Fakulta informatiky a informačných technológií
STU v Bratislave

Umelá inteligencia

Zhlukovanie

Zadanie 4

Zsolt Kiss

Utorok 14:00-15:50
Ing. Ivan Kapustík

2020/2021

Obsah

K-Means

Implementácia, vysvetlenie zdrojového kódu

public void run()

- Pomocou iterátora prechádzam zoznam bodov, vyberiem prvých k bodov, a pre každý z týchto bodov vytvorím zoznam, čo bude reprezentovať jeden klaster. Napríklad keď k = 20, tak budem mať 20 zoznamov na začiatku. Tieto zoznamy následne vložím do hash mapy. Po vytvorení k klastrov spracúvam ďalšie body tak, že postupne prechádzam všetky klastry a hľadám klaster s najnižšou vzdialenosťou od aktuálneho bodu. Bod potom vložím do tejto klastry a aktualizujem centroid.

private void initializeCluster(**int** clusterNumber, DataPoint dataPoint)

- Táto funkcia slúži na inicializáciu klastra. Vytvoríme objekt Cluster (ktorý je centroid) a zoznam klastrov. Do zoznamu vložíme dataPoint ktorú dostane funkcia ako parameter. Nakoniec vložíme zoznam na určité miesto do hash mapy.

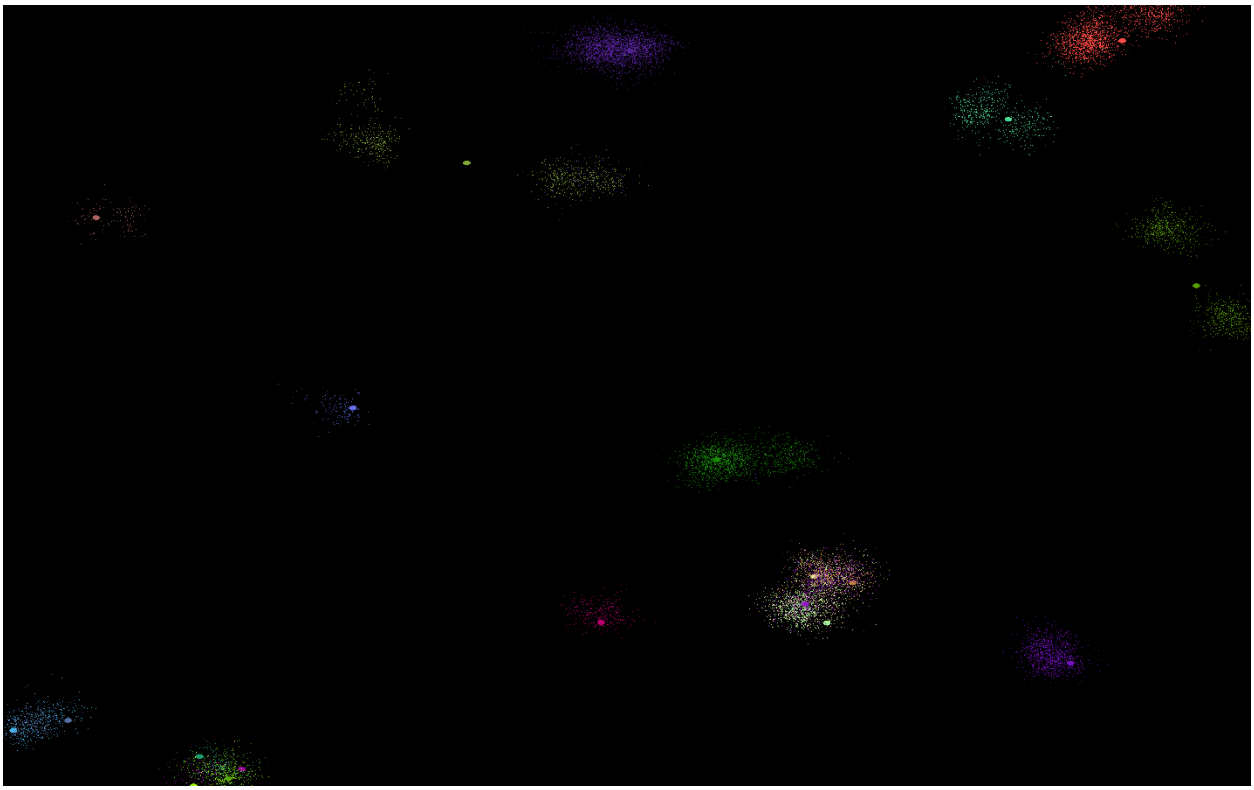
Výhody

- Jednoduché na pochopenie a implementáciu
- Rýchle a konvergentné v konečnom počte krokov

Nevýhody

- Rôzne počiatočné zoskupenia môžu viesť k rôznym finálnym zoskupeniam, preto je vhodné spustiť postup niekoľkokrát s rôznymi (náhodnými) počiatočnými zoskupeniami.
- Výsledné zoskupenie závisí od jednotiek merania. Ak premenné majú rôznu povahu alebo sa veľmi líšia vzhľadom na ich veľkosť, potom je vhodné ich štandardizovať.

Ukážka



K-Medoids

Implementácia, vysvetlenie zdrojového kódu

private void extractRandomMedoids (List<DataPoint> dataSet, List<Medoid> medoids)

- Náhodne vyberáme k medoidov

private void calculateCost(List<DataPoint> dataPoints, List<Medoid> medoids)

- Od každého bodu ku každému vypočítame vzdialenosť. Tieto vzidalenosti spočítame, a dostaneme cenu vytvorenia medoidu. Pre daný medoid nastavíme cenu vytvorenia.

public void run()

- Na začiatku spustíme hore uvedené funkcie, a potom nasleduje cyklus, ktorý zastavíme iba vtedy, ak po prechádzaní zoznamu medoidov nájdeme menšiu alebo rovnakú cenu ako predchádzajúca cena.

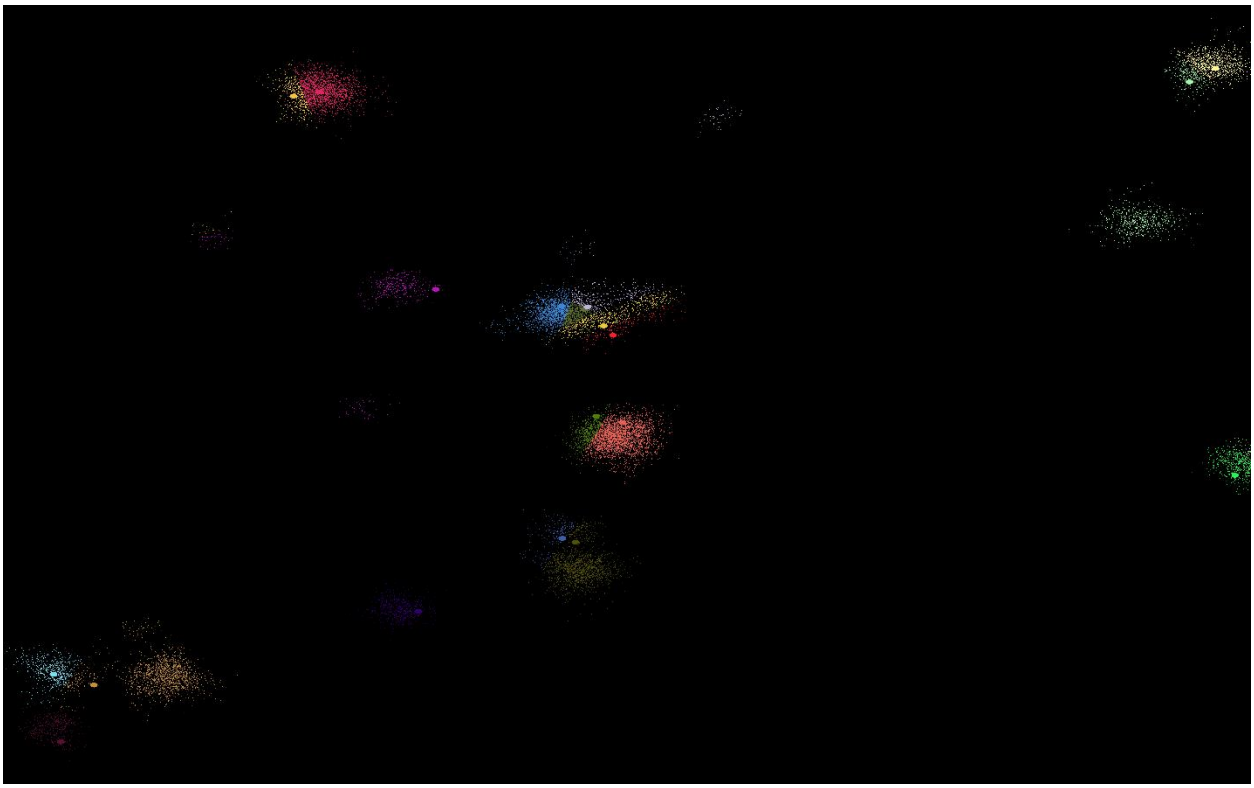
Výhody

- Jednoduché na pochopenie a implementáciu
- Rýchle a konvergentné v konečnom počte krokov
- Zvyčajne menej citlivý na odľahlé hodnoty ako k-means
- Umožňuje použitie všeobecných odlišností predmetov

Nevýhody

- Rôzne počiatočné sady medoidov môžu viesť k rôznym konečným zoskupeniam, preto sa odporúča postup vykonať niekoľkokrát s rôznymi počiatočnými sadami medoidy.
- Výsledné zoskupenie závisí od jednotiek merania. Ak premenné majú rôznu povahu alebo sú veľmi odlišné vzhľadom na ich premennú veľkosť, potom je vhodné ich štandardizovať.

Ukážka



Aglomeratívne zhlukovanie

Implementácia, vysvetlenie zdrojového kódu

public void run ()

- Na začiatku inicializujeme všetky body tak, aby každý bod bol aj samostatným klastrom, inými slovami, aby každý bod patril do iného klastra. Postupne prechádzame všetky body, hľadáme body, ktoré spĺňajú podmienku, že vzdialenosť je menšia, ako definovaná maximálna vzdialenosť. Ak spĺňajú podmienku, nastavíme preňho rovnaký clusterNumber, a označíme ho aby sme ho nemohli vložiť aj do iných klastier. Aktualizujeme centroid, a ďalej už porovnávame body ku aktualizovanému centroidu. Keďže prechádzame body postupne, môže sa stať, že nenájdeme také body, ktorých sme ešte nezariadili do niektorého klastra, a splnia podmienku, že vzdialenosť je menšia, ako definovná maximálna vzdialenosť. V tomto prípade vytvoríme nový centroid.

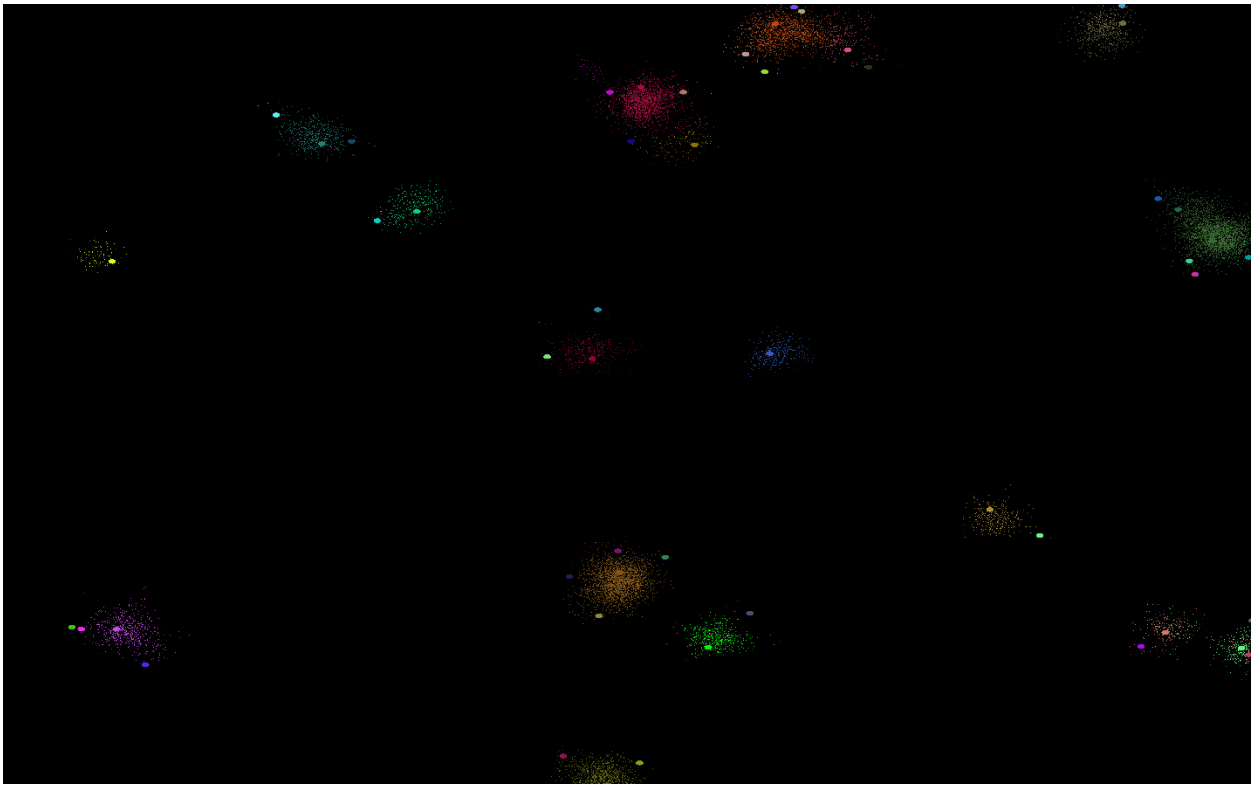
Výhody

- Pre väčšinu metód stačí mať maticu odlišnosti D medzi objektmi:
- Nevyžaduje znalosť počtu klastrov.

Nevýhody

- Závisí to od rozsahu údajov
- Výpočtovo zložité pre veľké súbory údajov

Ukážka



Divizívne zhlukovanie

Implementácia, vysvetlenie zdrojového kódu

public void run ()

- Na začiatku inicializujeme všetky body tak, aby každý bod patril do toho istého klastra. Postupne prechádzame všetky body, hľadáme body, ktoré spĺňajú podmienku, že vzdialenosť je menšia, ako definovaná maximálna vzdialenosť. Ak spĺňajú podmienku, nastavíme preňho rovnaký clusterNumber, a označíme ho aby sme ho nemohli vložiť aj do iných klastier. Aktualizujeme centroid, a ďalej už porovnávame body ku aktualizovanému centroidu. Keďže prechádzame body postupne, môže sa stať, že nenájdeme také body, ktorých sme ešte nezariadili do niektorého klastra, a splnia podmienku, že vzdialenosť je menšia, ako definovná maximálna vzdialenosť. V tomto prípade vytvoríme nový centroid.

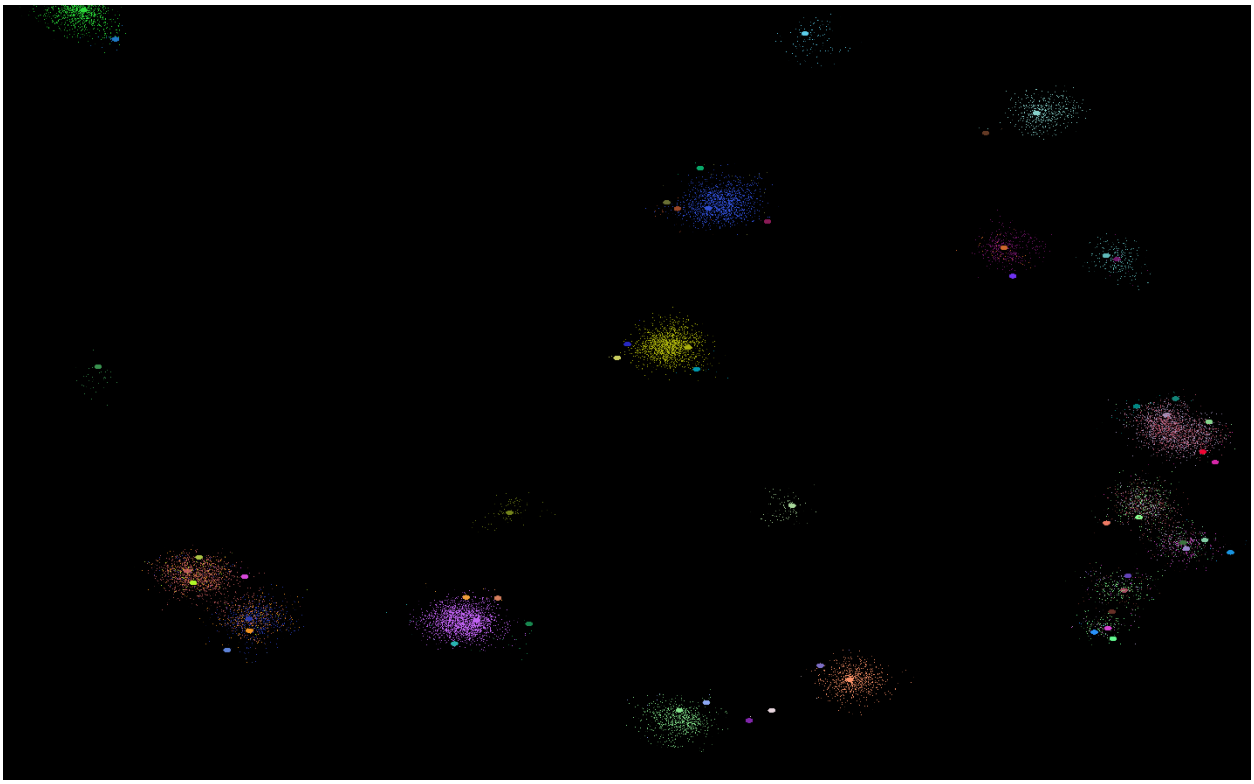
Výhody

- Pre väčšinu metód stačí mať maticu odlišnosti D medzi objektmi:
- Nevyžaduje znalosť počtu klastrov.

Nevýhody

- Závisí to od rozsahu údajov
- Výpočtovo zložité pre veľké súbory údajov

Ukážka



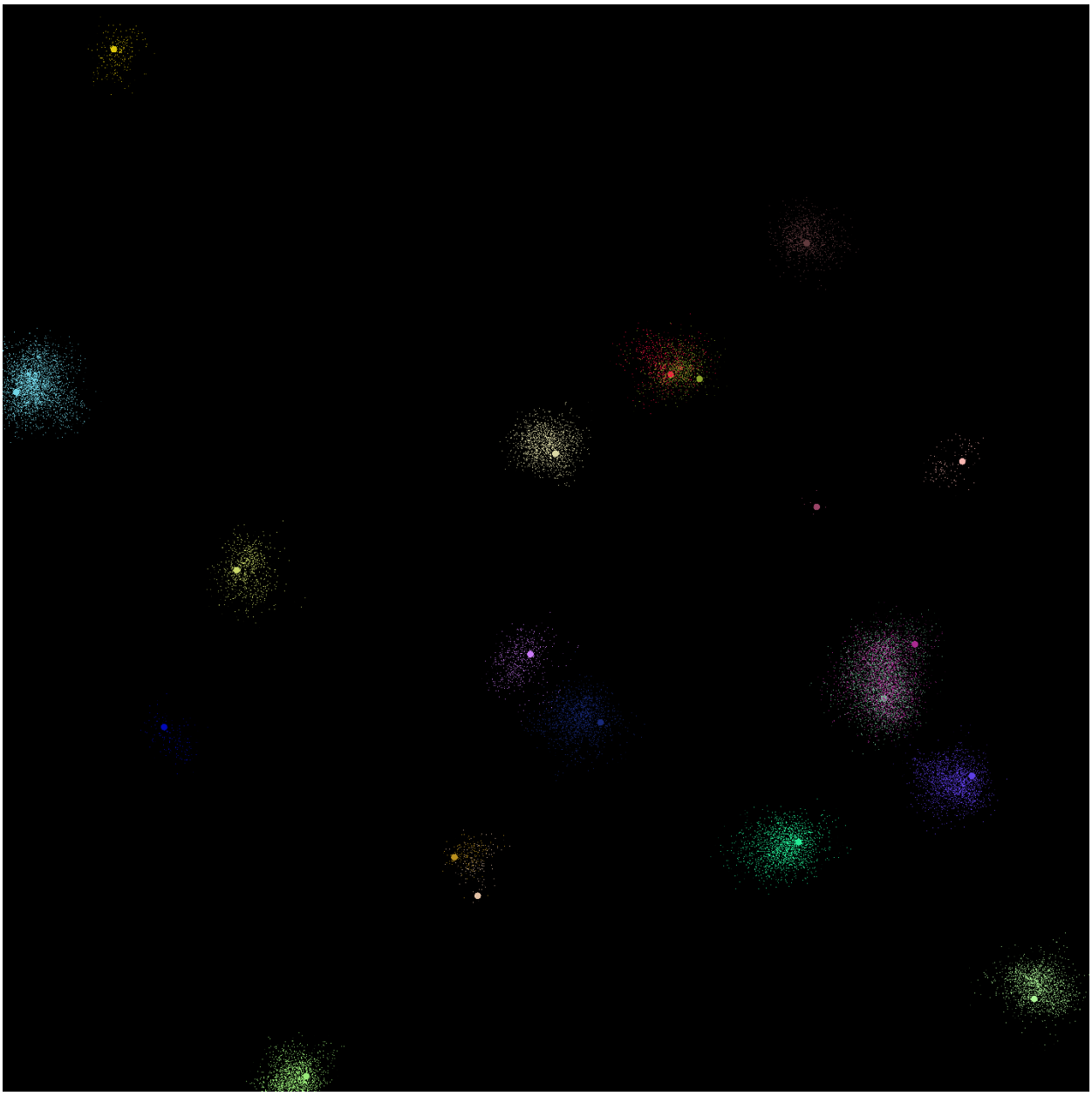
Testovanie

POZNÁMKA:
Na dole uvedených obrázkach môžeme vidieť, že výsledky pre aglomeratívne a divizívne zhlukovanie vyzerajú tak isto. Dôvodom je to, že jediná vec, ktorá je odlišná medzi dvoma kódmi je časť na začiatku, kde nastavím pre všetky body klastry, do ktorého patria. Pri aglomeratívnom zhlukovaní všetky body majú samostatný klaster, pri divizívnom klastrovaní všetky body patria do toho istého klastra (v tomto prípade do 0-teho). Preto som to robil tak, lebo sa mi nepodarilo naprogramovať algoritmus pre aglomeratívne zhlukovanie kvôli nedostatku pamäti v programovacom jazyku Java. Za to sa ospravedlňujem, ale dúfam, že môj projekt spĺňa aspoň minimálne požiadavky. Napíšem ale postup, ako by som to spravil:

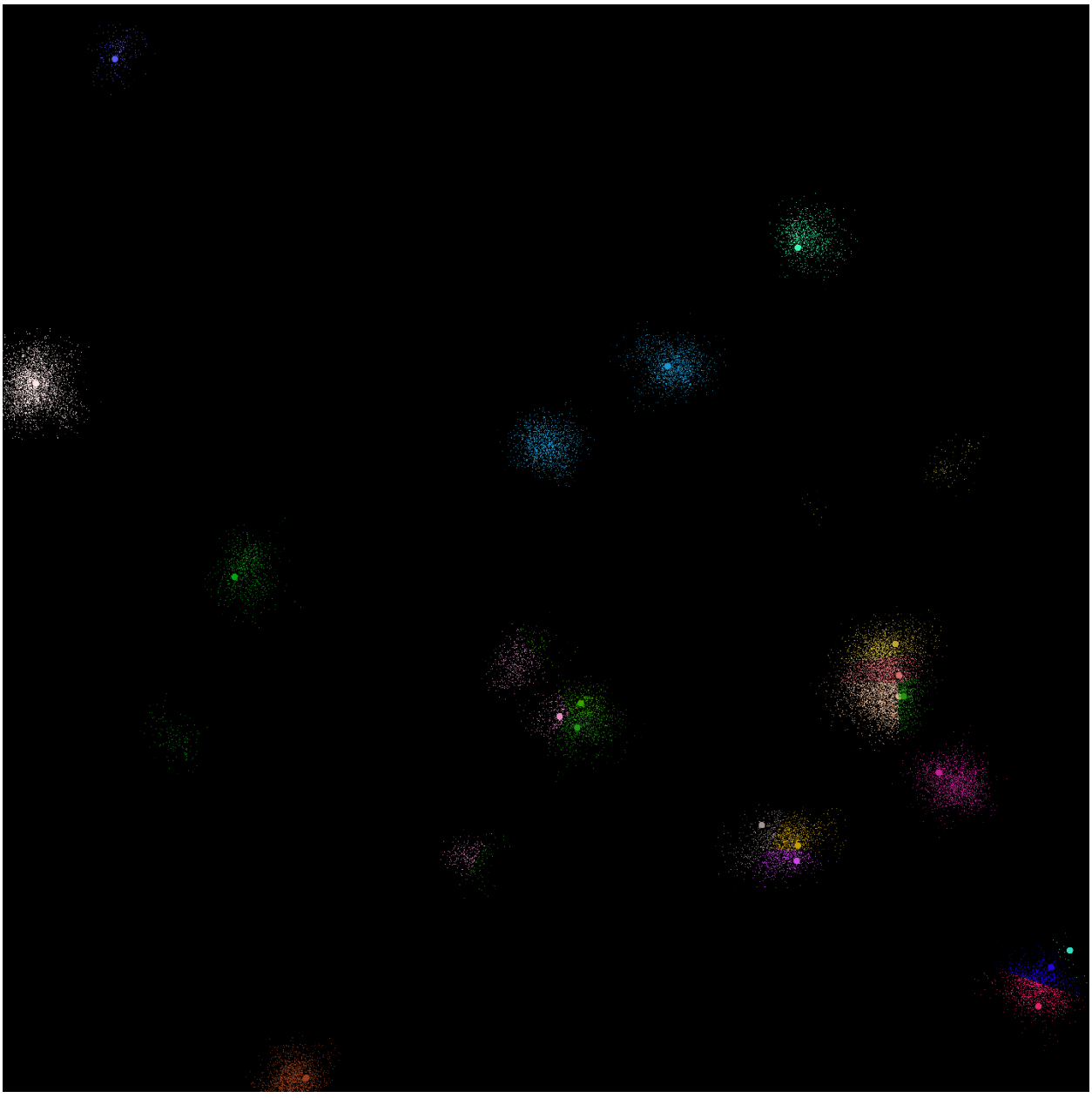
1. Pre všetky body vytvorím samostatný zoznam, ktorý na začiatku bude obsahovať 2 body, pôvodný bod a centroid, ktorý má na začiatku rovnaké súradnice ako pôvodný bod, líši sa iba v jednom atribúte, a to by bolo označenie, že tento bod je centroid. Tieto zoznamy spolu s bodmi následne vložím do hash mapy.
2. Vo vnorenom cykle prechádzam hash mapu postupne, a v zoznamoch hľadám bod, ktorý je centroid. V druhom cykle tak isto prechádzam hash mapu, zoznamy a hľadám centroid.
3. Nájdem centroid s nižšou vzdialenosťou ako 500 od aktuálneho centroidu, a celý zoznam prekopírujem do aktuálneho zoznamu. Druhý zoznam následne vymažem.
4. Aktualizujem súradnice centroidu a následne v dalšom kroku cykle budem porovnávať už k tomuto aktualizovanému bodu.

Pri testovaní som spúšťal všetky 4 algoritmy na tie isté body, aby som ich vedel porovnávať. Počet bodov bol vždy 40020, k bol nastavený na 20. V zadaní je uvedené, že zhlukovač je úspešný vtedy, keď vzdialenosť medzi stredom klastra a bodom patriacim k danému klastru je menšia ako 500. Keďže pri aglomeratívnom a divizívnom zhlukovaní nie je možné zadať k, preto som nastavil maximálnu vzdialenosť na 500. Aj napriek tomu môže nastať situácia, kedy táto vzdialenosť je väčšia, lebo po každom vložení bodu do klastra aktualizujeme súradnice centroida. Po vykreslení bodov som nastavil podmienku, že ak aspoň jeden takýto prípad nastane, považujem zhlukovač ako neúspešný.

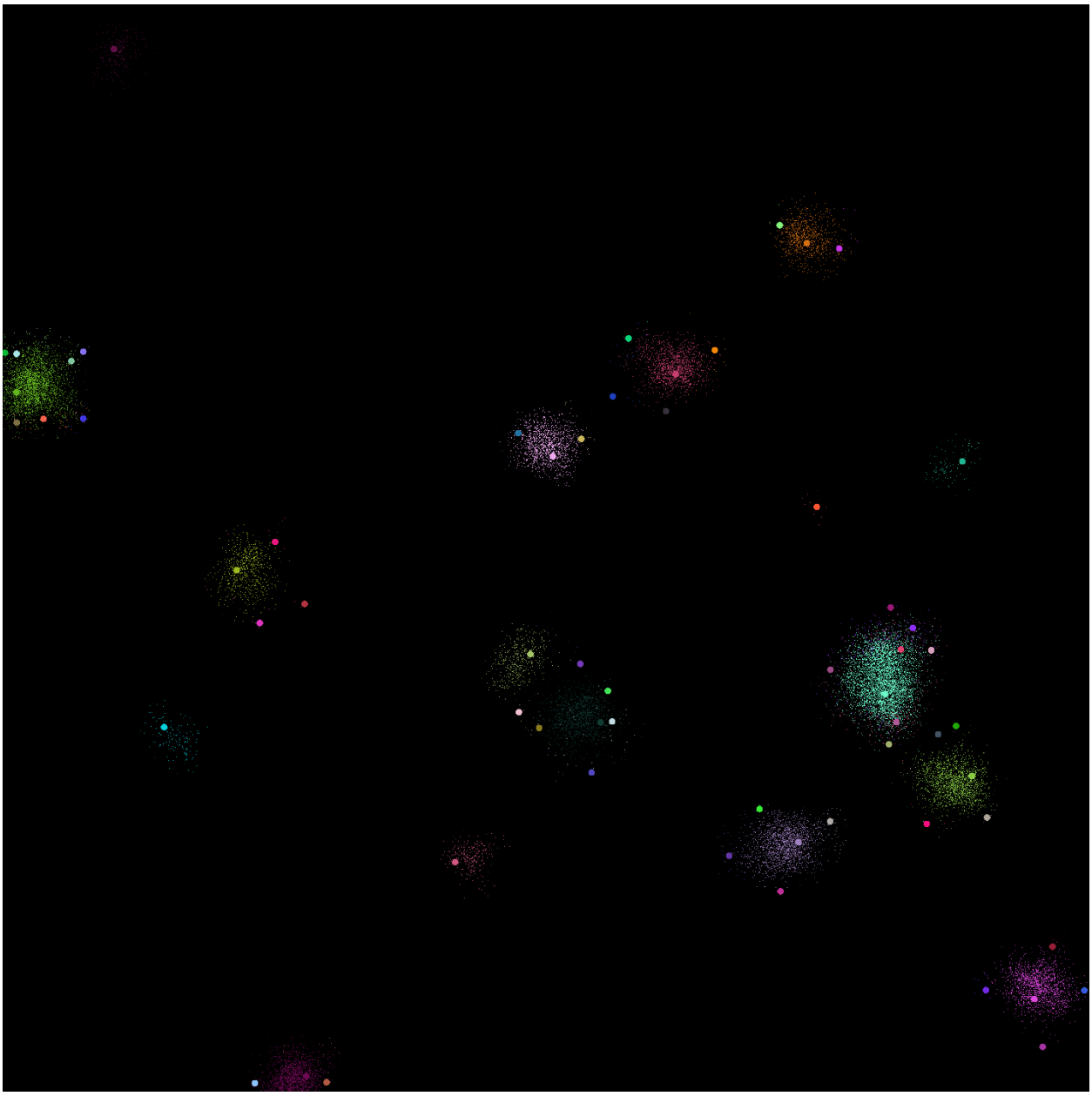
K-Means, k = 20



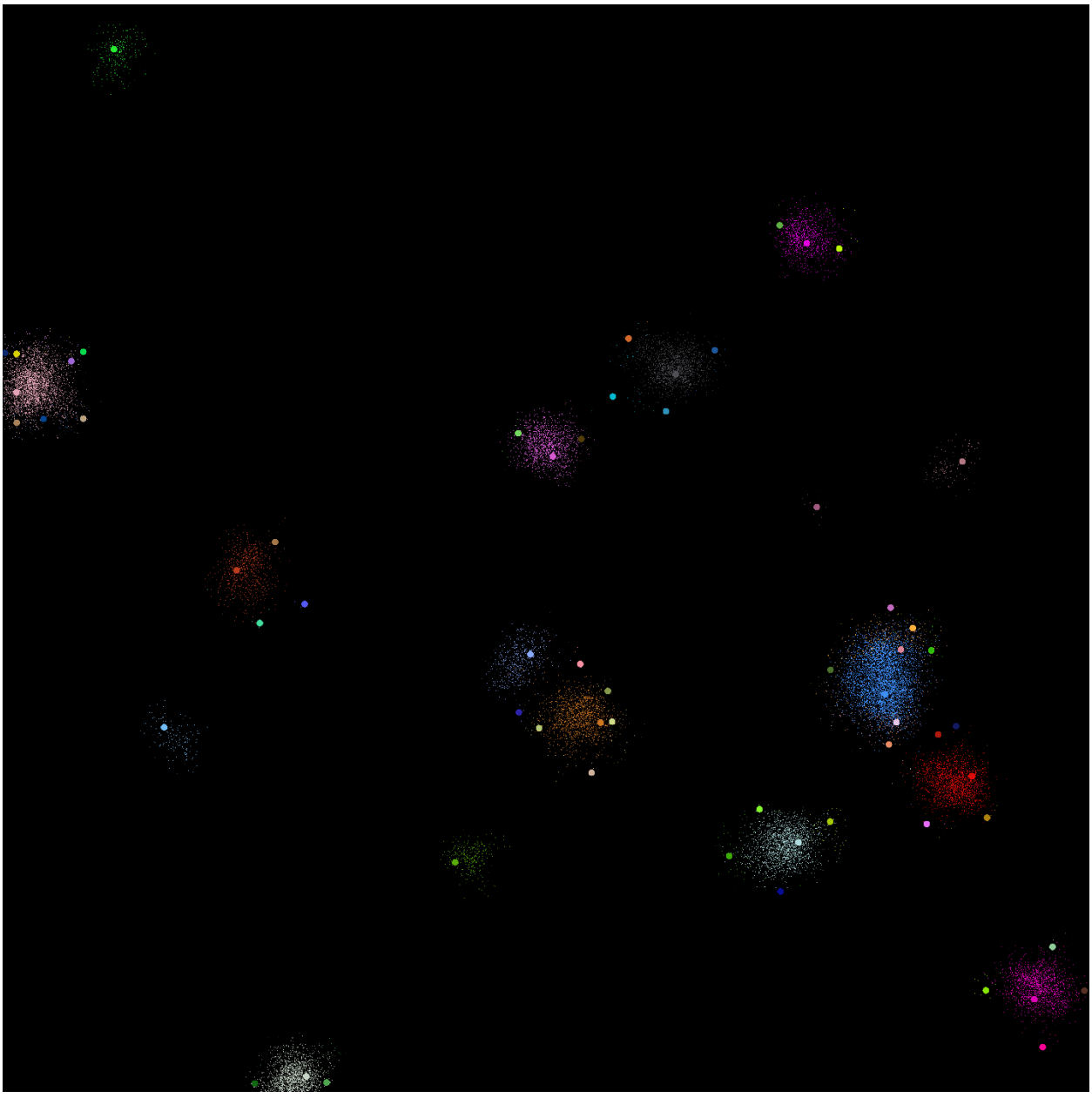
K-Medoids, k = 20



Aglomeratívne zhlukovanie, maximálny offset = 500



Divizívne zhlukovanie, maximálny offset = 500



Výstup

```
Vyberte jeden z moznosti:
1 - K Means zhlukovanie
2 - K Medoids zhlukovanie
3 - Aglomerativne zhlukovanie
4 - Divizivne zhlukovanie
5 - Spustit vsetky algoritmy
5
K Means zhlukovanie
Algoritmus prebehlo.
Vykreslenie ...
Aspon jeden bod mal vacsiu vzdialenost od centroidu ako 500-> Neuspesne zhlukovanie
Pocet takych bodov: 3499
Beh programu: 2947 ms

K Medoids zhlukovanie
Algoritmus prebehlo.
Vykreslenie ...
Aspon jeden bod mal vacsiu vzdialenost od centroidu ako 500-> Neuspesne zhlukovanie
Pocet takych bodov: 4601
Beh programu: 2762 ms

Aglomerativne zhlukovanie
Algoritmus prebehlo.
Vykreslenie ...
Aspon jeden bod mal vacsiu vzdialenost od centroidu ako 500 -> Neuspesne zhlukovanie
Pocet takych bodov: 1616
Beh programu: 2854 ms

Divizivne zhlukovanie
Algoritmus prebehlo.
Vykreslenie ...
Aspon jeden bod mal vacsiu vzdialenost od centroidu ako 500-> Neuspesne zhlukovanie
Pocet takych bodov: 1616
Beh programu: 2819 ms


Process finished with exit code 0
```