Setřídění posloupnosti metodou Select Sort

Zadání

Úkolem tohoto programu je seřadit posloupnost reálných čísel pomocí metody Select Sort. Vstupní data jsou zadána v textovém souboru. Výstup bude nahrán do nového textového souboru. Další požadavek na program je, aby uživatel zadal vstupní parametr pro seřazení řady vzestupně či sestupně.

Popis a rozbor problému

Jako první bude potřeba načíst data z textového souboru a uložit je do formátu, se kterým se bude nejlépe pracovat. Takto upravená data je poté potřeba seřadit Metodou Select Sort. Nakonec seřazenou posloupnost vyexportujeme jako textový dokument.

Popis algoritmů

Algoritmus použitý v programu se nazývá Select Sort. Jedná se o řadící algoritmus s časovou náročností $O(N^2)$. Algoritmus pracuje s tím, že v nahrané posloupnosti najde minimum, které pak prohodí s prvním prvkem. V dalším kroku program opět hledá minimum, ale nyní již vynechává první prvek. Po nalezení prohodí opět pozici minima a tentokrát 2. prvku. Algoritmus takto postupně projde celou posloupností až je seřazena. Pro řazení sestupně postupuje stejně, jen hledá maximum a ne minimum.

Pro větší datové soubory se tato metoda nevyužívá. Pro takové případy je spíše využita metoda Řazení haldou nebo Řazení slučováním, které mají náročnost O(NlogN).

Problematické situace

Problematické situace mohou nastat především u načítání dat z textového souboru. Data buď nemusí být nalezena, nebo mohou být uložena v nesprávném formátu. Toto je ošetřeno ukončením programu a upozorněním uživatele na daný problém. Další problém může nastat při výběru, zda se posloupnost bude řadit vzestupně, nebo sestupně. Program se při nedefinované metodě zeptá znovu.

Vstupní data

Neseřazená posloupnost je v textovém dokumentu s názvem *vstup_posloupnost.txt.* Data jsou zde tak, že na každém řádku je jedno reálné číslo posloupnosti. Desetinné místo čísel je odděleno tečkou (př. 10.02).

Výstupní data

Výstup je v textovém souboru *vystup_posloupnost.txt* a data jsou zde uložena, že na každém řádku je jeden prvek posloupnosti a desetinná místa čísel jsou oddělena tečkou.

Dokumentace

Program je rozdělen do 5 částí. První část je zpracování vstupního souboru ve formátu *.txt* a nahrání jeho obsahu do datové struktury *list.* Pomocí funkce *open* je nahrán textový soubor, dále jsou všechny jeho řádky nahrány do listu názvem *alist*. Každý řádek je reprezentován jako jedna položka listu ve formátu *float*. Program zde počítá také s některými problémy, které mohou nastat a v takovém případě vrátí chybovou hlášku a ukončí se. Takové případy jsou, že soubor vstupních dat nebyl nalezen, nebo program není oprávněn tento soubor otevřít.

Další funkce programu *sort(alist,method)* má 2 vstupní parametry, a to list nesoucí posloupnost reálných čísel a metodu, jakou budou data řazena. Funkce využívá dvou do sebe vnořených cyklů, kde zjišťuje, na jaké pozici se nachází minimum, nebo maximum (v závislosti na vybrané metodě) aktuální neseřazené posloupnosti. Když jej nalezne, tak se prvky na pozici minima a aktuální iterované pozici prohodí. Výstupem této funkce je vzestupně seřazená posloupnost *alist* ve formátu *list.*

Pro porovnávání jednotlivých prvků slouží funkce *comp(a,b,method)*. Funkce má 3 parametry, *a, b* jsou porovnávaná čísla a *method* je informace, zda se hledá minimum, či maximum. Funkce vrací jedno číslo, a to maximum nebo minimum.

Pro zjištění jakou metodou bude posloupnost řazena slouží funkce *metoda()*, která se zeptá uživatele. Při nekorektní odpovědi se zeptá znovu. Nakonec uloží metodu řazení do proměnné *method* ve formátu string.

Pro zajištění výstupu seřazené posloupnosti ve formátu *.txt* slouží poslední funkce programu *vystup(alist),* která má vstupní parametr již seřazenou posloupnost a zapíše ji do textového souboru, že každý prvek posloupnosti je na jednom řádku.

Závěr

Program řeší problémy nevalidních vstupních dat tím, že zahlásí chybu a ukončí se. Zde by bylo určitě možné alespoň zjistit podrobnější informaci o chybě. Déle program neřeší problém 2 a vice stejných čísel posloupnosti a nezanechává jejich pořadí v původní posloupnosti. To v některých případech může být užitečné.

Seznam literatury

Lekce 1 - Selection sort. *ITnetwork.net* [online]. [cit. 2021-02-12]. Dostupné z: https://www.itnetwork.cz/navrh/algoritmy/algoritmy-razeni/algoritmus-selection-sort-razeni-cisel-podle-velikosti/

Ověření, zda bod leží uvnitř konvexního mnohoúhelníku

Zadání

Pomocí metody Winding number, nebo Ray Crossing zjistěte, zda je bod uvnitř, nebo vně konvexního mnohoúhelníku. Body definující mnohoúhelník i hledaný bod jsou v jednom textovém souboru. Výstup s výsledkem bude také v textovém souboru.

Popis a rozbor problému

Jako první je potřeba získat informaci o polygonu a hledaného bodu z textového souboru a načíst je do formátu, ve kterém se s nimi bude nadále pracovat. V zadání je uvedeno, že má být použita metoda Winding number, nebo Ray Crossing. V tomto případě byl použit algoritmus Winding number. Pomocí jej je zjištěna poloha bodu vůči mnohoúhelníku a ta je vrácena jako textový soubor.

Popis algoritmu

Pro určení polohy body byl zvolen algoritmus Winding number. Metoda je založena na předpokladu, že když máme polygon *P* s vrcholy p_i a pozorovatel stojí v bodě *q* a chce vidět všechny p_i, tak když je uvnitř polygonu musí se otočit o 2π, když stojí vně, tak se otočí o úhel menší než 2π. To znamená, že suma všech úhlů p_i, *q*, p_{i+1} musí být rovna 2π pro *q* uvnitř polygonu. Tento výpočet lze aplikovat jak pro konvexní, tak nekonvexní polygony. Jedná se však o časově náročnou metodu, protože je zde pro zjištění kvadrantů úhlů zapotřebí vypočítat inverzní trigonometrickou funkci tan2.

Pro konvexní polygon lze tento algoritmus zjednodušit a pro zjištění polohy bodu vůči polygonu nám stačí vypočítat vektorový součin $w = u \times v$, kde $u = p_{i+1} - p_i$, $v = q - p_i$. Když je bod uvnitř, tak všechny součiny vyjdou se stejným znaménkem. Když počítáme protisměru hodinových ručiček, tak pro bod ležící vpravo od přímky definující hranu polygonu vyjde součin větší než 0, pro bod nalevo méně než 0 a pro bod na přímce 0.

Časová náročnost této metody je O(N).

Problematické situace

Problematické situace mohou nastávat u načítaní vstupních dat. V takovém případě program zahlásí chybu a ukončí se. Další problematické místo může nastat, když hledaný bod leží kolineárně s hranou polygonu nebo je přímo ve vrcholu. Tento problém je vyřešen, že body nacházející se na hraně polygonu jsou brány jako body mimo polygon. Vektorový součet u bodů na hraně a ve vrcholu je nula a program vyhodnotí bod jako mimo polygon.

Další problematická situace je, že nevíme, zda je polygon definován v CW, nebo CWW orientaci a u algoritmu Winding number se podle toho dovíjí znaménko vektorového součtu. Tento problém je odstraněn tím, že mezi sebou je vynásoben test předchozí a aktuální, když je <= 0, je bod mimo. Když je bod uvnitř je u všech vektorových součtů znamínko stejné a po jejich vynásobení je vždy výsledek kladný.

Poslední problém by mohl nastat, že zjednodušená verze výpočtu Winding number nefunguje u nekonvexních mnohoúhelníků. Z toho důvodu je zde proveden test, zda jsou všechny vnitřní úhly menší než 180 stupňů. Ale zde zase neznáme orientaci bodů polygonu a tím pádem při výpočtu úhlů u polygonu v CCW orientaci zjistíme velikost vnějších úhlů. Této informace je využito tak, že když jsou všechny úhly <180, nebo všechny úhly> 180, tak je konvexní.

Vstupní data

Data bodů, jak definující polygon, tak kritický bod, jsou uloženy v textovém souboru s názvem *vstup_polygon.txt*. Data jsou zde zapsána, že na prvním řádku jsou souřadnice

kritického bodu a na dalších řádcích body definující polygon. Každý bod je na novém. Souřadnice jednotlivých bodů jsou ve formátu souřadnice x „mezera“ souřadnice y. Souřadnice x a y jsou celá čísla (př. 5 -4).

Výstupní data

Výstup je uložen v textovém souboru *vystup_polygon.txt* a obsahuje textový řetězec s výsledkem.

Dokumentace

Program obsahuje 2 třidy. První, která se jmenuje *Points,* zajišťuje ukládání a práci s body. Ukládá do sebe souřadnice x a y bodu. Dále zajišťuje volání jednotlivých souřadnic bodů a jejich reprezentaci a případnou editaci pomocí setterů. Druhá třída s názvem *Polygon* nahraje list bodů uložených ve formátu třídy *Point,* který pak definuje polygon. Body jsou v listu seřazeny za sebou pravotočivě.

Třída *Polygon* obsahuje 3 další funkce. První funkce má za úkol vytvořit pomocí cyklu vytvořit dvojce bodů, které definují jednotlivé hrany polygonu a uložit je do listu. Funkce se nazývá *edges().* Výstupem této funkce je list dvojic bodů definující hrany polygonu.

Další funkce s názvem *konvex()* zajišťuje zda je zadaný polygon konvexní a lze na něj tedy uplatnit zjednodušenou formu algoritmu winding number. Funkce načte vždy trojici sousedních bodů a spočte úhle, který uloží do listu *uhly*. Takto projede postupně úhly u všech vrcholů polygonu. Nakonec se funkce podívá do listu *uhly*, a když jsou všechny úhlu buď <180, nebo všechny úhly> 180, tak je konvexní.

Poslední funkce třídy *Polygon* se jmenuje *contains(point)*. Do této funkce vstupuje ještě jedna vnější proměnná ve formátu třídy *Point*, která definuje zkoumaný bod. Do funkce vstupuje jako *point.* Dále pracuje s listem hran vytvořených pomocí funkce *edges().* Pomocí těchto proměnných se vypočte vektorový součet. Kvůli tomu že neznáme orientaci polygonu nevíme, zda mají být výsledky kladné, či záporné. Tento problém je vyřešen, že porovnáváme vždy aktuální a předchozí test, když jeden je nula, nebo mají rozdílná znaménka, tak je bod vně polygonu a program může ukončit testování a vrátit výsledek v podobě textového souboru. Když jsou znaménka shodná u všech testů. Je bod uvnitř.

Vstupní data jsou načítána z textového souboru, ve kterém je na každém řádku jeden bod. Na prvním řádku je zkoumaný bod a na dalších body definující polygon. První bod je uložen jako *bod* ve formátu *Point*. Zbytek bodů je nahrán do listu *alist*, kde každá položka obsahuje vrcholový bod polygonu ve formátu *Point*. Když vstupní soubor není nalezen, nebo program nemá právo jej číst, je program ukončen a vrací chybovou hlášku. Při vstupu jsou jednotlivé souřadnice rozděleny a z formátu *string* uloženy do *formátu int.*

Závěr

Nejdůležitější je asi zdůraznit, že program body na hraně polygonu bere jako body mimo. Původně jsem uvažoval o řešení pomocí algoritmu Ray crossing, ale narazil jsem na problém se singulárními případy a připadalo mi jednodušší využít algoritmus winding number než je ošetřovat. Dále jsem původně přemýšlel o zpracovávání vstupních jen jako listů čísel, ale

nakonec jsem se uchýlil k využití prcku objektově orientovaného programování a data jsem nahrával do tříd.

Seznam literatury

Is the Point Inside the Polygon? *Towards: Data Science* [online]. [cit. 2021-02-12]. Dostupné z: https://towardsdatascience.com/is-the-point-inside-the-polygon-574b86472119