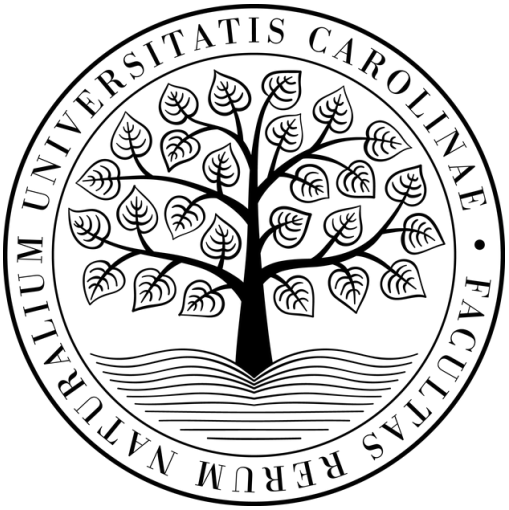
Univerzita Karlova

Přírodovědecká fakulta

Studijní program: Geografie a kartografie



Theodor Balek

2. ročník studijního programu Geografie a kartografie

Převod čísla z dvojkové do desítkové soustavy a zpět; Interpolace pomocí metody IDW

Dokumentace programů

Zkouškové úlohy

Předmět „Úvod do programování“

Akademický rok 2023/2024

Praha, 12. 2. 2024

Převod čísla z dvojkové do desítkové soustavy a zpět

Zadání

Převod čísla c, c ∈ Z^+, ze dvojkové soustavy do desítkové a naopak.
Zadané kladné reálné číslo převeďte z dvojkové (binární) soustavy do soustavy desítkové (decimální) a zpět. Zvolte reprezentaci (n+m) představující počet bitů pro celočíselnou a desetinnou část čísla. Určete, s jakou absolutní a relativní chybou je číslo uloženo (celé, reálná, reálná periodická, …), otestujte na několika různých příkladech (simulace float, double).

Algoritmus

Převod čísla z dvojkového do desítkového tvaru je založen na postupném násobení cifer dvojkového tvaru mocninami čísla dvě zvlášť pro celou a desetinnou část. Uveďme příklad převodu čísla 1010,0101. Jeho celou část tvoří 1010 a jeho desetinnou část tvoří 0101. Převod probíhá zprava doleva, tedy $0101_2 = (1 \cdot 2^{-4} + 0 \cdot 2^{-3} + 1 \cdot 2^{-2} + 0 \cdot 2^{-1})_{10} = 0{,}3125_{10}$ a $1010_2 = (0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3)_{10} = 10_{10}$. Výsledné číslo v desítkové soustavě je tedy 10,3125.

```
p = -(počet číslic desetinné části binárního čísla)
pro číslice v desetinné části binárního čísla (zprava doleva):
      desítkové číslo += číslice * 2**p
      p += 1
pro číslice v celé části binárního čísla (zprava doleva):
      desítkové číslo += číslice * 2**p
      p += 1
```

Převod opačným směrem probíhá pomocí postupného celočíselného dělení čísla dvěma a zápisu zbytků po tomto dělení, které probíhá pro celou část, a desetinná část se převádí postupným násobením dvěma zapisováním změny řádu. Použijme číslo 10,3125. $10:2 = 5\ (0)$; $5:2 = 2\ (1)$; $2:2 = 1\ (0)$; $1:2 = 0\ (1)$. Zbytky po dělení zapisujeme zprava doleva, celá část binárního tvaru čísla 10,3125 je tedy 1010. U desetinné části je třeba zjistit změnu řádu při vynásobení dvěma. $0{,}3125 \cdot 2 = \mathbf{0},625$; $0{,}625 \cdot 2 = \mathbf{1},25$; $0{,}25 \cdot 2 = \mathbf{0},5$; $0{,}5 \cdot 2 = \mathbf{1},0$. Binární tvar čísla 0,3125 je tedy 0101. 10,3125 se tedy převede zpět na číslo 1010,0101.

```
dokud celá část desítkového čísla >= 2:
      přidat (desítkové číslo % 2) do celé části binárního čísla
      desítkové číslo = desítkové číslo // 2
přidat (desítkové číslo % 2) do celé části binárního čísla
pro k v rozsahu desetinné části binárního čísla:
      přidat celou část dvojnásobku desetinné části desítkového čísla
            do desetinné části binárního čísla
      upravit desetinnou část, pokud dvojnásobek začíná na 1, tak, aby
            začínal na 0
```

Program

Program je napsaný v jazyce Python a využívá objektově orientovaného programování. Na začátku programu je navržena třída *bin*, která představuje vstupní číslo v binárním tvaru. Během inicializace této třídy je kontrolováno, zda je vstup zadaný uživatelem skutečně číslo, zda je v binárním tvaru a zda je kladné. Pakliže ano, přiřadí program číslu vlastní *self.__id* a rozdělí jej do dvou seznamů, *self.__n* pro celou část a *self.__m* pro desetinnou část.

Metoda *print* vypíše informace o zadaném čísle, a to jeho *self.__id*, jeho binární tvar a celou (*self.__n*) a desetinnou (*self.__m*) část tohoto binárního tvaru.

Metodou *to_dec* probíhá převod z binárního do decimálního tvaru pomocí výše popsaného algoritmu. Exponent *p* čísla 2 je zvolen jako záporná hodnota délky seznamu desetinné části (*self.__m*) a v každé iteraci seznamů *self.__m* i *self.__n* zprava doleva je tento exponent *p* zvětšen o 1. Výsledkem této metody je vypsání decimálního tvaru zadaného čísla.

Metoda *to_bin* převádí desítkový tvar získaný v metodě *to_dec* zpět do tvaru dvojkového pomocí algoritmu dělení celé části dvěma a násobení desetinné části dvěma. Celá a desetinná část dvojkového tvaru jsou ukládány do seznamů *self.__N* a *self.__M*. Výsledkem této metody je vypsání binárního tvaru čísla.

Program nejprve inicializuje proměnnou *a*, kterou uživatel zadá jako vstup, podle třídy *bin*. Poté jsou volány metody *print*, *to_dec* a *to_bin*.

Interpolace pomocí metody IDW

Zadání

Interpolace s využitím metody IDW.
Na vstupu je seznam souřadnic bodů P={pi}, kde pi = [xi, yi, zi] načtený ze souboru. Pro zadaný bod q=[x,y] určete metodou prostorové interpolace IDW hodnotu z(q). Váhy volte různými způsoby, a to jako reciproké hodnoty vzdáleností nebo čtverců vzdáleností. Pokud bod q padne mimo konvexní obálku P, informujte uživatele, že výsledkem bude extrapolovaná hodnota.

Algoritmus

Interpolace metodou IDW (Inverse Distance Weighted) počítá hodnotu v neznámém bodě na základě hodnot okolních bodů jako vážený průměr jejich hodnot, kde vzdálenost známého bodu od bodu s hledanou hodnotou je onou vahou. Čím větší je tato vzdálenost, tím menší je vliv bodu na výslednou hodnotu. Jedná se tedy o nepřímou úměru vyjádřenou $\frac{1}{r^{\alpha}}$, kde *r* představuje vzdálenost a *α* parametr metody, který se obvykle volí 1 pro větší vliv vzdálenějších bodů, nebo 2 pro snížení vlivu vzdálenějších bodů. Pro každý bod se známou hodnotou je třeba vypočítat jeho vzdálenost od bodu s hledanou hodnotou, poté z této vzdálenosti vypočítat váhu onoho bodu. Váhy je třeba znormovat, aby jejich součet činil 1,

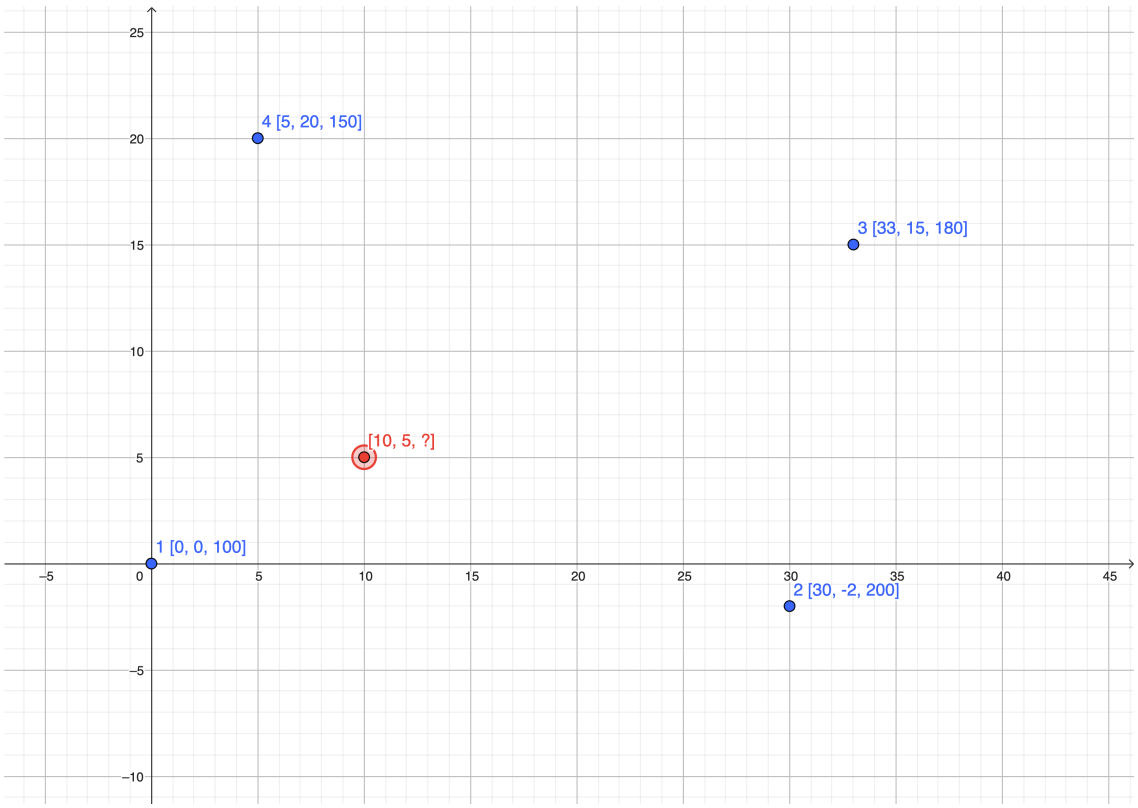
což se provede vydělením každé váhy součtem vah všech bodů. Dále se znormované váhy vynásobí hodnotami daných bodů a hledaná hodnota je sumou těchto dílčích hodnot.
Výpočet hodnoty lze uvést na příkladu, který je součástí testovací sady programu (test_4.txt). Na vstupu jsou čtyři body se známými souřadnicemi a výškou (obr. 1). Jednotlivé výpočty jsou zaneseny do tabulky 1. Nejprve je třeba vypočítat vzdálenost bodů od neznámého bodu pomocí rozdílu souřadnic a Pythagorovy věty. Následuje výpočet váhy podle zvoleného koeficientu, znormování váhy podle součtu vah a výpočet výšky vynásobením *z* souřadnicí. Součet těchto dílčích výšek pak tvoří hledanou hodnotu.

```
pro každý bod ze seznamu vstupních bodů:
      váha += 1/(vzdálenost**α)
pro každý bod ze seznamu vstupních bodů:
      z_souřadnice += ((1/(vzdálenost**α))/váha) * výška daného bodu
```

hledaný bod … [10,5,***?***]

bod	vzdálenost r	váha $1/r^{\alpha}$, α = 2	znormovaná váha (váha : 0,0158)	výška (znorm. váha · výška)
1 [0,0,100]	11,18	1/125	0,506	50,57
2 [30,-2,200]	21,19	1/449	0,141	28,16
3 [33,15,180]	25,07	1/629	0,1	18,1
4 [5,20,150]	15,81	1/259	0,253	37,93
součet		0,0158	1	***134,76***

Tab. 1: Výpočet neznámé výšky pomocí metody IDW.

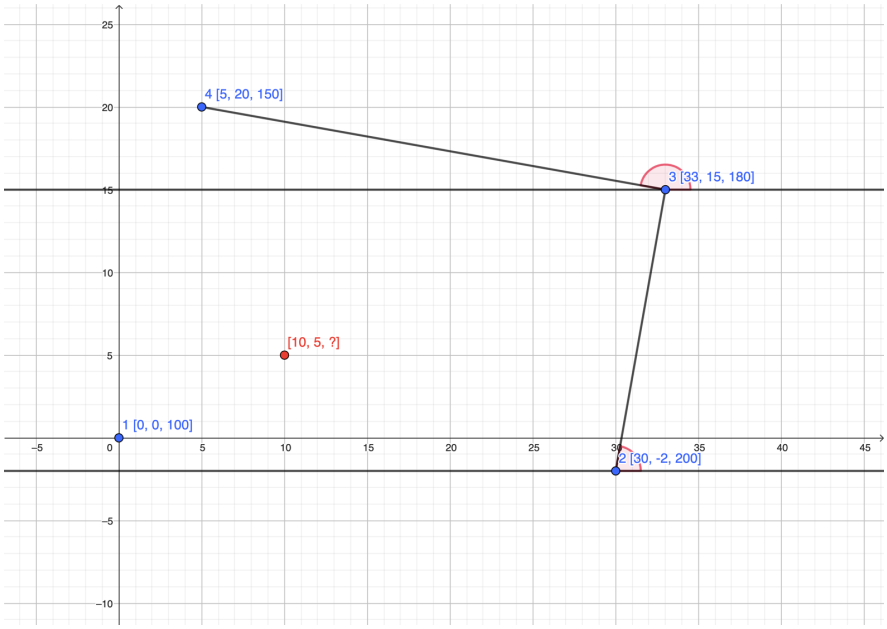


Obr. 1: Vizualizace testovacího souboru test_4.txt.

Další algoritmus použitý v programu se týká nalezení konvexní obálky a určení, zda se neznámý bod nachází uvnitř, či vně, tedy zda jde o interpolaci, či extrapolaci. K výpočtu konvexní obálky zadaných bodů je využit algoritmus *Graham Scan*. Ten nejprve nalezne bod s nejmenší souřadnicí y a poté seřadí ostatní body podle polárního úhlu, který svírá daný bod s přímkou rovnoběžnou s osou x procházející bodem s nejnižším y. Následně probíhá iterace bodů v tomto pořadí a porovnávání polárních úhlů každého bodu s bodem předchozím a před předchozím (obr. 2). Pokud je rozdíl úhlů kladný (úsečka se láme doleva), pak je daný bod součástí konvexní obálky. Ovšem pokud jsou úhly shodné či je jejich rozdíl záporný (úsečka se láme doprava), pak je daný bod z hranice konvexní obálky vyřazen a pokračuje porovnávání stále stejného bodu s novými předcházejícími body, dokud není rozdíl kladný.

```
konvexní obálka = []
pro i v rozsahu vstupních bodů:
      dokud délka konvexní obálky >= 2 a orientace aktuálního,
      předchozího a před předchozího bodu není levotočivá:
            vyjmi předchozí bod z konvexní obálky
      přidej aktuální bod do konvexní obálky
```

Graham Scan byl zvolen díky své efektivitě, jeho časová náročnost činí $n \cdot \log n$. Existuje více variant nalezení konvexní obálky, například algoritmus *Gift wrapping*, který začíná od bodu s nejmenší souřadnicí x a postupně prochází všechny body, které ovšem porovnává s veškerými ostatními body, a je tak značně neefektivní (n^2). Zajímavý je též algoritmus *Quickhull*, který rozdělí body dvě části podle přímky spojující bod nejvíce vlevo s bodem nejvíce vpravo a hledá nejvzdálenější body od této přímky. Poté opět dělí rovinu na dvě části a hledá nejvzdálenější body, dokud neleží všechny body v konvexní obálce.



Obr. 2: Tvorba konvexní obálky pomocí Graham Scan.

Program

Program v jazyce Python navrhuje dvě třídy, a to *inpt* a *point*. Ve třídě *inpt* je cílem zpracovat vstupní data (seznam bodů) z textového souboru. Každý bod musí mít v textovém souboru vlastní řádek a být zapsán ve formátu *id;x;y;z*. Program pak z tohoto souboru vytvoří seznam každého bodu a tyto seznamy uloží do společného seznamu *self.__points*. Metoda *inptprint* slouží k vypsání všech bodů a metoda *getpoints* vrací seznam bodů pro další využití.

Třída *point* je inicializována vstupním bodem, jehož výšku požaduje uživatel vypočítat. Uživatel zadává souřadnice x a y a parametr metody IDW. Program kontroluje, zda jsou vstupy číselného formátu a zda je parametr power kladný. Power se zpravidla zadává 1, nebo 2, ovšem program umožňuje zadat jakékoli přirozené číslo. Metoda *orientation* se využívá při hledání konvexní obálky a určuje rozdíl úhlů při *Graham Scan*, jenž je popsaný výše. Metoda *convex* určí, zda se jedná o interpolaci pomocí metod *lowest_y* a *sort_angle*. Metoda *calc_z* vypočítá výšku požadovaného bodu.

V metodě *calc_z* se nejprve vypočítá součet vah všech bodů *self.__w*, následuje výpočet výšky *self.__zet* postupným přičítáním dílčích výšek okolních bodů pomocí již popsaného algoritmu. Jednotlivé souřadnice jsou volány pomocí indexů. Pokud uživatel zadá bod, který se nachází ve vstupních datech, a jehož výška je tedy známá, výpočet je přeskočen a dojde k vypsání dané známé hodnoty. K tomu slouží proměnná *self.__cond*, která nabývá hodnoty 0 (zadán shodný bod), resp. 1 (zadán odlišný bod).

Hlavní myšlenka určení polohy bodu vzhledem ke konvexní obálce (metoda *convex*) spočívá v jeho přidání ke vstupním bodům a výpočtu konvexní obálky. Pokud se bod stane součástí hranice konvexní obálky, pak se jedná o extrapolaci; obálka se musela zvětšit. Pokud bod není součástí hranice konvexní obálky, jedná se o interpolaci; bod byl během algoritmu vyjmut a nachází se uvnitř. Algoritmus nejprve volá metodu *lowest_y*, která vrátí bod s nejnižší souřadnicí *y* ze vstupních bodů. Pomocí funkce tangens dochází následně k výpočtu polárního úhlu, který je přidán do seznamu ke každému bodu (*id;x;y;z;úhel*), a k novému seřazení bodů dle tohoto úhlu pomocí metody *sort_angle*. V případě, že mají dva body stejný úhel, jsou řazeny podle vzdálenosti od *self.__p0*. Po seřazení je informace o úhlu ze seznamu opět vyjmuta pro porovnání na konci metody.

Self.__hull je seznam bodů tvořících konvexní obálku. Nejprve se do seznamu přidají první dva body a každý další přidaný bod je porovnáván podle směru lomu úsečky vypočteném metodou *orientation*. Pokud se úsečka láme doprava, je předchozí bod vyjmut. Výsledná konvexní obálka je porovnávána se vstupním bodem uživatele. Pokud se bod nachází v konvexní obálce, jedná se o extrapolaci, v opačném případě o interpolaci.

Speciálním případem je zadání bodu obsaženého ve vstupních hodnotách, v takovém případě se jedná o interpolaci, což je ošetřeno proměnnou *self.__cond*.

Výstupem metody *calc_z* je tedy vypsání hledané výšky zadaného bodu a informace o poloze bodu vzhledem k bodům vstupním.

V programu je inicializována proměnná *q* podle třídy point, kde uživatel zadá cestu k souboru se známými body, souřadnice x a y bodu neznámé výšky a parametr metody IDW. Volána je metoda *calc_z* a *convex*.

Program byl testován na 4 souborech (test_1-4.txt) s různým počtem bodů, které nabývaly jak datového typu integer, tak datového typu float.

Zdroje

Convex Hull Algorithm. *GeeksforGeeks* [online]. 2023 [cit. 12. 2. 2024]. Dostupné z: https://www.geeksforgeeks.org/convex-hull-algorithm/?ref=lbp