Univerzita Karlova v Prahe
Přírodovědecká fakulta
Katedra aplikované geoinformatiky a kartografie

Úvod do programování

zadanie č. 122 – Algoritmus Douglas-Peucker

technická správa

Daniela DANČEJOVÁ
Geografie a kartografie, 2. ročník
Čierne Pole, 2022

Zadanie

Vytvorte program, ktorý zjednoduší vstupnú lomenú čiaru pomocou algoritmu Douglas-Peucker.

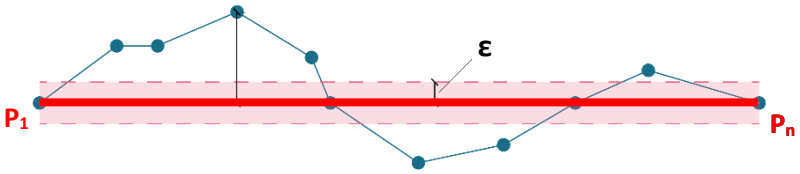
Rozbor úlohy

Douglasov-Peuckerov algoritmus, často označovaný aj ako Ramerov-Douglasov-Peuckerov algoritmus je generalizačný algoritmus, ktorý zjednodušuje priebeh líniových prvkov [1], [2]. Z krivky definovanej určitým počtov uzlov vytvorí krivku s menším počtom uzlov tak, aby sa čo najlepšie zachoval jej pôvodný tvar, pričom množstvo eliminovaných bodov závisí na nastavení parametrov algoritmu. Tento algoritmus nachádza využitie predovšetkým v kartografii a geoinformatike.

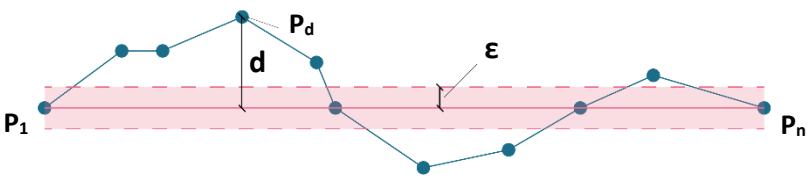
Popis algoritmu

Algoritmus prebieha na krivke C definovanej n bodmi P ($C = (P_1, P_2, P_3, ..., P_n)$) a parametrom ε, ktorý vyjadruje najmenšiu prípustnú vzdialenosť pre zachovanie uzlu.

Prvým krokom je základná aproximácia krivky spojením bodov $\overline{P_1P_n}$, ktoré budú tvoriť začiatok a koniec zjednodušenej línie; tieto body nebudú algoritmom nikdy odobraté.



Následne algoritmus prejde zostávajúce body $P_2, ..., P_{n-1}$ a nájde bod P_d, ktorý sa od spojnice bodov $\overline{P_1P_n}$ nachádza zo všetkých zvyšných bodov najďalej vo vzdialenosti d.



Vzdialenosť d sa porovná s hodnotou ε; ak platí vzťah:

$$d > \varepsilon ;$$

tak sa bod P_d v krivke **ponechá** a krivka C sa rozdelí na segmenty $C_1 = (P_1, P_2, ..., P_d)$ a $C_2 = (P_d, P_{d+1}, ..., P_n)$, na ktorých je algoritmus opäť aplikovaný. Rekurzia sa zastaví, keď $d \leq \varepsilon$;

rekurzia sa vtedy ukončí a z krivky sa vyhodia všetky neponechané body, výsledkom je krivka zachovávajúca tvar pôvodnej krivky s menším počtom uzlov.

Ilustrácie prevzaté z: [1]

Vstupné dáta

Vytvorený program požaduje vstupné dáta vo formáte *JSON*, resp. *GEOJSON*. Vstup by mal bezpodmienečne obsahovať zoznam usporiadaných dvojíc súradníc bodov x a y, ktoré charakterizujú priebeh krivky.

K programu sú priložené vzorové dáta *sample_line.json*, kde sú súradnice bodov umiestnené v ceste, ktorá je preddefinovaná exportom v softvéri ArcGIS Pro:

["features"][0]["geometry"]["paths"][0].

S touto cestou program pracuje a v prípade jej nezachovania nebude fungovať. Vlastné umiestnenie súradníc je potrebné špecifikovať nahradením tejto cesty priamo v skripte na riadku 118:

```
with open (infilename, encoding = "utf-8") as sample_line:
        lines = json.load(sample_line)
        points = lines["features"][0]["geometry"]["paths"][0]
```

Podobne program preddefinovane pracuje so súborom pomenovaným ako priložené zdrojové dáta. Vlastný názov sa dá definovať prepísaním hodnoty *"sample_line.json"* na riadku 111. Je potrebné, aby sa vstupný súbor nachádzal v rovnakom adresári ako program.

Vzorové dáta používajú súradnicový systém S-JTSK. Na súradnicovom systéme vlastných vstupov nezáleží, avšak treba dbať na nastavenie hodnoty ε, ktorá je pre vzorové dáta a ukážku algoritmu nastavená na vyššiu hodnotu (50), ako môže byť pre iné súradnicové systémy potrebné.

Výstupné dáta

Prvým výstupom programu je nový súbor vo formáte *JSON* obsahujúci zjednodušenú vstupnú krivku. Preddefinovaný názov výstupu je *simplified_line.json*, ktorého názov je možné zmeniť na riadku 112. Program ukladá súradnice zachovaných bodov v ceste:

["features"]["geometry"][0];

výstupný súbor teda obsahuje len usporiadané dvojice súradníc bodov.

Druhým výstupom je okno s dvoma grafmi; prvý graf znázorňuje tvar a počet uzlov pôvodnej krivky a druhý graf znázorňuje tvar a počet uzlov zjednodušenej krivky. Tento výstup slúži ako ukážka spracovania algoritmom.

Dokumentácia

Požiadavky

Je potrebné mať na zariadení nainštalované knižnice *numpy* a *matplotlib*.

Metódy

Program využíva 5 metód.

distance(x1, y1, x2, y2) – Metóda, ktorá vypočíta vzdialenosť medzi dvoma bodmi danými súradnicami *[x1, y1] a [x2, y2]* aplikovaním Pytagorovej vety.

calculate_area(x1, y1, x2, y2, x3, y3) – Metóda, ktorá vypočíta obsah trojuholníka, ktorý je definovaný súradnicami troch bodov pomocou Herónovho vzorca. Používa metódu *distance*.

triangle_height(base, area) – Metóda, ktorá vypočíta výšku trojuholníka definovaného v metóde *calculate_area*, ide o spôsob určenia vzdialenosti bodu od spojnice prvého a posledného bodu krivky.

douglas_peucker(points, epsilon) – Metóda, ktorá obsahuje implementáciu algoritmu Douglas-Peucker, ktorý funguje na princípe popísanom vyššie. Požaduje parametre *points* – vstupné súradnice bodov definujúce krivku, a *epsilon*, ktorého hodnotu je možné ľubovoľne nastaviť na riadku 135.

create_figure(graph1_x, graph1_y, graph2_x,graph2_y) – Metóda, ktorá vykreslí grafy znázorňujúce priebehy krivky pred a po zjednodušení.

Priebeh

Po spustení programu v rovnakom adresári, v akom je vstupná krivka, sa prevedie zjednodušenie tejto krivky. Program skontroluje, či je vstup validný a či obsahuje potrebné parametre, následne načíta hodnotu ε, s ktorou pracuje v metóde *douglas_peucker*. Pôvodnú krivku uloží do premennej *points*. Algoritmus pracuje na rekurziách; postupne vyhodnocuje, ktoré body zachová a ukladá ich do zoznamu *vertices_dump*, ktorý sa po ukončení algoritmu uloží do novej premennej *result*. V metóde *create_figure* sa vizualizuje výstup algoritmu a porovná sa s pôvodným tvarom krivky.

Námety na vylepšenie

Program by bolo potrebné vylepšiť o zjednodušenie prístupu používateľom, ktorý musí vlastné nastavenia (napríklad hodnoty ε, špecifikovania cesty k súradniciam či názov súboru) previesť vo vnútri programu. Podobne sa žiada vyriešiť nastavenie umiestnenia súradníc, ktoré môžu byť v rôznych vstupoch umiestnené inak; bolo by vhodné, aby program sám vedel dvojice súradníc nájsť a pracovať s nimi bez ohľadu na štruktúru vstupného súboru. V neposlednom rade by bolo vhodné, aby bolo možné nastaviť formát výstupu, ktorý je preddefinovane ukladaný vo formáte *JSON*, ako aj možnosť jednoduchšie nastaviť vnútornú štruktúru výstupného súboru.

Zoznam použitých zdrojov

[1] FLEISCHMANN, M. (2021): Line simplification algorithms. *https://martinfleischmann.net/line-simplification-algorithms/* (6. 2. 2022)

[2] LEE, S. (2021): Simplify Polylines with the Douglas Peucker Algorithm. *https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1* (9. 2. 2022)

[3] Python Software Foundation: Python 3.10.2 documentation. *https://docs.python.org/3/* (9. 1. 2022)