

## Zadanie 3

# Aglomeratívne zhlukovanie s využitím centroidu a medoidu

Umelá inteligencia

Frederik Duvač

3. semester

1.12. 2023

# Obsah

<b>1. Teoretické východiská</b>	<b>2</b>
1.1. Technické pozadie	2
1.2. Zadanie	2
1.3. Aglomeratívne zhlukovanie	2
1.4. Bod zhlukovania	3
1.4.1. Centroid	3
1.4.2. Medoid	3
<b>2. Implementácia</b>	<b>3</b>
2.1. Reprezentácia údajov	3
2.2. Aglomeratívne zhlukovanie	5
2.3. Používateľské rozhranie	7
2.4. Vizualizácia	8
<b>3. Testovanie</b>	<b>9</b>
<b>4. Zhodnotenie</b>	<b>12</b>

# 1. Teoretické východiská

## 1.1. Technické pozadie

Hardvér, na ktorom bol testovaný program, je notebook Lenovo Legion S7, 12. generácia Intel® Core™ i5-12500H, 16 GB RAM. Celý projekt je implementovaný v programovacom jazyku Python 3. Pre zrýchlenie výpočtov bol použitý interpreter PyPy vo verzí 3.12.

## 1.2. Zadanie

Majme 2D priestor veľkosti -5000 až +5000, do tohto priestoru sa na úvod vygeneruje 20 bodov. Následne sa vygeneruje 20000 ďalších bodov podľa nasledujúcich pravidiel:

1. Náhodne vyberte jeden zo **všetkých doteraz vytvorených** bodov v 2D priestore. (nielen z prvých 20)  
Ak je bod príliš blízko okraju, tak zredukujete príslušný interval, uvedený v nasledujúcich dvoch krokoch.
2. Vygenerujte náhodné číslo X\_offset v intervale od -100 do +100
3. Vygenerujte náhodné číslo Y\_offset v intervale od -100 do +100
4. Pridajte nový bod do 2D priestoru, ktorý bude mať súradnice ako náhodne vybraný bod v kroku 1, pričom tieto súradnice budú posunuté o X\_offset a Y\_offset

Cieľom je spraviť aglomeratívny zhlukovač pre 2D priestor podľa centroidu a medoidu. Následne je potrebné vyhodnotiť jednotlivé klastre.

## 1.3. Aglomeratívne zhlukovanie

Algoritmus aglomeratívneho zhlukovania je používaný v oblasti analýzy dát a štatistiky na zoskupovanie podobných pozorovaní do zhlukov. Tento algoritmus začína s každým pozorovaním ako samostatným zhlukom a postupne ich spája na základe podobnosti, až kým nie je dosiahnutý požadovaný počet zhlukov.

Kroky algoritmu sú nasledovné:

1. **Incializácia:** Na začiatku algoritmu máme každý bod v dátach ako jeden zhluk.
2. **Matica vzdialenosí:** Vytvoríme spodnú trojuholníkovú maticu vzdialenosí, kde každý riadok a stĺpec predstavuje jeden zhluk s hodnotami reprezentujúcimi vzdialenosí medzi zhlukmi.
3. **Hľadanie najbližších zhlukov:** Opakovane hľadáme dva najbližšie zhluky v matici a zlučujeme ich do jedného zhluku.
4. **Aktualizácia matice:** Po zlúčení dvoch zhlukov aktualizujeme vzdialenosnú maticu tak, aby obsahovala nový zhluk a vzdialenosí od

neho k ostatným zhlukom. Týmto spôsobom sa veľkosť matice postupne znižuje, čo umožňuje efektívnejšie spracovanie.

5. **Kritérium ukončenia:** Algoritmus pokračuje, kým nedosiahne požadovaný počet zhlukov.

## 1.4. Bod zhlukovania

### 1.4.1. Centroid

Centroid zhluku je geometrické ľažisko všetkých bodov v zhluku. Je to bod, ktorý je určený ako priemer súradníc všetkých bodov v zhluku.

### 1.4.2. Medoid

Medoid je bod v zhluku, ktorý minimalizuje celkovú vzdialenosť od ostatných bodov v zhluku. Inak povedané, je to ten bod, ktorý má od ostatných bodov v zhluku najmenšiu súčtovú vzdialenosť.

## 2. Implementácia

### 2.1. Reprezentácia údajov

Program bol navrhnutý objektovo a teda, hlavná funkcia `main()` je zodpovedná len za používateľske rozhranie, kde sa nastavia hodnoty potrebné pre zbehnutie programu a nakoniec len zobrazí výsledok v GUI okne.

Jadro programu je trieda `Clustering`, ktorá obsahuje hlavnú metódu `agglomerative_clustering()`, `update_distance_matrix()`, pomocné funkcie na zhlukovanie a funkcie na generovanie bodov. Z výkonnostného dôvodu sú body reprezentované jednoduchou dátovou štruktúrou pythonu tuple obsahujúce iba súradnice (x, y).

Matica vzdialenosťí je reprezentovaná dvojrozmerným listom. Je dôležité povedať, že uložená matica je iba spodná trojuholníková matica, pretože horná časť matice je symetrická, takže je zbytočné s ňou pracovať a už vôbec ukladať ju, čo zabezpečí značnú redukciu výpočtov. Úvodná matica vzdialenosťí je vytváraná spolu s generovaním nových bodov. V prípade zlúčenia klastrov sa prepočíta matica iba pre tento novo vytvorený klaster (viď obr. 2.).

Program si ukladá zoznam klastrov `clusters` a zoznam centroidov/medoidov do listu `clustering_points`. Týmto sa zabráni opäťovnej rekalkulácií všetkých centroidov alebo medoidov. Centroid alebo medoid sa prepočíta iba v prípade zlúčenia klastru.

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

Obr. 1. Príklad reprezentovania matice  
(zdroj: <https://www.youtube.com/watch?v=T1ObCUpjq3o>)

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.23	0.2	0		
P4	0.37	0.20	0.19	0	
P5	0.34	0.14	0.34	0.29	0

Obr. 2. Príklad zlúčenia matice  
(zdroj: <https://www.youtube.com/watch?v=T1ObCUpjq3o>)

```
class Clustering:
    def __init__(self,
                 amount,
                 cluster_count,
                 clustering_type,
                 seed
                 ) :
        ...
        ...

    def agglomerative_clustering(self):
        ...

    def update_distance_matrix(self, merged_indices):
        ...
```

```

@staticmethod
def calculate_cluster_centroid(cluster):
    ...

def calculate_cluster_medoid(self, cluster):
    ...

def evaluate_cluster(self, i, cluster, color):
    ...

def get_new_row(self, point):
    ...

def generate_init_points(self, count: int, seed: int):
    ...

def generate_other_points(self):
    ...

```

## 2.2. Aglomeratívne zhlukovanie

Kroky algoritmu aglomeratívneho zhlukovania sú podrobnešie vysvetlené v kapitole [1.3](#).

Hlavná funkcia algoritmu `agglomerative_clustering()`, prechádza maticu vzdialenosť za účelom nájdenia minimálnej hodnoty (tento dvojitý for cyklus nastaví indexy zlučovaných klastrov opačne z dôvodu používania spodnej trojuholníkovej matice), zlučuje klastre a nakoniec volá funkciu `update_distance_matrix()`.

```

def agglomerative_clustering(self):
    progress_bar = tqdm(
        desc="Agglomerative clustering", total=self.amount +
INITIAL_POINTS - self.cluster_count, unit="cluster"
    )

    while len(self.clusters) > self.cluster_count:
        # Find the two closest clusters
        min_distance = float('inf')
        merge_indices = (0, 1)

        for i in range(len(self.distance_matrix)):

```

```

        for j in range(i):
            if self.distance_matrix[i][j] < min_distance:
                min_distance = self.distance_matrix[i][j]
                merge_indices = (j, i)

        # Merge the two closest clusters
        self.clusters[merge_indices[0]].extend(
self.clusters.pop(merge_indices[1]))

        # Merge clustering points
        self.clustering_points[merge_indices[0]] =
self.clustering_point(self.clusters[merge_indices[0]])
        del self.clustering_points[merge_indices[1]]

        # Update the distance matrix
        self.update_distance_matrix(merge_indices)
        progress_bar.update(1)

    progress_bar.close()

def update_distance_matrix(self, merged_indices):
    # Update the distance matrix after merging clusters
    new_cluster_index = merged_indices[0]
    distances_to_new_cluster = []

    for i in range(len(self.clusters)):
        if i != new_cluster_index:
            distance =
self.calculate_distance(self.clustering_points[new_cluster_index],
self.clustering_points[i])
            distances_to_new_cluster.append(distance)

    # Remove the old distances related to the merged clusters
    del self.distance_matrix[merged_indices[1]]
    for i in range(merged_indices[1] + 1,
len(self.distance_matrix)):
        del self.distance_matrix[i][merged_indices[1]]

    # Update distance matrix to the merged cluster
    new_row_distances =
distances_to_new_cluster[:merged_indices[0]]
    self.distance_matrix[merged_indices[0]] = new_row_distances
+ [0]

```

```

# Update distance matrix from the merged cluster
    for i in range(merged_indices[0] + 1,
len(self.distance_matrix)):
        self.distance_matrix[i][new_cluster_index] =
distances_to_new_cluster[i - 1]

```

## 2.3. Používateľské rozhranie

Používateľské rozhranie je rozdelené na CLI a GUI. CLI poskytuje používateľovi nastaviť seed, počet klastrov, výber typu či sa bude zhlukovať podľa centroidu alebo podľa medoidu a nakoniec počet bodov koľko má obsahovať 2D priestor. Ďalej sa zobrazí log, kde sú zobrazené čísla klastrov a vyhodnotenie s príslušnou farbou klastrov.

GUI zobrazuje riešenie, bližšie si ho priblížime v ďalšej kapitole [Vizualizácia](#).

```

Input world seed 🌱 >> 123456789
💡 Input number of clusters (15 - 20) >> 20
⚙️ Choose clustering type using centroid 1 or medoid 2 >> 1
💡 Input number of points >> 10000
Generating points: 100% [██████████] 10020/10020 [00:00<00:00, 16112.31point/s]
Agglomerative clustering: 100% [██████████] 10000/10000 [06:09<00:00, 27.06cluster/s]

Cluster evaluation:
✓ cluster number: 0 | ○ average point distance from centroid: 93.97963816115335 | 🌱 #DB7093
✓ cluster number: 1 | ○ average point distance from centroid: 164.94075165388023 | 🌱 #FF0000
✓ cluster number: 2 | ○ average point distance from centroid: 159.87691484117116 | 🌱 #E9967A
✓ cluster number: 3 | ○ average point distance from centroid: 131.83845379033303 | 🌱 #808080
✓ cluster number: 4 | ○ average point distance from centroid: 156.41545996482023 | 🌱 #008080
✓ cluster number: 5 | ○ average point distance from centroid: 171.82645440742365 | 🌱 #COCOCO
✓ cluster number: 6 | ○ average point distance from centroid: 141.40061314167647 | 🌱 #7CFC00
✓ cluster number: 7 | ○ average point distance from centroid: 144.72918078293864 | 🌱 #483D8B
✓ cluster number: 8 | ○ average point distance from centroid: 186.42029793638517 | 🌱 #FA8072
✓ cluster number: 9 | ○ average point distance from centroid: 166.017235781294 | 🌱 #008000
✓ cluster number: 10 | ○ average point distance from centroid: 139.13732563255647 | 🌱 #7B68EE
✓ cluster number: 11 | ○ average point distance from centroid: 128.14532724395772 | 🌱 #AFEEEE
✓ cluster number: 12 | ○ average point distance from centroid: 139.93778193462927 | 🌱 #6A5ACD
✓ cluster number: 13 | ○ average point distance from centroid: 102.09293634067704 | 🌱 #FF4500
✓ cluster number: 14 | ○ average point distance from centroid: 161.14601999920123 | 🌱 #4682B4
✓ cluster number: 15 | ○ average point distance from centroid: 179.00520925767376 | 🌱 #800000
✓ cluster number: 16 | ○ average point distance from centroid: 148.69388787522288 | 🌱 #F08080
✓ cluster number: 17 | ○ average point distance from centroid: 155.3495460898223 | 🌱 #FF69B4
✓ cluster number: 18 | ○ average point distance from centroid: 110.93036984133606 | 🌱 #00FA9A
✓ cluster number: 19 | ○ average point distance from centroid: 173.34860758081814 | 🌱 #7FFF00

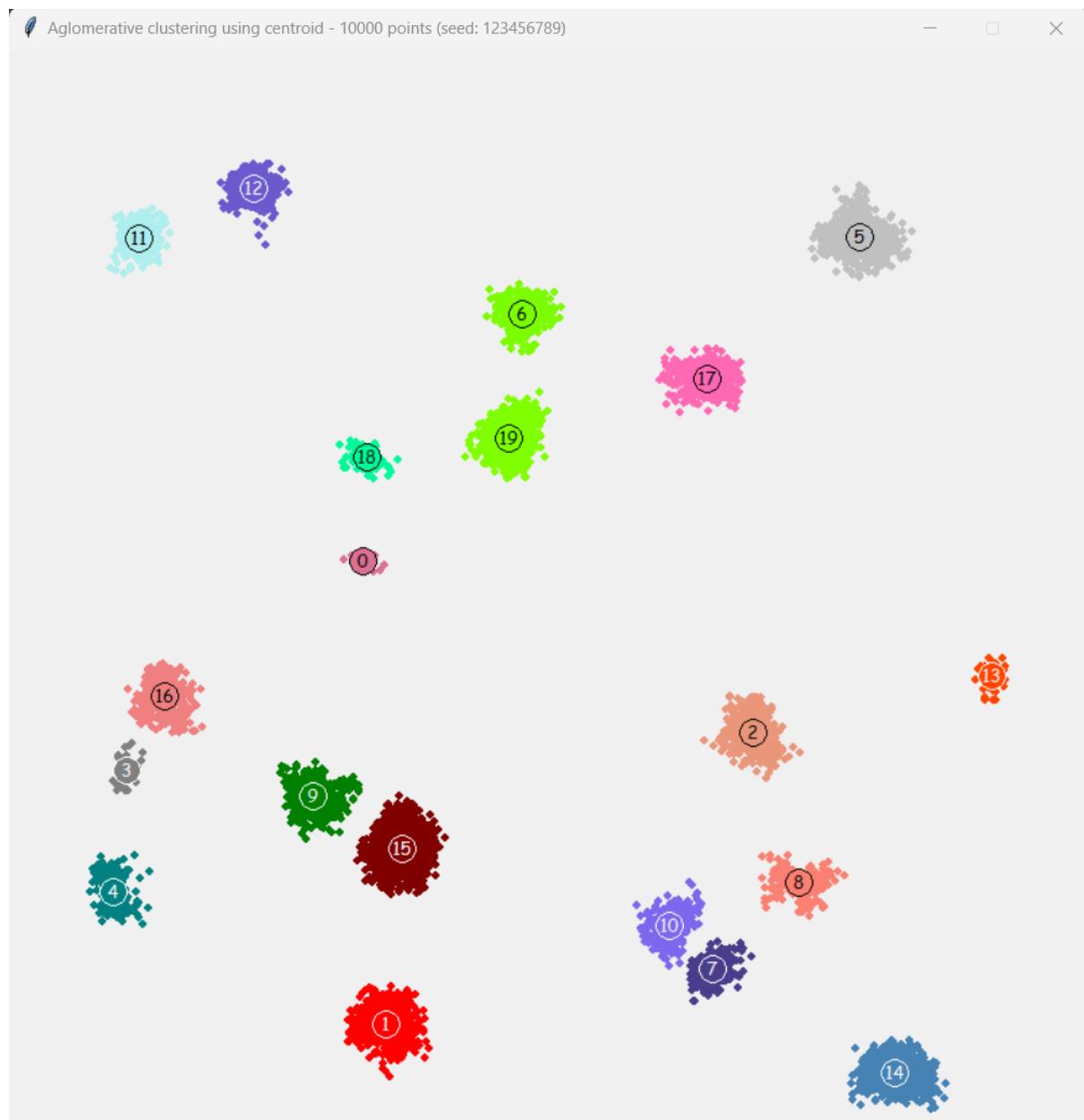
💡 INFO: RESULT IS DISPLAYED IN WINDOW
- Agglomerative Clustering

```

Obr. 3. Príklad CLI

## 2.4. Vizualizácia

Finálne klastre sú graficky vizualizované pomocou grafickej knižnice Tkinter. Každý klaster má náhodnú unikátnu farbu zo stanoveného zoznamu farieb v programe. Finálny centroid alebo medoid klastru je zobrazený 3x väčším bodom s číslom príslušného klastru. Názov okna je dynamický, čo znamená, že informuje o aglomeratívnom klastrovaní na základe úvodných nastavení používateľa. Pre lepšiu čitateľnosť čísel bola implementovaná funkcia, ktorá vyhodnocuje svetlosť farby.

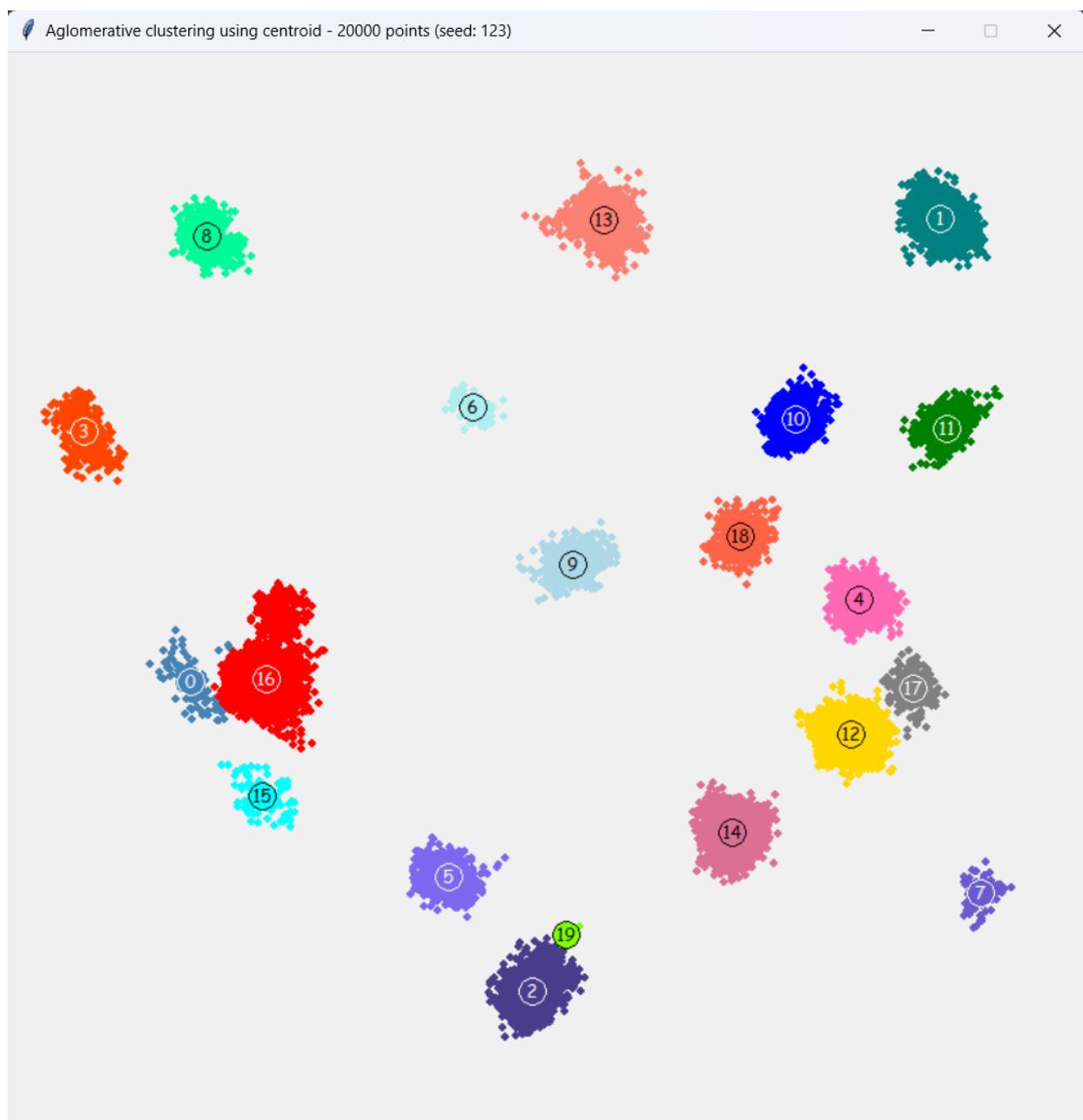


Obr. 4. Príklad vizualizácie - Klastrovanie 10000 bodov s využitím centroidu

### 3. Testovanie

Testovanie prebiehalo na nasledujúcich datasetoch a časy testovania sú brané z knižnice Tqdm, ktorá poskytuje sofistikované konzolové progress bary.

Test (seed:123)	Klastre	Dataset	Centroid	Medoid
1.	20	5000	00:42	00:42
2.	20	10000	05:57	05:59
3.	20	20000	45:55	47:12



Obr. 5. Agglomeratívne klastrovanie - 20000 bodov (45 minút)

```

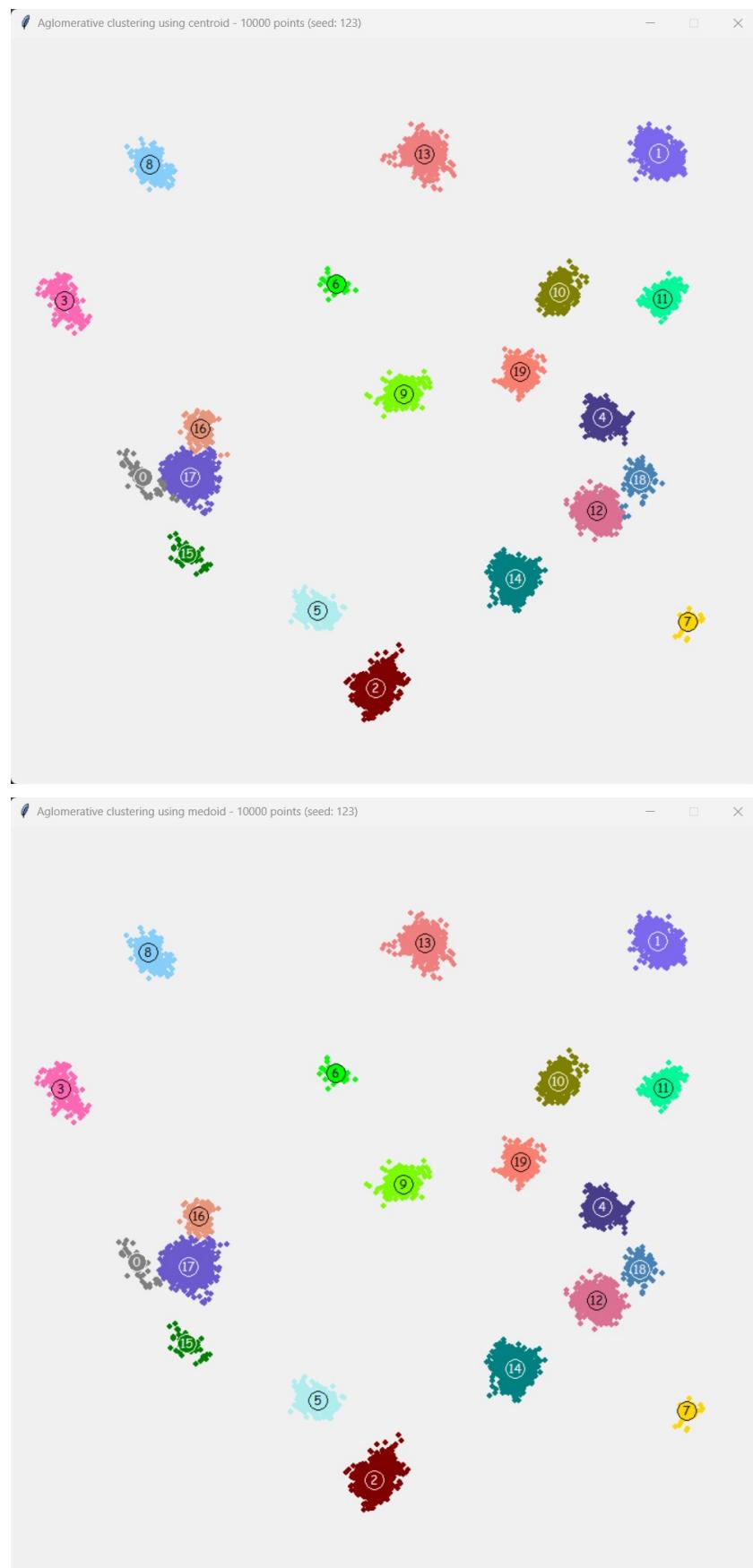
Input world seed 🌎 >> 123
🌐 Input number of clusters (15 - 20) >> 20
⚙️ Choose clustering type using centroid 1 or medoid 2 >> 1
🌐 Input number of points >> 20000
Generating points: 100%|██████████| 20020/20020 [00:02<00:00, 9942.76point/s]
Agglomerative clustering: 100%|██████████| 20000/20000 [45:55<00:00, 7.26cluster/s]

Cluster evaluation:
✓ cluster number: 0 | ○ average point distance from centroid: 181.12462379419628 | 🌎 #4682B4
✓ cluster number: 1 | ○ average point distance from centroid: 186.67290058920386 | 🌎 #008080
✓ cluster number: 2 | ○ average point distance from centroid: 177.4983225628142 | 🌎 #483D8B
✓ cluster number: 3 | ○ average point distance from centroid: 200.6588460580884 | 🌎 #FF4500
✓ cluster number: 4 | ○ average point distance from centroid: 153.40125136101867 | 🌎 #FF69B4
✓ cluster number: 5 | ○ average point distance from centroid: 156.92685625560244 | 🌎 #7B68EE
✓ cluster number: 6 | ○ average point distance from centroid: 133.22237307114028 | 🌎 #AFEEEE
✓ cluster number: 7 | ○ average point distance from centroid: 139.7006760665005 | 🌎 #6A5ACD
✓ cluster number: 8 | ○ average point distance from centroid: 161.11444617698555 | 🌎 #00FA9A
✓ cluster number: 9 | ○ average point distance from centroid: 154.57602924044969 | 🌎 #ADD8E6
✓ cluster number: 10 | ○ average point distance from centroid: 160.82464058739535 | 🌎 #0000FF
✓ cluster number: 11 | ○ average point distance from centroid: 147.9451516129518 | 🌎 #008000
✓ cluster number: 12 | ○ average point distance from centroid: 166.86412898281355 | 🌎 #FFD700
✓ cluster number: 13 | ○ average point distance from centroid: 179.04428137699898 | 🌎 #FA8072
✓ cluster number: 14 | ○ average point distance from centroid: 178.8179485435347 | 🌎 #DB7093
✓ cluster number: 15 | ○ average point distance from centroid: 160.26356567172107 | 🌎 #00FFFF
✓ cluster number: 16 | ○ average point distance from centroid: 227.60616136138665 | 🌎 #FF0000
✓ cluster number: 17 | ○ average point distance from centroid: 137.1565787345683 | 🌎 #808080
✓ cluster number: 18 | ○ average point distance from centroid: 157.64256258667783 | 🌎 #FF6347
✓ cluster number: 19 | ○ average point distance from centroid: 73.98904771010292 | 🌎 #7FFF00

ℹ️ INFO: RESULT IS DISPLAYED IN WINDOW
    - Agglomerative Clustering

```

Obr. 6. Aglomeratívne klastrovanie - 20000 bodov



Obr. 7. Porovnanie vizualizácie klastrovania 10000 bodov pre centroid a medoid

```

Input world seed 🌎 >> 123
# Input number of clusters (15 - 20) >> 20
⚙ Choose clustering type using centroid 1 or medoid 2 >> 1
✖ Input number of points >> 10000
Generating points: 100% [██████████] 10000/10000 [00:01<00:00, 9999.80point/s]
Agglomerative clustering: 100% [██████████] 10000/10000 [05:57<00:00, 27.98cluster/s]

Cluster evaluation:
✓ cluster number: 0 | ○ average point distance from centroid: 216.94611285913289 | ⓘ #000000
✓ cluster number: 1 | ○ average point distance from centroid: 179.11549265420656 | ⓘ #7B48E6
✓ cluster number: 2 | ○ average point distance from centroid: 170.778072080027 | ⓘ #000000
✓ cluster number: 3 | ○ average point distance from centroid: 199.6229493023484 | ⓘ #F6984
✓ cluster number: 4 | ○ average point distance from centroid: 141.71519884159287 | ⓘ #483D8B
✓ cluster number: 5 | ○ average point distance from centroid: 145.9664915796753 | ⓘ #AEEEEE
✓ cluster number: 6 | ○ average point distance from centroid: 126.1715469619209 | ⓘ #00FF00
✓ cluster number: 7 | ○ average point distance from centroid: 113.4346400121104 | ⓘ #FFD700
✓ cluster number: 8 | ○ average point distance from centroid: 148.5807477682764 | ⓘ #87CEFA
✓ cluster number: 9 | ○ average point distance from centroid: 145.2710932858037 | ⓘ #7CFCC00
✓ cluster number: 10 | ○ average point distance from centroid: 149.42440278865232 | ⓘ #008000
✓ cluster number: 11 | ○ average point distance from centroid: 131.4324732077688 | ⓘ #00FA9A
✓ cluster number: 12 | ○ average point distance from centroid: 152.8430343139811 | ⓘ #0B7093
✓ cluster number: 13 | ○ average point distance from centroid: 167.84220835075578 | ⓘ #F08080
✓ cluster number: 14 | ○ average point distance from centroid: 166.1360706526706 | ⓘ #008080
✓ cluster number: 15 | ○ average point distance from centroid: 129.26547476948116 | ⓘ #008000
✓ cluster number: 16 | ○ average point distance from centroid: 148.8666183050918 | ⓘ #E9967A
✓ cluster number: 17 | ○ average point distance from centroid: 167.71507925351449 | ⓘ #6A5AC0
✓ cluster number: 18 | ○ average point distance from centroid: 134.91484107721848 | ⓘ #468284
✓ cluster number: 19 | ○ average point distance from centroid: 147.86061698717486 | ⓘ #FA8072

✖ INFO: RESULT IS DISPLAYED IN WINDOW
- Agglomerative Clustering

Input world seed 🌎 >> 123
# Input number of clusters (15 - 28) >> 20
⚙ Choose clustering type using centroid 1 or medoid 2 >> 2
✖ Input number of points >> 10000
Generating points: 100% [██████████] 10000/10000 [00:00<00:00, 16216.86point/s]
Agglomerative clustering: 100% [██████████] 10000/10000 [05:59<00:00, 27.84cluster/s]

Cluster evaluation:
✓ cluster number: 0 | ○ average point distance from medoid: 160.43218914030032 | ⓘ #000000
✓ cluster number: 1 | ○ average point distance from medoid: 179.0112539419065 | ⓘ #7B68E6
✓ cluster number: 2 | ○ average point distance from medoid: 170.55165492254 | ⓘ #000000
✓ cluster number: 3 | ○ average point distance from medoid: 201.326002775463 | ⓘ #FFA984
✓ cluster number: 4 | ○ average point distance from medoid: 141.8099471316022 | ⓘ #483D8B
✓ cluster number: 5 | ○ average point distance from medoid: 145.55645150933827 | ⓘ #AEEEEE
✓ cluster number: 6 | ○ average point distance from medoid: 125.70357226036589 | ⓘ #00FF00
✓ cluster number: 7 | ○ average point distance from medoid: 111.96335736177645 | ⓘ #FFD700
✓ cluster number: 8 | ○ average point distance from medoid: 147.7461492560287 | ⓘ #87CEFA
✓ cluster number: 9 | ○ average point distance from medoid: 145.2170379489142 | ⓘ #7CFCC00
✓ cluster number: 10 | ○ average point distance from medoid: 149.41670423564284 | ⓘ #008000
✓ cluster number: 11 | ○ average point distance from medoid: 131.2074643164758 | ⓘ #00FA9A
✓ cluster number: 12 | ○ average point distance from medoid: 153.58171616046752 | ⓘ #0B7093
✓ cluster number: 13 | ○ average point distance from medoid: 168.135563830335618 | ⓘ #F08080
✓ cluster number: 14 | ○ average point distance from medoid: 166.0515470821068 | ⓘ #008080
✓ cluster number: 15 | ○ average point distance from medoid: 129.8688767641546 | ⓘ #008000
✓ cluster number: 16 | ○ average point distance from medoid: 140.072278986007 | ⓘ #E9967A
✓ cluster number: 17 | ○ average point distance from medoid: 170.46013272049197 | ⓘ #6A5AC0
✓ cluster number: 18 | ○ average point distance from medoid: 126.03346301642148 | ⓘ #468284
✓ cluster number: 19 | ○ average point distance from medoid: 147.82790342877405 | ⓘ #FA8072

✖ INFO: RESULT IS DISPLAYED IN WINDOW
- Agglomerative Clustering

```

Obr. 8. Porovnanie vyhodnotenia klastrovania 10000 bodov pre centroid a medoid

## 4. Zhodnotenie

Program korektné zhľukuje body na základe uvedeného testovania a vyhodnocovania klastrov. Taktiež vďaka niekoľko násobnému a neustálemu debugovaniu implementovaných algoritmov programu, sledovaním zmien hodnôt v matici vzdialenosí a kalkulácií príslušných centroidov alebo medoidov po zlučovaní klastrov, možeme povedať, že projekt aglomeratívne klastrovanie bol úspešne implementovaný.