ÚVOD DO PROGRAMOVÁNÍ

Dokumentace ke zkoušce

Daniel PUMR, 1. ročník Kartografie a geoinformatika

Praha, 16. 2. 2019

Zadání a cíle

Cílem této práce je popsat tvorbu dvou programů, které poskytují řešení k následujícím problémům:

Dány dvě posloupnosti čísel. Nalezení jejich sjednocení.

Vypočet vzdálenosti bodu P1[φ1, λ1] , P2[φ2, λ2] na kouli.

Ke každému problému je v této práci věnován jeden oddíl, kde je nejprve rozebrán problém, následně je navrženo řešení a případně jsou představeny obecné vzorce. V další části je potom popsáno fungování programu, jsou představena úskalí ve fungování a jak jsou nebo mohou být tyto problémy řešeny. **Dále je přidána dokumentace popisující fungování programu z hlediska uživatele.** Nakonec jsou v závěru navrženy možnosti vylepšení programů.

Sjednocení posloupností

Sjednocením je v matematice označována operace, při které ze dvou množin vzniká nová množina, která obsahuje prvky obou vstupních množin. Prvky jsou v množině zastoupeny právě jednou, jak ostatně vyplývá ze samotné definice prosté množiny (Přispěvatelé Wikipedie 2019). Posloupnost je funkcí, jejíž definičním oborem je množina přirozených funkcí, která zároveň určuje pořadí hodnot v posloupnosti, oborem hodnot pak může být libovolná množina (Havrlant 2014).

Základním problémem při definování obecného algoritmu je nutnost unikátnosti čísel ve výsledné množině, jinak by bylo velmi snadné množiny spojit. Konkrétně jsou problémem tedy čísla z průniku množin, který je nutné definovat a daná čísla zapsat do výsledné množiny pouze jednou. Definováním průniku jsou pak jasně určeny i doplňky v obou množinách, které jsou zbývající částí množiny sjednocení.

Řešení pomocí programovacího jazyka Python

Nejjednodušším řešením je definovat v pythonu vstupní posloupnosti čísel rovnou jako datový typ množina (anglicky set), nad nímž je možné standardní množinové operace, mezi které patří i sjednocení, rovnou provádět (Pilgrim 2011). Sjednocení je pak při programování provedeno metodou union(), která vrací novou množinu obsahující prvky z množiny, na kterou je metoda volána, a z množiny, která je dána jako parametr metody.

Problémem tohoto způsobu je skutečnost, že množina je v pythonu dána jako neuspořádaná kolekce hodnot, a proto není zachována posloupnost, ani nelze k jednotlivým hodnotám přistupovat podle definiční hodnoty. Proto musí být množina převedena na seznam a teprve pak seřazena.

Jinou možností by bylo pracovat pouze se seznamy a nevycházet z metody union, která ostatně ani nelze na seznamy aplikovat. V takovém případě je třeba určit průnik množin, určit tato čísla k zápisu pouze jednou a doplnit je o doplňky z množin.

V této práci byly vytvořeny dva programy, kde každý přistupuje k problematice jedním z těchto způsobů. Program nazvaný *posloupnost1.py* využívá množin a metody union. Druhý program nazvaný posloupnost2.py pak pracuje pouze se seznamy a výstupem je kromě sjednocení také informace o průniku a doplňcích.

Struktura programu *posloupnost1*

Prvním vstupem do programu je informace, zda chce uživatel pracovat s desetinnými čísly či nikoli. Tato informace je důležitá při převodu datového typu *string* získaného při zadání posloupností uživatelem na datový typ *float* či *integer*, kde *integer* je pro práci s celými čísly rozhodně přehlednější i vhodnější, ale po zadání desetinného čísla program končí s chybou.

Dalším vstupem jsou posloupnosti, jejichž sjednocení bude hledáno. Čísla zadává uživatel oddělené mezerou.

V další části zdrojového kódu jsou vstupní data, dosud uložená jako řetězec, převedena na množinu. Pomocí *if* a *else* jsou data převeden buď do formátu *integer* nebo *float*. To je učiněno pomocí metody *split* a funkcí *map* a *int/float*. Kde metoda *split* rozdělí řetězec podle mezer do seznamu, v němž je pak na jednotlivé prvky aplikována pomocí funkce *map* funkce *int*. Nakonec je výsledný seznam pomocí funkce *set* převeden na množinu. Celá tato část kódu je ošetřena na výjimku *ValueError* způsobenou zadáním jiného vstupu než čísel. V takové situaci je zobrazena chybová hláška a program je ukončen.

Nakonec je vytvořena nová proměnná, do které je pomocí metody *union* uložena množina sjednocení. ta je pak uživateli vytištěna do programu díky funkci *sorted* jako seřazený seznam.

Struktura programu *posloupnost2*

Stejně jako u předchozího programu je prvním vstupem do programu informace, zda chce uživatel pracovat z desetinnými čísly či nikoli. Druhým vstupem jsou posloupnosti, jejichž sjednocení bude hledáno.

V další části zdrojového kódu jsou stejně jako u předchozího programu vstupní data nejprve pomocí funkcí *map* a *int/float* převedena na číselný datový formát. Do proměnných jsou ovšem uložena oproti předchozímu programu jako seznam (*list*). Tato část kóde je také ošetřena na výjimku při zadání jiného vstupu než čísel.

Dále jsou vytvořeny proměnné s prázdným seznamem pro vznikající průnik a doplňky. V následujícím *for* cyklu pak program bere jednotlivé hodnoty v první posloupnosti a ověřuje, zda se nachází v průniku množin či se jedná o doplněk. Nakonec je pomocí dalšího *for* cyklu ověřen ještě doplněk v druhé množině. specifikem tohoto způsobu oproti předchozímu programu mimo je, že v případě zadání uživatelem většího počtu stejných hodnot do jedné posloupnosti, budou tyto hodnoty i ve výsledné posloupnosti vícekrát.

Průnik a doplňky jsou pak společně seřazeny pomocí funkce *sorted* do výsledného sjednocení. Nakonec program uživateli ukáže výsledné sjednoceni, ale také průnik a doplňky.

Fungování z pohledu uživatele

Tato část je napsána jako dokumentace pro uživatele k využití programu. Oba programy fungují z hlediska uživatele stejně s tím rozdílem, že v programu *posloupnost2* získá uživatel kromě sjednocení i informaci o průniku a doplňcích.

Po spuštění se program nejprve zeptá, zda chcete pracovat s desetinnými čísly. V případě, že chcete, aby dokázal program desetinná čísla zpracovat, na pište "ano". Program akceptuje také zápis "ANO" a "Ano". Pokud nezapíšete nic či cokoli jiného než jednu z vybraných možností, program bude pracovat pouze s celými čísly a po zadání desetinného skončí s chybou.

Dále je třeba postupně zadat vstupní posloupnosti, ze kterých bude vytvořeno sjednocení. Jednotlivá čísla oddělujte mezerou. Desetinná čísla zapisujte s tečkou. Pokud bude posloupnost zapsána s desetinnými čísly ačkoli tak nebyl program nastaven, desetinná čísla budou zapsána s čárkou, nebo budou-li použity jiné znaky než čísla, program skončí s chybou. V jedné posloupnosti by se nemělo jedno číslo objevovat dvakrát. V takovém případě se objeví dvakrát i ve sjednocené posloupnosti.

Po zadání těchto vstupů program ukáže výslednou sjednocenou posloupnost.

Závěr

Program se zdá být dobře funkční v obou přístupech. z hlediska programování je určitě jednodušší první přístup, kde samotnou tvorbu sjednocení zcela zajišťuje programovací jazyk. druhý přístup naopak umožňuje při případném rozšíření práce programu větší kontrolu nad hodnotami a zároveň rovnou poskytuje informace o průniku a doplňcích. Velmi snadno by také mohla být přidána možnost informovat i o rozdílu, aby informace nejzákladnějších množinových operacích byly kompletní.

Výpočet vzdálenosti bodů na kouli

Cílem programu je nalézt co nejkratší spojnici mezi dvěma body na kouli (ortodromu) a spočítat její délku. Základním předpokladem pro výpočet ortodromy je nalezení velikosti úhlu svíraného spojnicemi středu koule s danými body na povrchu. Výpočet tohoto úhlu vychází ze sférické trigonometrie, konkrétně je určen na základě kosinové věty na sférickém trojúhelníku následující rovnicí:

$$\sigma = \arccos(sin\varphi_1 \cdot sin\varphi_2 + cos\varphi_1 \cdot cos\varphi_2 \cdot \cos(\lambda_2 - \lambda_1)$$

Kde σ reprezentuje počítaný úhel a $\varphi_{1,2}$ $\lambda_{1,2}$ reprezentují souřadnice bodů. Po spočtení úhlu je pak při znalosti poloměru možné snadno určit vzdálenost na povrchu pomocí následujícího vztahu, kde je úhel zadán v radiánech:

$$d = \sigma \cdot r$$

Řešení pomocí programovacího jazyka Python

V Pythonu lze poměrně snadno zavést výpočet výše zmíněných rovnic pomocí základních aritmetických operací a pomocí goniometrických funkcí *sin*, *cos* a *acos*, které jsou v pythonu importovány z modulu *math*. Jedinou další komplikaci je nutnost dbát na výpočet v radiánech, což ovšem snadno řeší funkce *radians*, jež byla také importována z modulu *math*.

Struktura programu

Program se nejprve dotáže pomocí funkce input uživatele, s jakým chce pracovat poloměrem koule. Zadaná hodnota je následně pomocí funkce *float* převedena na číselný datový typ. Tato operace je ošetřena výjimkou, která působí, že pokud uživatel poloměr nezadá, nebo jej zadá chybně – nejedná se o čísla, je výpočet dále prováděn s hodnotou *6378,11 km*, což je střední poloměr planety Země. Po provedení těchto operací program informuje uživatele o tom, s jakou hodnotou pro průměr počítá.

Dalším vstupem od uživatele jsou souřadnice bodů, mezi kterými je vzdálenost počítána. Zde je také pomocí *try* a *except* ošetřena výjimka na chybu *ValueError*, která řeší situaci, kdy souřadnice nejsou zadány jako platné číslo. Kromě toho je pro každé zadání pomocí *while* cyklu nastaveno, že zadané hodnoty musí odpovídat možnému rozsahu šířky a délky. Pokud tomu tak není, vyzve program k novému zadání hodnoty.

Poté program informuje uživatele o hodnotách, s kterými provede výpočet a nakonec jsou provedeny samotné výpočty, kde jsou hodnoty také rovnou převedeny pomocí funkce *radians* na radiány.

Nakonec program vytiskne uživateli výslednou vzdálenost.

Fungování programu z hlediska uživatele

Po spuštění programu budete ze všeho nejdříve otázáni, jaký poloměr má koule, na která bude vzdálenost mezi body počítána. Hodnotu udávejte v kilometrech, desetinná čísla pište za tečku. Pokud hodnotu nezadáte, nebo zadáte hodnotu, která není v platném formátu, bude program počítat s hodnotou 6371,11, která reprezentuje střední poloměr planety Země.

Před dalším postupem vás program informuje o tom, kterou hodnotu přijal a bude s ní počítat. Následně je třeba zadat souřadnice bodů, pro které bude vzdálenost počítána. Šířku a délku bodu zadávejte ve stupních, desetinná čísla oddělujte tečkou. V případě, že zadáte hodnotu, které souřadnice nemohou nabývat, program vás o tom informuje a vyzve vás k opakovanému zadání. V případě, že zadáte jinou než číselnou hodnotu, program bude ukončen.

Po správném zadání souřadnic program souhrnně vypíše, se kterými hodnotami počítal a pod tím vypíše vzdálenost bodů v kilometrech.

Závěr

Program je založen na jednoduché aplikaci rovnice pro výpočet délky ortodromy. V tom by neměli nastávat při průběhu programu žádné závažnější problémy. Z hlediska dalších rozšíření programu by určitě bylo možné přidat do výstupu informaci o azimutu ortodromy u prvního z bodů. Případně by mohl program počítat i vzdálenost a azimut mezi body při propojení loxodromou, která poskytuje další možnost, jak body na kouli spojit.

Zdroje

HAVRLANT, L. (2014): Posloupnosti, Matematika.cz, https://matematika.cz/posloupnosti (16. 2. 2019).

PILGRIM, M. (2011): Přirozené datové typy, Ponořme se do Pythonu, http://diveintopython3.py.cz/native-datatypes.html#lists (17. 2. 2019).

PŘISPĚVATELÉ WIKIPEDIE (2019): Množina, Wikipedia, https://cs.wikipedia.org/wiki/Množina (16. 2. 2019).