole Tutorialspoint 2022).

Popis zvoleného algoritmu

Pro získání průniku dochází nejprve k vzestupnému setřídění dvou vstupních posloupností. Pro to byl zvolen algoritmus *Selection Sort*, který je založen na procházení seznamu a nalezení nejmenší hodnoty. Tato hodnota je zařazena na začátek setříděného seznamu a tento proces je opakován pro všechny zbývající prvky v dosud nesetříděném seznamu. Každý nově přiřazený prvek do již setříděného seznamu je nejprve porovnán s ostatními prvky a poté zařazen na odpovídající pozici.

Pro získání průniku takto setříděných posloupností jsou postupně načítány a porovnávány prvky z obou posloupností. Pokud je načtený prvek z první posloupnosti menší, načte se následující prvek z první posloupnosti a opět dochází k porovnání. Postup je analogický v případě, že je menší prvek ze druhé množiny. Pokud se oba prvky rovnají, dochází k zápisu do výstupního seznamu a načtení dalších prvků (GeeksforGeeks 2022).

Pseudokód zvoleného algoritmu

Funkce SortList (input_list)

cyklus *procházení pozic i seznamu hodnot*

min = počáteční pozice

cyklus *procházení pozic hodnot od i+1*

pokud *je hodnota na pozici menší než na min*

zápis *hodnoty do min*

prohození *minimální hodnoty s porovnávanou hodnotou*

navrácení *input_list*

Funkce IntersectionOfLists (l1, l2)

m = délka seznamu l1

n = délka seznamu l2

i, j = inicializace na hodnotu 0

intersection = nový seznam

last = inicializace na None

dokud *je i menší než m a zároveň j menší než n*

pokud *hodnota seznamu l1 na pozici i je menší než hodnota l2 na pozici j*

zvýšení *i o 1*

jinak pokud *hodnota seznamu l2 na pozici j je menší než hodnota l1 na pozici i*

zvýšení *j o 1*

jinak pokud *hodnota seznamu l1 na pozici i se rovná hodnotě l2 na pozici j*

pokud *hodnota l1 na pozici i se nerovná last*
připojení *hodnoty do intersection*
zápis *hodnoty do výstupního souboru*
přiřazení *hodnoty do last*
zvýšení *i o 1*
zvýšení *j o 1*
navrácení *intersection*

Struktura programu

Program sestává z 90 řádek včetně komentářů a odsazení a obsahuje pět metod.

První metoda *SequenceFromFile* slouží pro otevření souboru a načtení vstupních dat. Pomocí *try* a *except* bloků je odchytáván špatně zadaný název souboru či cesta k jeho umístění, nedostatečné oprávnění pro přístup k souboru a další případné chyby spojené s otevíráním a čtením souboru. V případě chyby je do konzole vypsána chybová hláška definující problém a program skončí. Po otevření souboru je také zkontrolováno, zda obsahuje nějaká data a v případě prázdného souboru je opět vypsána chybová hláška a program skončí. Funkce vrací seznam hodnot typu *string* načtených ze souboru.

Druhá metoda *WriteToFile* slouží pro zápis výsledné posloupnosti čísel a duplicitních prvků do výstupního souboru. Podobně jako v předchozí metodě jsou zde pomocí *try* a *except* bloků ošetřeny možné chyby při otevírání souboru. Výstupní soubor je otevřen v módu *a*, který umožňuje připisování dat na aktuální konec souboru a nedochází tak k přepisu již existujících dat.

Třetí metoda *StrToInt* převádí seznam s hodnotami typu *string* na seznam s hodnotami typu *integer*. Toho je docíleno pomocí cyklu, který postupně prochází prvky seznamu a převádí je. Pokud narazí na prvek, který není číslem a nejde tak převést na datový typ *integer*, vypíše chybovou hlášku a daný prvek přeskočí.

Čtvrtá metoda *SortList* slouží k vzestupnému setřídění posloupnosti. Algoritmus této metody je podrobněji popsán v předchozí kapitole, ale je založen na *for* cyklu s vnořeným druhým cyklem, který prochází dosud nesetříděné prvky a vybírá z nich minimum.

Pátá metoda *IntersectionOfLists* vrací průnik dvou vstupních seznamů. Přesný algoritmus je opět podrobněji popsán v předchozí kapitole, ale základem je střídavé načítání a porovnávání prvků obou posloupností. Pokud se rovnají, jedná se o shodný prvek pro obě posloupnosti odpovídající průniku a dojde tak k zápisu do výstupního seznamu.

V následující části kódu jsou nejprve do proměnných *sequence_1* a *sequence_2* funkcí *SequenceFromFile* uloženy posloupnosti ze vstupních souborů ve formě seznamu. Na tyto seznamy jsou aplikovány funkce *StrToInt* a *SortList* a výsledky jsou uloženy do seznamů *l1* a *l2*.

Na závěr je vytvořen nový seznam *intersection* a do něj uložen výsledek funkce *IntersectionOfLists.*

Vstupní a výstupní data

Program načítá vstupní data ze dvou textových souborů s názvy *input1.txt* a *input2.txt*, které obsahují na první řádce všechny celočíselné prvky posloupnosti oddělené mezerou. Při jiném názvu souborů je potřeba změnit kód na řádkách 84 nebo 85, v závorce změnit název souboru či upravit cestu k jeho umístění. Pokud by byly od sebe prvky odděleny jiným znakem než mezerou, je potřeba upravit kód programu na řádce 10 a to tak, že se daný znak zapíše do uvozovek do závorky funkce *split().*

Výsledná množina prvků představující průnik dvou vstupních množin je zapisována do textového souboru s názvem *output.txt*. Obdobně jako u vstupních souborů je v případě jiného názvu nutné upravit kód na řádce 77. Pro oddělení prvků výsledné posloupnosti jinak než mezerou je potřeba upravit kód na řádce 77 a to tak, že se v části *f"{element} "* mezera za složenou závorkou nahradí požadovaným znakem.

Problematická místa a možná vylepšení

Problematické může být zapisování do souboru v módu *append*, kdy dochází k připisování na konec souboru a časem je zde tak velké množství výstupů. Řešením by bylo nezapisovat prvky průniku posloupoností jednotlivě zároveň s jejich připojením do seznamu *intersection,* ale místo toho otevírat výstupní soubor v módu *write* a seznam *intersection* do něj zapsat celý najednou.

Pro zkrácení a zjednodušení programu by bylo možné vynechat funkci *StrToInt*, která je založena na *for* cyklu, který prochází všechny prvky posloupnosti a převádí je z datového typu *string* na *integer*. Stejného výsledku by bylo možné docílit například použitím funkce *map()* při načítání dat ze souboru. Tato možnost však neumožňuje upozornit na konkrétní chybné prvky a přeskočit je, tudíž by program skončil i při výskytu jen jednoho nečíselného prvku v posloupnosti.

Zdroje

GEEKSFORGEEKS (2021): Python Intersection of two lists, https://www.geeksforgeeks.org/python-intersection-two-lists/ (6.2.2022).

GEEKSFORGEEKS (2022): Union and Intersection of two sorted arrays, https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/amp/ (10.2.2022).

TUTORIALSPOINT (2022): Python - Sorting Algorithms, https://www.tutorialspoint.com/python_data_structure/python_sorting_algorithms (7.2.2022).

WIKIPEDIA (2022a): Intersection (set theory), https://en.wikipedia.org/wiki/Intersection_(set_theory) (6.2.2022).

WIKIPEDIA (2022b): Sequence, https://en.wikipedia.org/wiki/Sequence (5.2.2022).