Umelá inteligencia: Zadanie II.

Klastrovanie

Ondrej Krajčovič

Zadanie:

Vašou úlohou je naprogramovať zhlukovač pre 2D priestor, ktorý zanalyzuje 2D priestor so všetkými jeho bodmi a rozdelí tento priestor na k zhlukov (klastrov). Implementujte rôzne verzie zhlukovača, konkrétne týmito algoritmami:

k-means, kde stred je centroid

k-means, kde stred je medoid

divízne zhlukovanie, kde stred je centroid

Vyhodnocujte úspešnosť/chybovosť vášho zhlukovača. Za úspešný zhlukovač považujeme taký, v ktorom **žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500.**

Vizualizácia: pre každý z týchto experimentov vykreslite výslednú 2D plochu tak, že označkujete (napr. vyfarbíte, očíslujete, zakrúžkujete) výsledné klastre.

Ondrej Krajčovič

Všeobecné informácie:

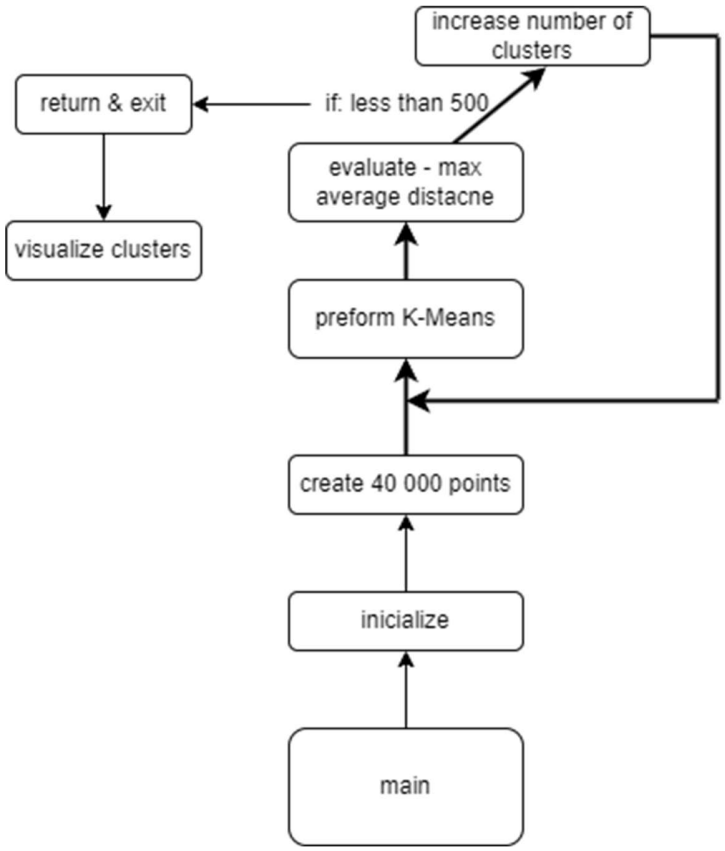
Moje riešenie pozostáva z štyroch python súboroch typu *.py*:

- *UI02_centroid.py*
- *UI02_medoid.py*
- *UI02_divisive.py*

Každý program obsahuje generovanie bodov, následné pomocné funkcie a funkciu main z ktorej sa pomocné funkcie volajú, a následne dáta, ktoré sa z nich vracajú sú vizualizované

- *random*
- *time*
- *torch (optimalizácia)*
- *tkinter (vizualizácia)*
- *numpy (pomocná knižnica na výpočty)*
- *warnings (ignorácia warningu z numpy knižnice)*

Torch: knižnica pomocou ktorej optimalizujem dĺžku času, počas ktorého program beží. Funguje tak, ž e mieto počítania zložitých operácií (napríklad výpočet euklidovskýc vzdialeností) prebieha na GPU (graphics processing unit) miesto na CPU (procesor). Nakoľko na storji na ktorom so tento projekt vytváral, mám 8 GB VRAM(pamäť grafickej jednotky), priebeh program to značne zrýchlilo. Okrem toho som na zoznamy použil knižnicu Numpy, ktorá ich spracuváva oveľa rýchelišie



Obr.1: všeobecná schéma riešenia (platen pre všetky tri implementácie)

Ondrej Krajčovič

I. K-Means kde centrom je Centroid:

Pseudokód:

Náhodne vyber K bodov ako počiatočné centroidy(centroid je definovaný ako euklidovský stred 2D hromady bodov)

Priraďovanie bodov k centroidom:

Slučka prechádzajúca každý bod:

Vypočítaj vzdialenosť bodu ku každému centroidu

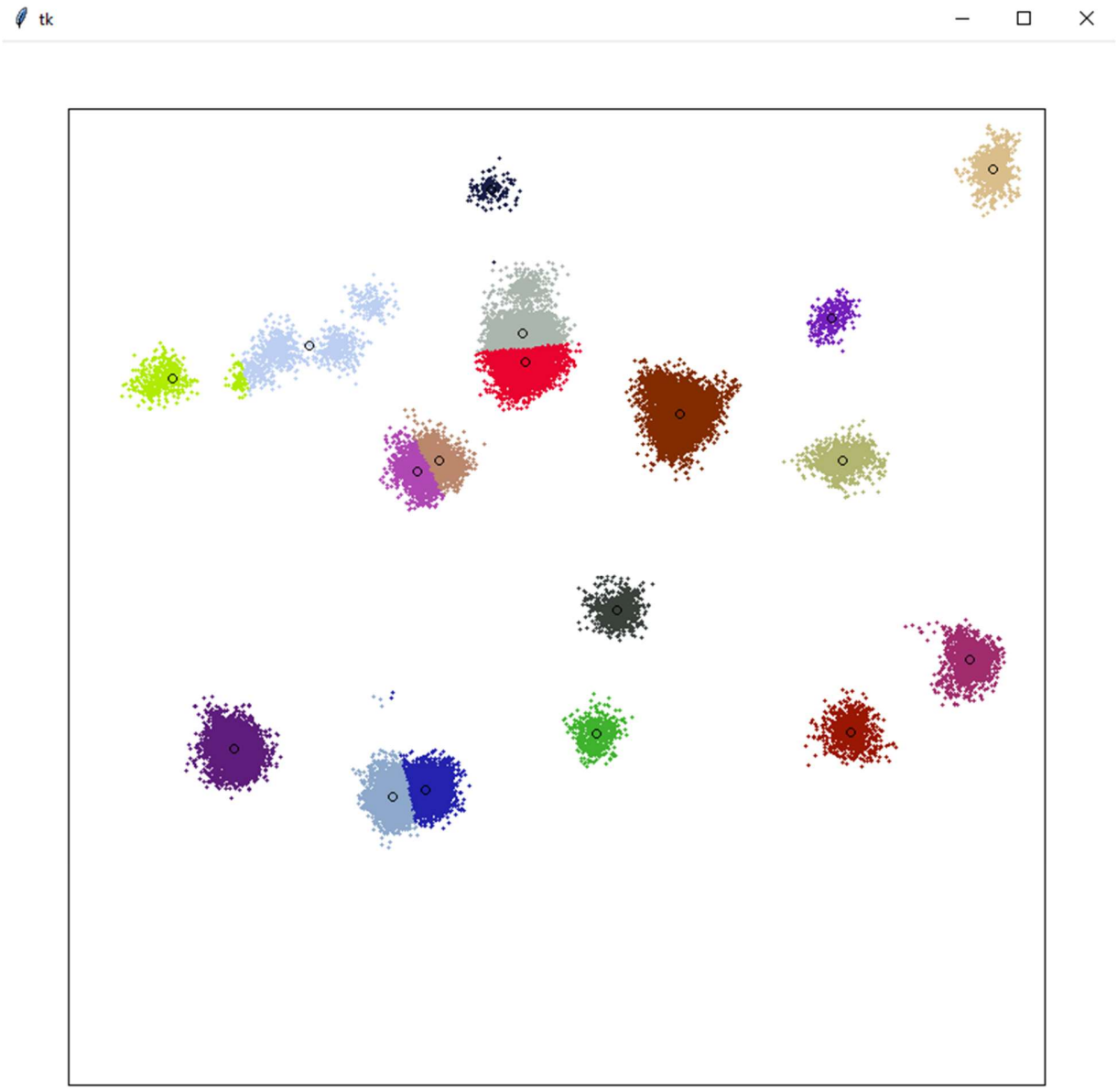
Priraď bod k najbližšiemu centroidu

Aktualizácia centroidov:

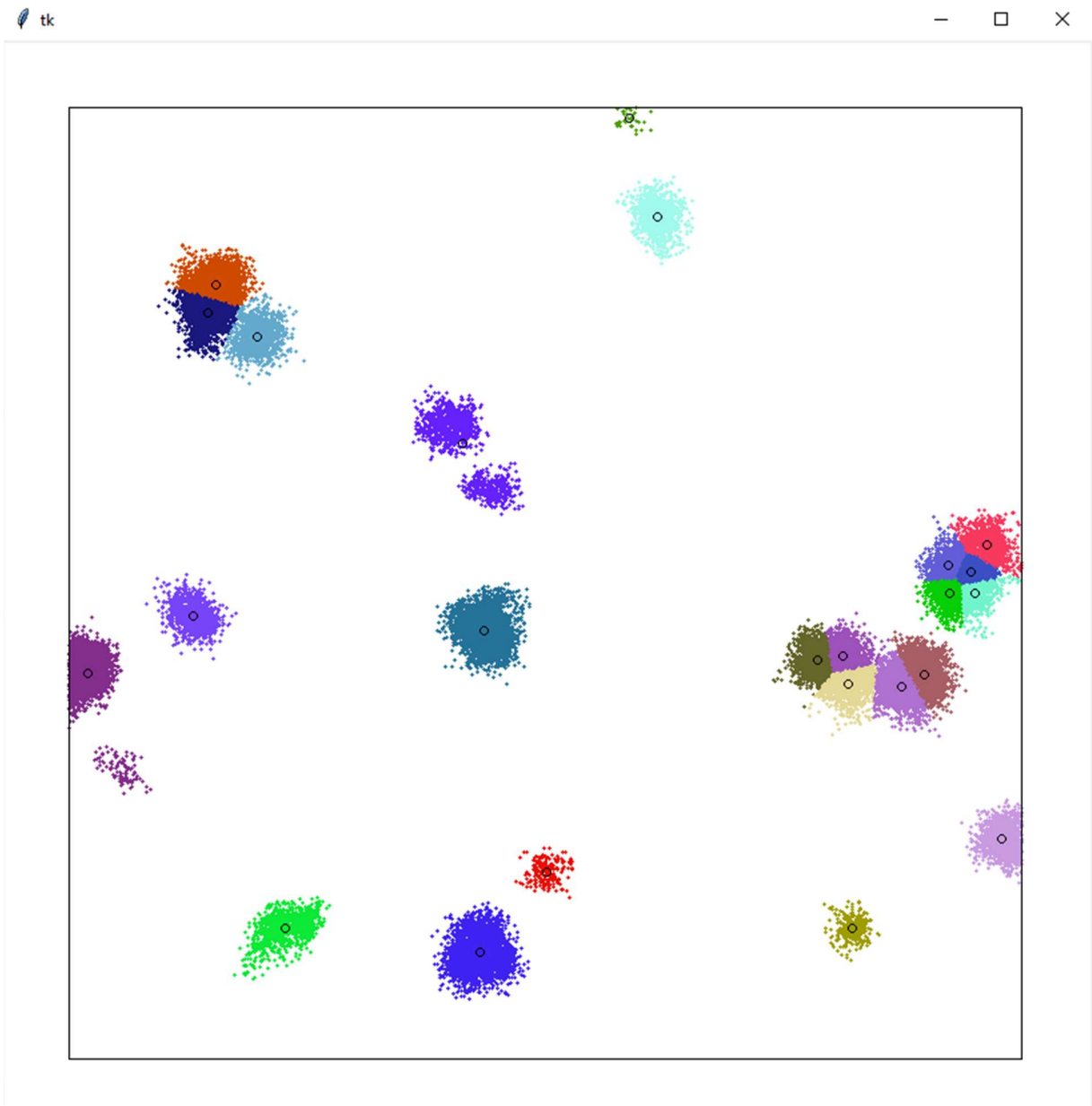
Slučka prechádzajúca každý klaster:

Vypočítaj nový centroid ako priemer všetkých bodov v klastri

V prípade že sa výsledky opakujú ukonči program



Obr.2: Ukážka jedného z lepších výsledkov po riešení problému algorimom K-Means kde centrom boli centroidy



Obr.3: Ukážka jedného z horších výsledkov po riešení problému algorimom K-Means kde centrom boli centroidy

Ondrej Krajčovič

II. K-Means kde centrom je Medoid:

Pseudokód:

Náhodne vyber K bodov ako počiatočné centroidy(centroid je definovaný ako bod najbližší k euklidovskému stredu 2D hromady bodov)

Priraďovanie bodov k centroidom:

Slučka prechádzajúca každý bod:

Vypočítaj vzdialenosť bodu ku každému centroidu

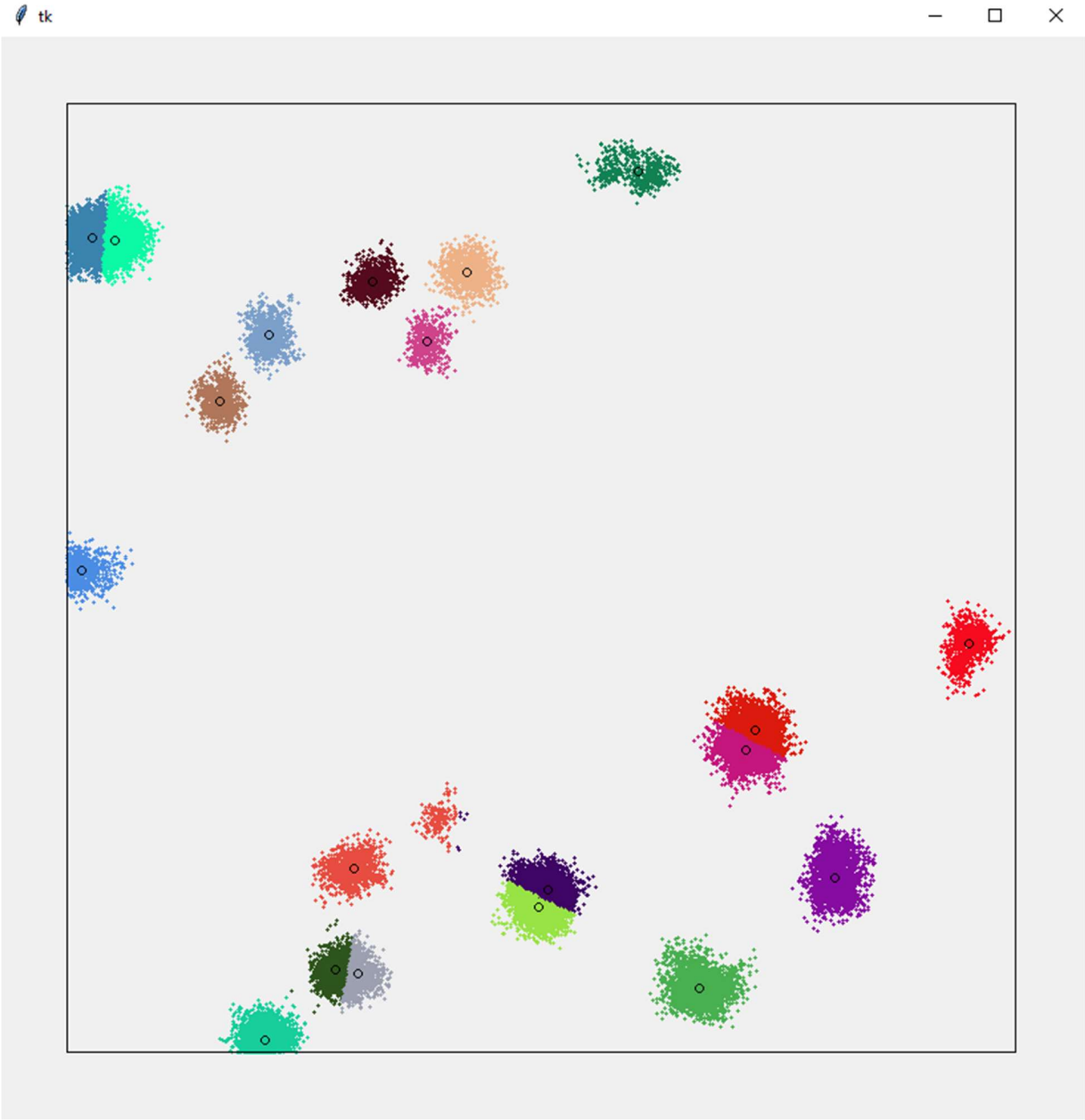
Priraď bod k najbližšiemu centroidu

Aktualizácia centroidov:

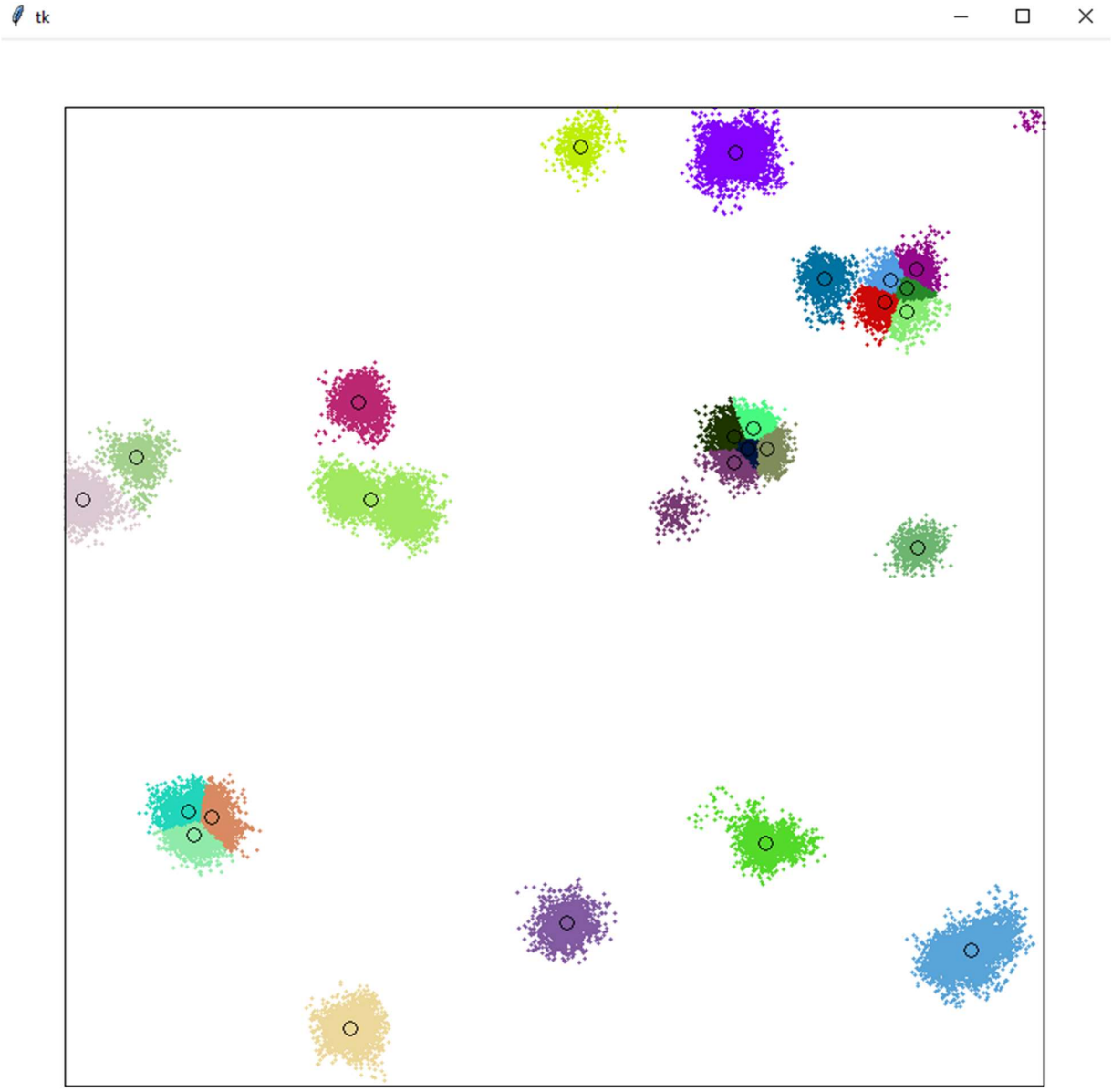
Slučka prechádzajúca každý klaster:

Vypočítaj nový centroid ako priemer všetkých bodov v klastri

V prípade že sa výsledky opakujú ukonči program



Obr.4: Ukážka jedného z lepších výsledkov po riešení problému algorimom K-Means kde centrom boli medoidy



Obr.5: Ukážka jedného z horších výsledkov po riešení problému algorimom K-Means kde centrom boli medoidy

III. Divízne K-Means

Opis algoritmu:

Všetky body inicializuj do jedného klastra

postupné delenie klastra:

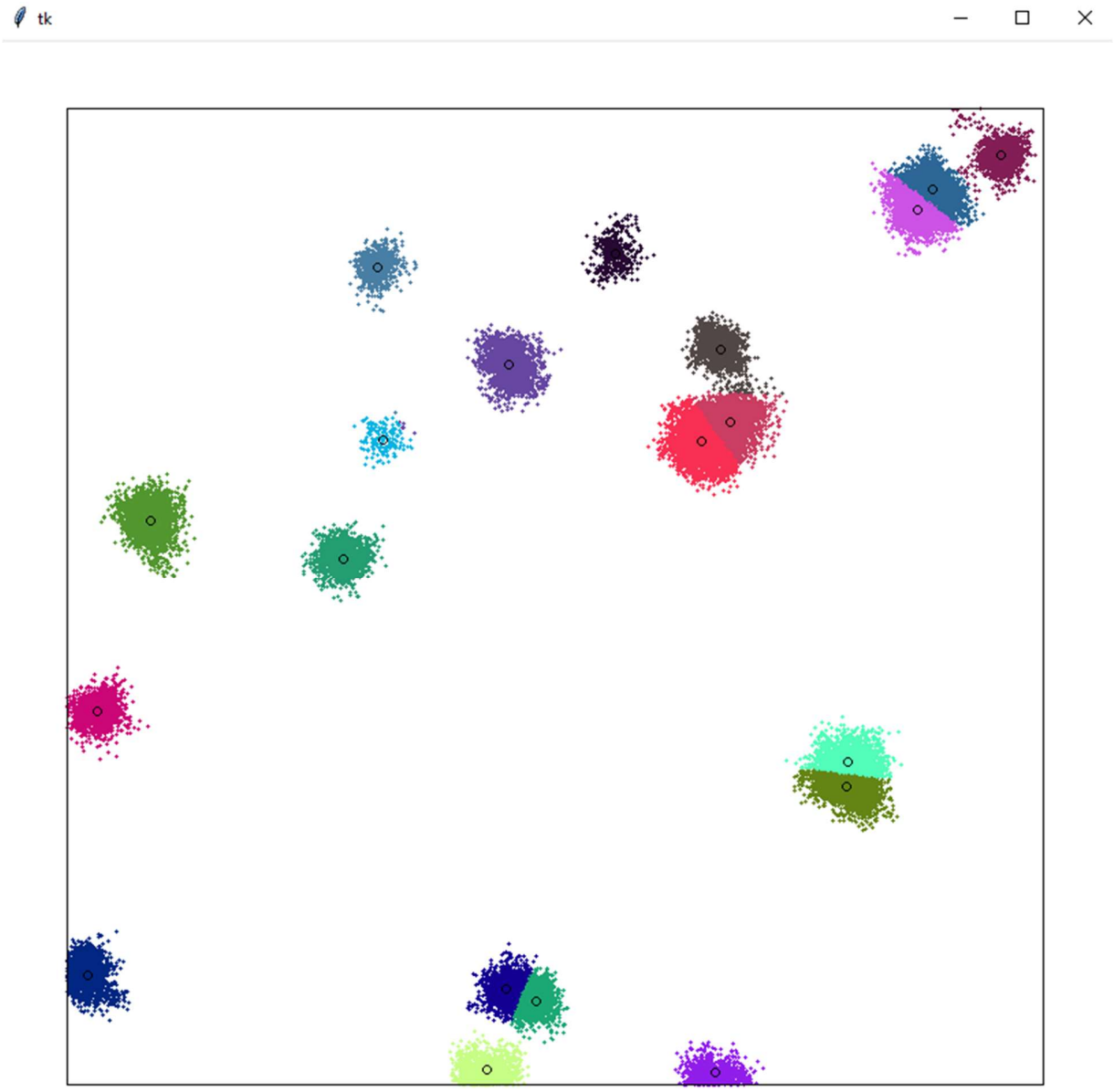
Vyber najrozľahlejší klaster

Vyber dva náhodné body v tomto klastri ako počiatočné centroidy

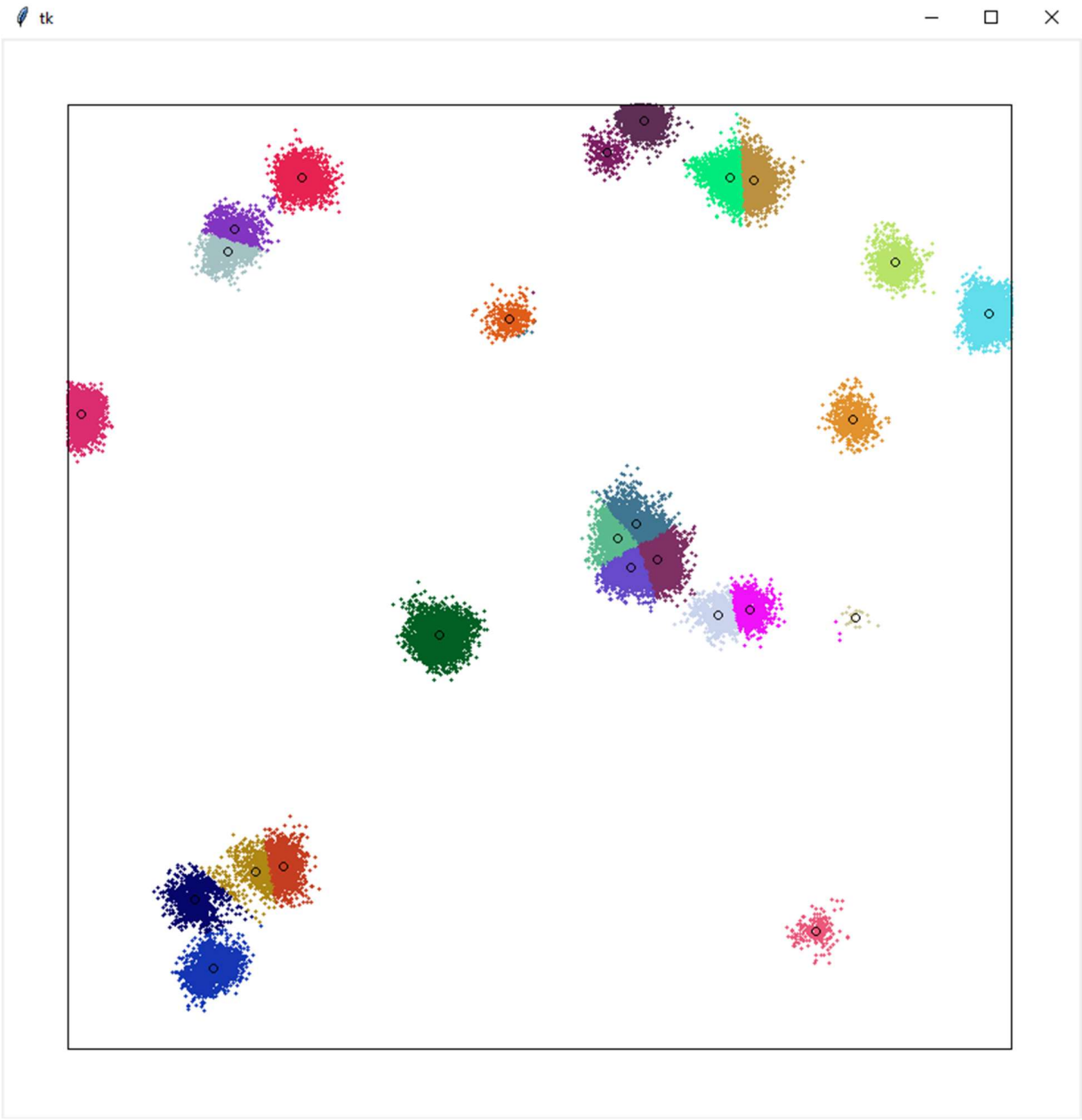
Priraďovanie a aktualizácia:

na dvoch pod-klusteroch vykonaj k-means algoritmus

Opakuj delenie, kým nedosiahneš k(25) klastrov



Obr.6: Ukážka jedného z horších výsledkov po riešení problému algorimom divíznym K-Means



Obr.7: Ukážka jedného z horších výsledkov po riešení problému algorimom divíznym K-Means

IV. Záver:

Pomocným programom som 1000-krát spustil main.py, to jest oba algoritmy a porovnával som dĺžku cesty nájdenej jednotlivými algoritmami a čas za ktorý danú cestu našli.
Výsledkom bolo nasledovné:

Približný priemer času riešenia behu programu bol pri riešení klastrovania centroidmi bol ~30s

Približný priemer času riešenia behu programu bol pri riešení klastrovania centroidmi bol ~200s

Približný priemer času riešenia behu programu bol pri riešení klastrovania centroidmi bol ~0.5s

Najefektívnejšie teda v mojom prípade bolo riešenie pomocov Divízneho K-Means algoritmu

Zďaleka najneefektívnejší bol K-Means kde centrom boli medoidy. Výsledky neboli vyzuálne lahodiace, program beží často na viacero iterácií, a celkový čas riešenia nie je absolútne uspokujúci/potešujúci

V prípade že programy generovali príliš veľké množstvo Klastrov, nechal som ich bežať pre optimálnejší výsledok