Univerzita Karlova
Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

ÚVOD DO PROGRAMOVÁNÍ

Nalezení sjednocení dvou posloupnosti čísel

dokumentace

Lukáš Nekola
2. ročník, NKARTGD
Humpolec 2022

Obsah

1. Zadání

Úkolem je vytvoření programu, který bude schopen ze dvou posloupností čísel zaznamenat jejich sjednocení. Tyto dvě posloupnosti čísel budou zadány uživatelem, který jednotlivá čísla oddělí mezerou. Na počátek si mohu definovat, že uživatel bude moci také zadávat čísla s desetinnými místy.

2. Rozbor problému

Nejprve je nezbytné rozebrat problematiku sjednocení, jakožto výrokového problému. Dle webového portálu Matematická logika (2022) se jedná o operaci pro spojení dvou množin. Výsledná množina tedy obsahuje takové prvky, které se vyskytují alespoň v jedné z počátečních množin. Další pravidlo říká, že výsledná množina prvků nesmí obsahovat duplicitní hodnoty. V rámci úkolu se bude jednat o sjednocení dvou posloupností čísel typu *float*. Nicméně abychom byly korektní, tento fakt by mohl být aplikovatelný na více způsobů sjednocení věnující se seznamům či jednorozměrným polím.

3. Existující algoritmy a možná provedení

V prostředí Python je možné využít několik možností pro zjištění průniku sjednocení dvou seznamů hodnot. Prvním způsobem je použití *operátoru* | pro sjednocení dvou množin. Druhý způsob se opírá o využití funkce *union* fungující na stejném principu jako zmíněný operátor (GeeksforGeeks, 2021). Další možností využití sjednocení je vlastní naprogramování metody. Možným způsobem je vytvoření jedné množiny ze dvou existujících, následné odstranění duplicitních hodnot a jejich seřazení. Druhý způsob, který byl využit v rámci této práce, je vzestupné seřazení vstupních seznamů a následný střídavý zápis do nového pole hodnot (GeeksforGeeks, 2022).

Jelikož existuje mnoho způsobů řazení hodnot, rozhodl jsem některé z nich popsat. Prvním, nejrychlejším a nejpoužívanějším algoritmem je *Quick Sort*, který funguje na principu „rozděl a panuj“. Jedním z nejjednodušších algoritmů je tzv. *Bubble Sort*. Postupně prochází seznam a určuje prvky, které jsou ve špatném pořadí na základě sousedství. Dalším algoritmem je *Insertion Sort*, fungující na principu rozdělení seznamu na setříděnou a nesetříděnou část, přičemž z nesetříděné části jsou postupně jednotlivé

prvky přidávány do setříděné. Mezi další algoritmy řadíme například *Merge Sort* či *Selection Sort*, který byl využit v rámci této práce. (Tutorialpoits, 2022)

4. Popis zvoleného algoritmu

V této části hodlám popsat algoritmy, které byly využity v rámci této práce. Následující část rozdělím na část věnující se setřídění seznamu hodnot a samotného sjednocení. Sjednocení seznamu předchází setřídění a z toho důvodu bude popsáno jako první. Ve 4. kapitole jsem zmínil, že pro setřídění byl využit algoritmus s názvem *Selection Sort*. Jedná se o poměrně jednoduchý algoritmus sloužící pro malé množství dat. Princip je založen na procházení seznamu hodnot a nalezení nejmenší hodnoty. Tato hodnota je zařazena na počátek nového seznamu. Poté je celý postup opakovaný bez již seřazených hodnot. Tento proces je ukončen ve chvíli vyčerpání prvků v seznamu hodnot.

Nyní bych rád popsal algoritmus pro zjištění sjednocení dvou uspořádaných seznamů. Na začátku je nutné vytvořit dvě pomocné proměnné *i*, *j*. Popisovaným proměnným je přiřazena hodnota 0, která odpovídá prvnímu prvku seznamu. Každá z definovaných proměnných je určuje pozici prvku ve svém seznamu hodnot. Poté jsou porovnány dva prvky dvou seznamů, přičemž menší prvek je zapsán do nového pole a proměnná *i* či *j* je zvýšena o 1. Následně je porovnán druhý prvek prvního seznamu a první prvek druhého. Tento proces se opakuje do projití celých uspořádaných seznamů hodnot. Pokud je jeden ze seznamů delší než druhý, jsou zbývající hodnoty zapsány do nově vznikajícího pole. Dále je nezbytné ošetřit a odstranit případné duplicitní hodnoty z jednotlivých seznamů. Tento fakt je zajištěn podmínkou zápisu hodnoty do nového seznamu v případě rozdílnosti od předchozího zapsaného prvku. (GeeksforGeeks, 2020)

5. Struktura programu (datové struktury, metody)

Samotný program je vyvinut o celkové délce 71 řádků včetně komentářů a odsazení pro snazší přehlednost. Součástí programu jsou 3 metody:

a) V první části je definována metoda pro ošetření vstupu, ale také pro převod vstupních textových řetězců na seznam hodnot. Toto provedení je možné na základě jednotícího separátoru, kterým je mezera. Následuje cyklus, který postupně prochází jednotlivé prvky nově vzniklého seznamu a převede je do

formátu *float*. Ošetření správného formátu, resp. otestování hodnot při vstupu je zajištěno pomocí výjimky. V případě nevhodně zadaného formátu bude prvek vynechán a v následných částech nebude využit.

b) Následuje funkce pro seřazení hodnot, kdy její algoritmus je podrobněji popsán ve 4. kapitole. Nicméně bych dále uvedl, že pro zpracování *Selection Sort* se vyžívá dvou konečných cyklů, kdy jeden z nich je vnořený do druhého a prochází dosud neseřazené prvky. Z těchto prvků vybírá nejmenší z nich a zapíše ho do nového pole hodnot.

c) V této části se budu věnovat metodě pro určení sjednocení, která byla již opět definována ve 4. kapitole. V první části jsou tedy zjištěny délky již uspořádaných seznamů a hodnoty následně zapsány do proměnných *m* a *n*. Po vytvoření pomocných proměnných *i* a *j* s hodnotou 0 jsou v těle WHILE cyklu a podmínek v něm obsažených postupně zapisovány hodnoty do nově definovaného seznamu hodnot. Tento zápis je podmíněn rozdílnou hodnotou prvku, který byl zapsán v předchozí iteraci cyklu. Na konci metody jsou do nového seznamu zapsány zbývající prvky delšího původního seznamu.

d) Nyní hodlám popsat hlavní část programu, která komunikuje s uživatelem. Jedná se tedy o vytvoření dvou textových řetězců hodnot, přičemž jednotlivé prvky musí být odděleny mezerou. Tyto seznamy hodnot jsou nejdříve ošetřeny a převedeny na seznamy dle bodu a. Následně jsou seznamy seřazeny dle bodu b. Následuje vytvoření sjednocení dle bodu c.

e) V poslední části je výsledek sjednocení vypsán do konzole v případě, že jeho délka není rovna 0. Při zadání nekorektních hodnot či prázdných množin je uživatel informován.

6. Popis vstupních/výstupních dat

Za vstupní data považujeme 2 textové řetězce, které se zadávají do konzole, resp. příkazové řádky. Vstupní data by měla být v některém z číselných formátů. Výstup je opět vytisknut do konzole ve formátu *float*. Popisující program lze snadno upravit na možnost zadávání pouze celých čísel. Příklad vstupu a výstupu nalezneme v textovém dokumentu, který je přílohou této dokumentace.

7. Problematická místa

Za problematické místo můžeme považovat například výstup ve formátu *float*, který je méně přehledný než samotný formát pro definici celých čísel. Za další nevýhodu může být považován algoritmus pro seřazení hodnot z důvodu pomalejšího průběhu při větším objemu dat. Nicméně pro účely sjednocení je tento algoritmus dostačující. Dalším možným problémem je ošetření a informace o nulovém vstupu na konci programu.

8. Možná vylepšení

Možná vylepšení se týkají předchozí kapitoly. Nicméně bych dodal následující podněty. Dle různorodých požadavků by bylo možné ukončení programu při špatně zadaných hodnotách. Jejich přeskočení a upozornění na chybu je součástí provedeného algoritmu. Za další vylepšení můžeme považovat volitelný počet vstupních množin.

9. Pseudokód

Funkce listNumber (inputstring)
listNum = prázdný seznam hodnot
segSplit = rozdělení vstupu ve formátu vstupního řetězce dle separátoru
***cyklus** postupného procházení jednotlivých hodnot seznamu seqSplit*
***vložení** hodnoty ve formátu float do prázdného seznamu hodnot*
***výpis** špatného formátu, **vynechání** hodnoty*
***navrácení** listNum*

Funkce sortASC(inputList)
***cyklus** procházení pozic i seznamu hodnot*
minIndex = počáteční pozice
***cyklus** procházení pozic hodnot od i +1*
***pokud** je hodnota na pozici menší než hodnota na minIndex*
***zápis** hodnoty do minIndex*
***prohození** minimální hodnoty s porovnávanou hodnotou*
***navrácení** inputList*

Funkce unionFunction(list1, list2)
m, n = délka vstupních seznamů
i, j = inicializace na hodnotu 0
out = vytvoření prázdného seznamu
prev = pomocná proměnná s nulovou hodnotou
***cyklus dokud** i je menší než m a zároveň j je menší než n*
***pokud** list1[i] je menší než list2[j]*
***pokud** list1[i] je rozdílný od předchozí zapsané hodnoty (prev)*
***přiřazení** hodnoty list1[i] do seznamu out*
prev = přiřazení hodnoty list1[i]
***zvýšení** i o 1*

__pokud__ list2[j] je menší než list1[i]
__pokud__ list2[1] je rozdílný od předchozí zapsané hodnoty (prev)
__přiřazení__ hodnoty list2[j] do seznamu out
prev = přiřazení hodnoty list2[j]
__zvýšení__ j o 1
__pokud__ list2[j] se rovná list1[i]
__pokud__ list2[1] je rozdílný od předchozí zapsané hodnoty (prev)
__přiřazení__ hodnoty list2[j] do seznamu out
prev = přiřazení hodnoty list2[j]
__zvýšení__ i o 1
__zvýšení__ j o 1
__cyklus dokud__ i je menší než m
pokud hodnota list1[i] je rozdílná od hodnoty proměnné prev
přiřazení hodnoty list1[i] do seznamu out
prev = přiřazení hodnoty list1[i]
zvýšení i o 1
__cyklus dokud__ j je menší než n
__pokud__ hodnota list2[j] je rozdílná od hodnoty proměnné prev
__přiřazení__ hodnoty list2[j] do seznamu out
prev = přiřazení hodnoty list2[j]
__zvýšení__ j o 1
__navrácení__ seznamu hodnot out

__zadání__ 1. textového řetězce __uživatelem__
__úprava__ 1. textového řetězce dle metody __listNumber__ a seřazení dle metody __sortASC__
__zadání__ 2. textového řetězce __uživatelem__
__úprava__ 2. textového řetězce dle metody __listNumber__ a seřazení dle metody __sortASC__
res = volání funkce __unionFunction__ na 2 upravené textové řetězce

__pokud__ délka výstupu je nulová
__výpis__ obeznámení uživatele v terminálu
__pokud__ je délka výstupu nenulová
__výpis__ sjednocení do konzole bez okolních závorek a separátorů v podobě mezer

10. Zdroje

GeeksforGeeks (2020): *Union and Intersection of two sorted arrays*

https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/ (cit. 01.02.2022)

GeeksforGeeks (2021): *Union() function in Python*

https://www.geeksforgeeks.org/union-function-python/ (cit. 21.01.2022)

GeeksforGeeks (2022): Union and Intersection of two sorted arrays
https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/ (cit. 01.02.2022)

Matematická logika (2022): *Sjednocení množin*
https://www2.karlin.mff.cuni.cz/~portal/logika/?page=sjed (cit. 21.01.2022)

Tutorialpoints (2022): *Python - Sorting Algorithms*
https://www.tutorialspoint.com/python_data_structure/python_sorting_algorithms (cit. 01.02.2022)