**FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA**

Ilkovičova 2, 842 16 Bratislava 4

**2022/2023**

Počítačové a komunikačné siete

**Zadanie č.2**

**Návrh programu a komunikačného protokolu**

**Cvičiaci: Ing. Miroslav Bahleda, PhD. Vypracoval: Šimon Valíček**

**Čas cvičení: Piatok 8:00 – 9:40 AIS ID: 116320**

Obsah

**Návrh programu a komunikačného protokoli**

1. **Úvod........................................................................................................................................3**
   1. *Opis zadania......................................................................................................................3*
   2. *Implementačné prostredie................................................................................................4*
2. **Implementácia.........................................................................................................................4**
   1. *Úvod..................................................................................................................................4*
   2. *Generovanie bodov...........................................................................................................4*
   3. *Aglomeratívne zhlukovanie...............................................................................................4*
   4. *Klastre...............................................................................................................................5*
3. **Testovanie...............................................................................................................................9**
4. **Zhodnotenie...........................................................................................................................10**
5. **Záver.......................................................................................................................................11**
6. **Úvod**
   1. **Opis zadania**

**Zadanie úlohy:**

Máme 2D priestor, ktorý má rozmery X a Y, v intervaloch od -5000 do +5000. Tento 2D priestor vyplňte 20 bodmi, pričom každý bod má náhodne zvolenú polohu pomocou súradníc X a Y. Každý bod má unikátne súradnice (t.j. nemalo by byť viacej bodov na presne tom istom mieste).

Po vygenerovaní 20 náhodných bodov vygenerujte ďalších 20000 bodov, avšak tieto body nebudú generované úplne náhodne, ale nasledovným spôsobom:

1. Náhodne vyberte jeden zo **všetkých doteraz vytvorených** bodov v 2D priestore. (nie len z prvých 20)  
   Ak je bod príliš blízko okraju, tak zredukujete príslušný interval 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

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:

* aglomeratívne zhlukovanie, kde stred je centroid
* aglomeratívne zhlukovanie, kde stred je medoid (stačí 5000 bodov)

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.

Dokumentácia musí obsahovať opis konkrétne použitých algoritmov a reprezentácie údajov. V závere zhodnoťte dosiahnuté výsledky ich porovnaním.

**Poznámka:** Je vhodné použiť rôzne optimalizácie pre dostatočne efektívnu prácu Vášho zhlukovača:

* Použitie 2-rozmernej matice vzdialenosti dvojíc bodov. Naplnenie takejto matice má kvadratickú zložitosť. Potom sa hľadá najbližšia dvojica (najmenšie číslo v matici), to má opäť kvadratickú zložitosť, ale nenásobia sa tie časy, len sčítavajú. Po výbere najbližšej dvojice túto dvojicu treba zlúčiť a tým sa zníži veľkosť matice o 1 (lebo sa zníži počet zhlukov o 1) Pri tom sa aktualizujú len vzdialenosti pre tento nový zhluk (len jeden stĺpec/riadok, zvyšok matice ostáva nezmenený).
* PyPy je implementácia programovacieho jazyka Python. PyPy často beží rýchlejšie ako štandardná implementácia CPython, pretože PyPy používa just-in-time kompilátor.
  1. **Implementačné prostredie**

Pre prácu na tomto zadaní som sa rozhodol tak, ako aj pri predošlých a zvolil som si jazyk JavaScript a implementačné prostredie Visual Studio Code. Rozhodol som sa tak z toho dôvodu, že súčasťou zadania je vykresľovanie získaných výsledkov a kde sa dajú vykresliť údaje krajšie, než v HTML canvase ? Navyše som dostal zadanie **c)**, ktoré má povesť najdlhšie trvajúceho zadania, a tak som sa rozhodol spraviť to celé animované, aby sme sa nenudili počas prezentovania, kým sa to celé spustí. Z týchto dôvodov sa mi tento výber implementačných prostriedkov javila ako najlepšia voľba.

1. **Implementácia**
   1. **Úvod**

V tejto kapitole sa budem venovať už samotnému kódu. Podrobne vysvetlím, na akých princípoch funguje aglomeratívne zhlukovanie, čo je centroid, medoid, ako sa počítajú a ako to všetko spolu súvisí.

* 1. **Generovanie bodov**

Na začiatku je potrebné, vygenerovať si náhodné body. Jeden bod je reprezentovaný objektom, ktorý má nasledujúce vlastnosti:

* Id
* Súradnica X
* Súradnica Y
* Farba 1
* Farba 2

Vo všeobecnosti by stačilo použiť jednu farbu, avšak nakoľko pre odľahčenie kódu si pre centroid a medoid generujem rovnaké body, aby som následné matice vzdialeností počítal len raz. Takto reprezentovaných bodov si vygenerujem na začiatku 20.

Po vygenerovaní 20 bodov, vypočítam ich vzdialenosti, t.j. každého s každým a vytvorím si maticu. Pri **n** bodoch, je veľkosť matice **n\*(n-1)/2**.

Následne z týchto 20 bodov vyberiem náhodný bod, vypočítam jeho súradnice na základe zadanáho offsetu a ho pridám do môjho poľa bodov. Ďalej vezmem daný bod a vypočítam jeho vzdialenosti od všetkých ostatných bodov. Vzdialenosti pridám do matice vzdialeností.

Tento princíp opakujem, až pokiaľ nedosiahnem počet bodov, ktorý je uvedený v zadaní, t.j. 20 000 pre centroid a 5000 pre medoid, pričom tieto nové body už negenerujem z pôvodných 20, ale vždy vyberiem náhodný bod z poľa bodov, ktoré sa teda každou iteráciou zvýši o 1.

* 1. **Aglomeratívne zhlukovanie**

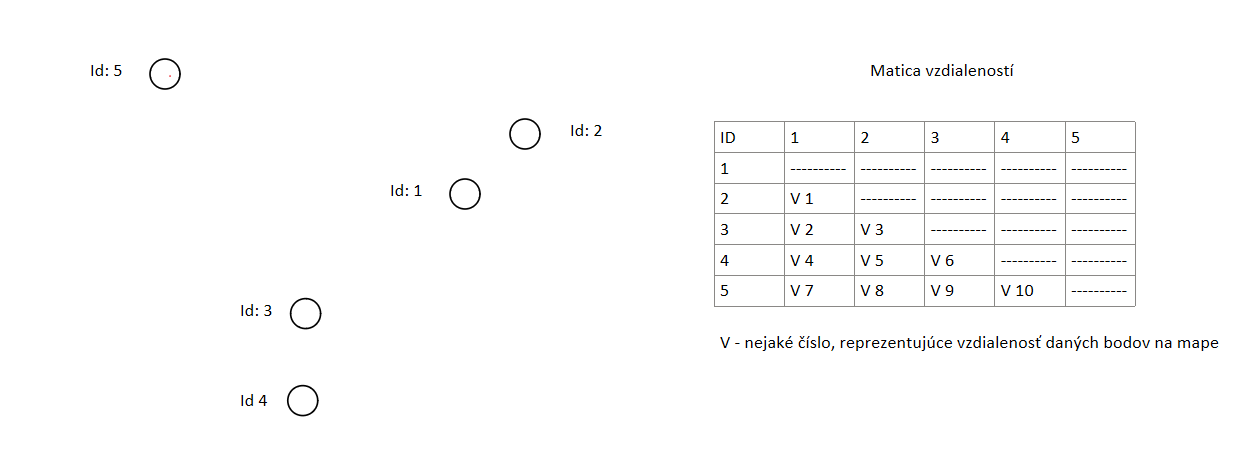
Princíp aglomeratívneho zhlukovania spočíva v tom, že na začiatku algoritmu máme rovnaký počet bodov ako klastrov. Každou iteráciou tohto algoritmu znížime počet klastrov o 1, nakoľko vždy zhlukujeme dva body, ktoré sú od seba najmenej vzdialené. Pre lepšie pochopenie si poďme vysvetliť, čo to vlastne klaster je a ako ho reprezentovať pomocou centroidu a medoidu.

* 1. **Klastre**

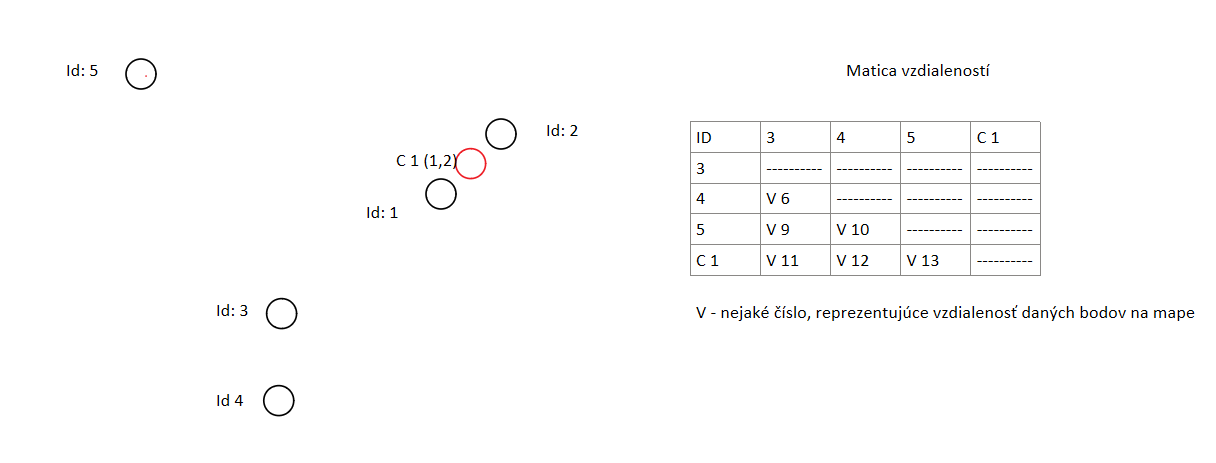
Klaster je zhluk bodov, ktorých vzdialenosť od centroidu (resp. medoidu) je nanajvýš 500, ako je to napísané v zadaní. Nakoľko spočiatku nemám žiadne centroidy, jeden môj klaster je vlastne jeden bod na mape. Znamená to teda, že na začiatku pracujem s 20 020 klastrami. Klastre začínam vytvárať nasledovne (pre ukážku to vysvetlím na piatich bodoch, nakoľko pre 20 020 bodov by som to nestihol spraviť ani za niekoľko mesiacov, ale princíp je rovnaký):

**Centroid**

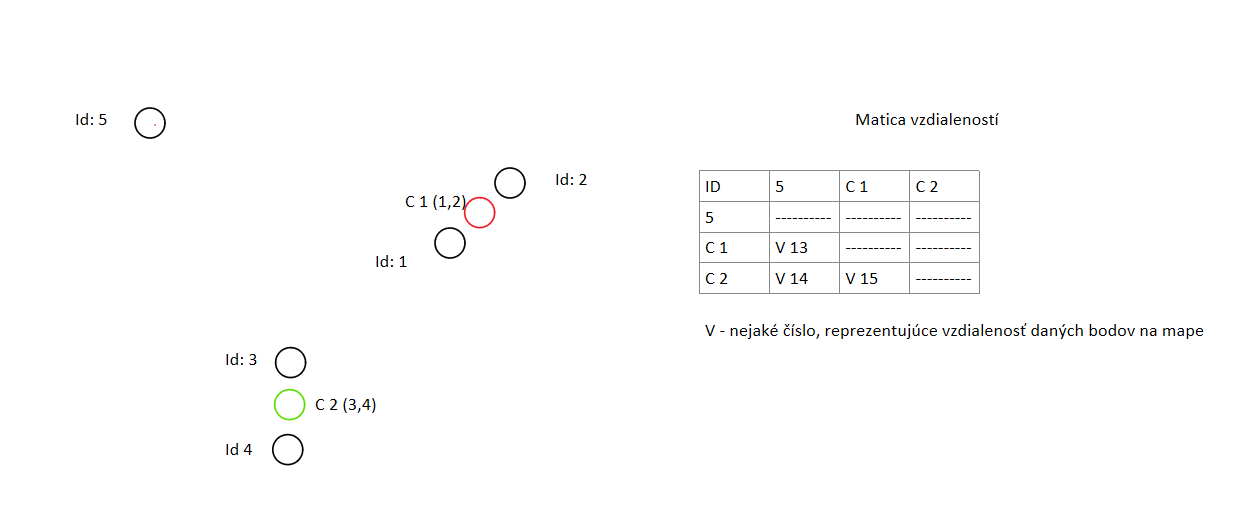
Ako som už písal vyššie, na začiatku máme **n** bodov a maticu obsahujúcu **n\*(n-1)/2** bodov. Na príklade pre mojich 5 bodov to na začiatku vyzerá nejak takto, pričom každý bod je momentálne jeden klaster. Moje súčastné klastre teraz sú: **(1),(2),(3),(4),(5)**.



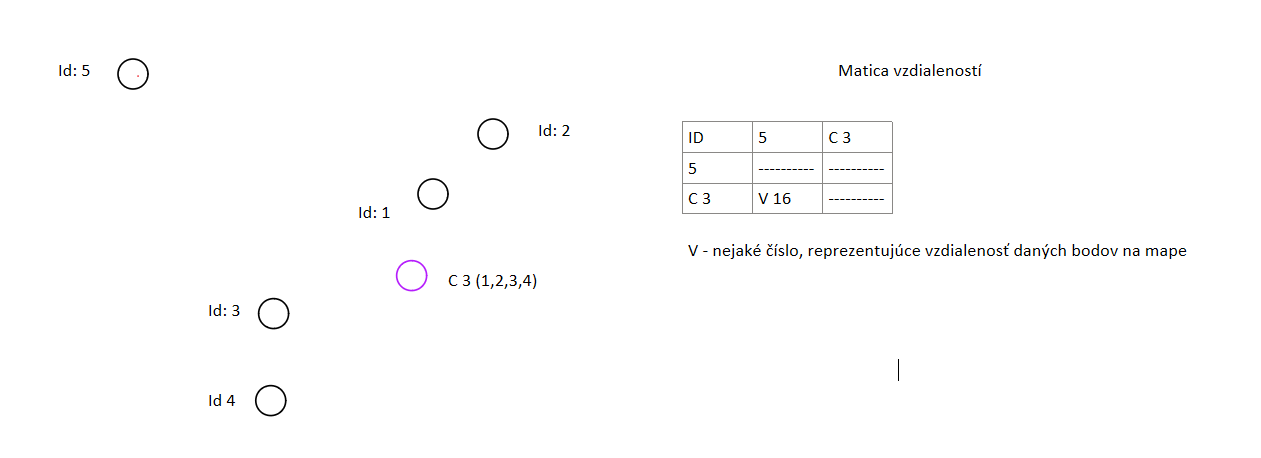
V prvom kroku preiterujem maticu a nájdem najmenšiu vzdialenosť dvoch bodov. Povedzme, že pre môj príklad to bude vzdialenosť V 1, čiže vzdialenosť bodov 1 a 2. Pozriem, či je vzdialenosť týchto dvoch bodov menšia ako 500. Ak áno, zavolám si funkciu, ktorá mi pre tieto dva body vytvorí centroid. Po skončení tejto funkcie, dostanem nasledujúci výsledok.



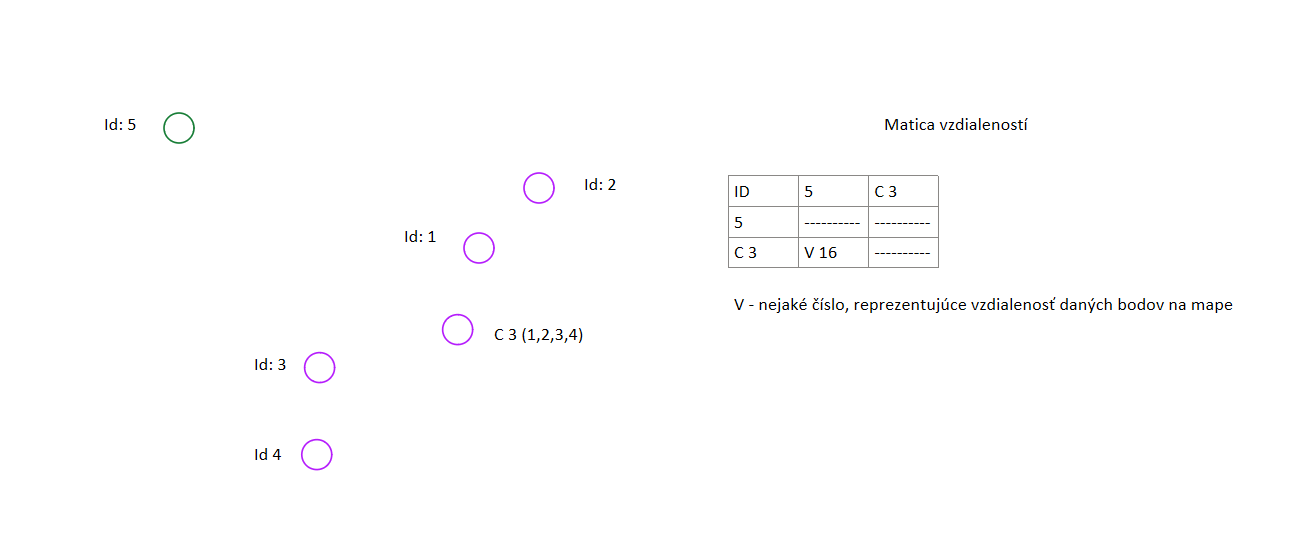
Body 1 a 2 sú už teraz reprezentované centroidom C1, ktorého súradnice sú aritmetickým priemerom súradníc bodov 1 a 2. Tomuto centroidu som priradil neajkú farbu. Následne som upravil maticu vzdialeností, z ktorej som odstránil všetky vzdialenosti od bodov 1 a 2. Namiesto nich som prepočítal vzdialenosti centroidu od ostatných bodov a pritdal tieto do matice. Pre lepšie pochopenie som zachoval pôvodné čísla vzdialeností, aby bolo jasne vidieť, ktoré do tabuľky pribudli, a naopak, ktoré sa v nej už nenachádzajú. Môj počet klastrov sa znížil o jedna. Moje súčastné klastre teraz sú: **(1,2),(3),(4),(5).** Pokračujem tým, že preiterujem maticu a nájdem najmenšiu vzdialenosť, ktorá sa v nej nachádza. Povedzme, že to bude vzdialenosť V 6, čiže vzdialenosť bodov 3 a 4. Pozriem, či je táto vzdialenosť menšia ako 500. Ak áno, znovu si zavolám funkciu, ktorá mi pre tieto dva body vytvorí centroid. Po skončení tejto funkcie dostanem nasledujúci výsledok.



Body 3 a 4 sú už teraz reprezentované centroidom C1, ktorého súradnice sú aritmetickým priemerom súradníc bodov 3 a 4. Tomuto centroidu som priradil neajkú farbu. Následne som upravil maticu vzdialeností, z ktorej som odstránil všetky vzdialenosti od bodov 3 a 4. Namiesto nich som prepočítal vzdialenosti centroidu od ostatných bodov a pritdal tieto do matice. Môj počet klastrov sa opäť znížil o jedna. Moje súčastné klastre teraz sú: **(1,2),(3,4),(5).**  Opäť budem pokračovať tým, že preiterujem maticu a nájdem najmenšiu vzdialenosť, ktorá sa v nej nachádza. Povedzme, že to bude vzdialenosť V 15, čiže vzdialenosť centroidov C1 a C2. Pozriem, či je táto vzdialenosť menšia ako 500. Ak áno, znovu si zavolám funkciu, ktorá mi pre tieto dva body (centroidy) vytvorí centroid. Po skončení tejto funkcie dostanem nasledujúci výsledok.



Vidíme, že z pôvodných centroidov C1 a C2 sa stal centroid C3, pričom jeho súradnice sú aritmetickým priemerov bodov, ktoré boli obsahom pôvodných centroidov, teda bodov (1,2,3,4) Tomuto novému centroidu som priradil nejakú farbu. Následne som upravil maticu vzdialeností, z ktorej som odstránil všetky vzdialenosti od bodov resp. centroidov C1 a C2. Namiesto nich som prepočítal vzdialenosti centroidu od ostatných bodov a pridal tieto do matice. Môj počet klastrov sa opäť znížil o jedna, teda moje súčastné klastre teraz sú: **(1,2,3,4),(5)**. Opäť pokračujem tým, že preiterujem maticu a nájdem najmenšiu vzdialenosť, ktorá sa v nej nachádza. Povedzeme, že je to vzdialenosť V 16, čiže vzdialenosť bodu 5 od centroidu C3. Pozriem, či je táto vzdialenosť menšia, ako 500. Povedzme, že v tomto prípade je vzdialenosť väčšia ako 500. Tu sa môj cyklus teda uzatvára a ďalšia iterácia už neprebehne. Teraz je potrebné priradiť klastrom ich farby. Momentálne mám dva klastre, ako je možné vidieť na osiach matice vzdialeností. Moje klastre sú: **(1,2,3,4),(5).** Nakoľko prvý klaster obsahuje centroid, ktorý je vyfarbený, všetkým bodom, ktorých súradnicami je centroid reprezentovaný, priradím farbu tohto centroidu. Nakoľko pre bod 5 centroid neexistuje a nemá ešte ani pridelenú farbu, nejakú mu náhodne vygenerujem. Výsledok vyzerá nejak takto.



Vo výsledku mám dva klastre, a teda aj dve farby. V praxi sa s najväčšou pravdepodobnosťou nestane, že by nejaký bod zostal stáť sám, avšak toto bola len ilustrácia, aby sme mali predstavu ako celý tento algoritmus funguje. V tomto prípade, by bol stredom klastra **(5)** centroid, ktorý by bol zároveň aj bodom. Ako vravím, v praxi je nízka šanca, že sa nejaký bod vygeneruje izolovaný od ostatných, preto spravidla každý klaster má svoj centroid, ktorý zároveň nie je žiadnym z bodov, ktoré doňho patria.

**Medoid**

V princípe, aglomeratívne zhlukovanie, kde stredom je medoid funguje na podobnej báze, ako pri centroide. V praxi to znamená, že body sa generujú rovnako, a rovnako tak sa počíta aj matica vzdialeností. Jediný rozdiel je v tom, že medoid sa počíta trocha odlišným spôsobom ako centroid. V prípade centroidu, išlo o aritmetický priemer bodov, ku ktorým patrim, zvyčajne tak teda išlo o bod, ktorý sa nachádzal mimo iných bodov v rámci klastra, aj keď pri veľkom štastí (resp. smole) to nemusí byť pravidlo. Naopak, medoid je vždy jedným z bodov v rámci klastra, do ktorého patrí.

Algoritmus na jeho zistenie je podobný tomu pri centroide, no pre lepšie pochopenie sa ho pokúsim podrobnejšie opísať v nasledujúcich riadkoch.

Na začiatku máme opäť niekoľko náhodne vygenerovaných bodov, a maticu vzdialeností všetkých týchto bodov. V prvom kroku algoritmu vyberieme najmenšiu vzdialenosť z tejto matice. Tu môžu nastať tieto tri prípady:

**1. Žiaden z týchto bodov nie je medoid**

V takomto prípade vezmeme tieto dva body a určíme si, ktorý z nich je medoid. Medoid je ten z bodov, ktorého suma vzdialeností od ostatných bodov v rámci klastra je najmenšia. V tomto prípade, je úplne jedno, ktorý bod si zvolíme ako medoid v rámci klastra, nakoľko je medzi nimi len jedna vzdialenosť, a teda suma ich vzdialeností od ostatných bodov v rámci klastra je rovnaká. Ja som sa rozhodol, že v takomto prípade si za medoid zvolím prvý z týchto dvoch bodov. Následne vzdialenosti týchto bodov vymažeme z matice vzdialeností, rovnako ako sme to robili aj pri centroidoch vyššie a pridáme vzdialenosti medoidu (ktorým budú odteraz reprezentované) o ostatných bodov.

**2. Jeden z týchto bodov je medoid, druhý nie**

V takomto prípade je potrebné spojiť tento bod a medoid do klastra, pričom treba opäť prekalkulovať medoid. Tu môžu nastať dva prípady.

**a) Nový bod, ktorý sme pridali do klastra je novým medoidom, alebo nejaký bod, ktorý je iný než náš povodný medoid je teraz novým medoidom**

V takomto prípade je potrebné vymazať z matice vzdialeností vzdialenosti pôvodného medoidu od ostatných bodov, zároveň tak aj vzdialenosti nového bodu, ktorý sme pridali do klastra. Následne prepočítam vzdialenosti nového medoidu od ostatných bodov a pridám ich do matice.

**b) Medoid zostáva pôvodný**

V prípade, že konkurz na náš nový medoid vyhral ten bod, ktorý bol pôvodne medoidom, vymažeme z matice iba vzdialenosti ostatných bodov od bodu, ktorý sme do klastra pridali a zároveň pridáme do klastra tento bod. Vzdialenosť medoidu od ostatných bodov v matici sa nezmenila, nakoľko medoid je stále ten istý bod, čo pôvodne, zmenil sa iba obsah medoidu, ktorý sa navýšil o jeden nový bod.

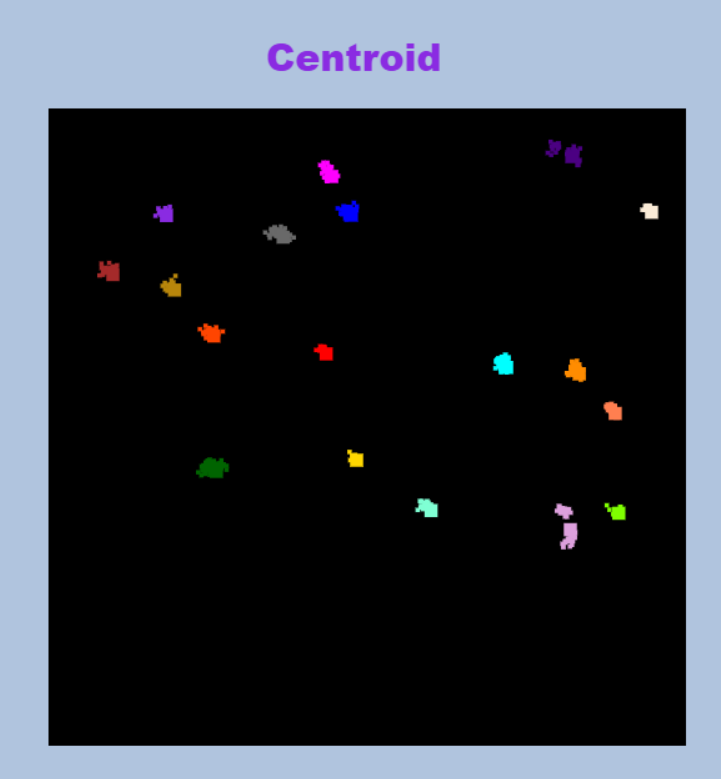
**3. Oba tieto body sú medoidy**

V takomto prípade je potrebné zlúčiť všetky body, reprezentované týmito medoidami, vymazať pôvodné dva medoidy a prepočítať nový medoid. Následne vymazať z matice vzdialeností všetky vzdialenosti týchto medoidov od ostatných bodov. Ďalej pridáme do matice vzdialenosti medoidu od ostatných bodov.

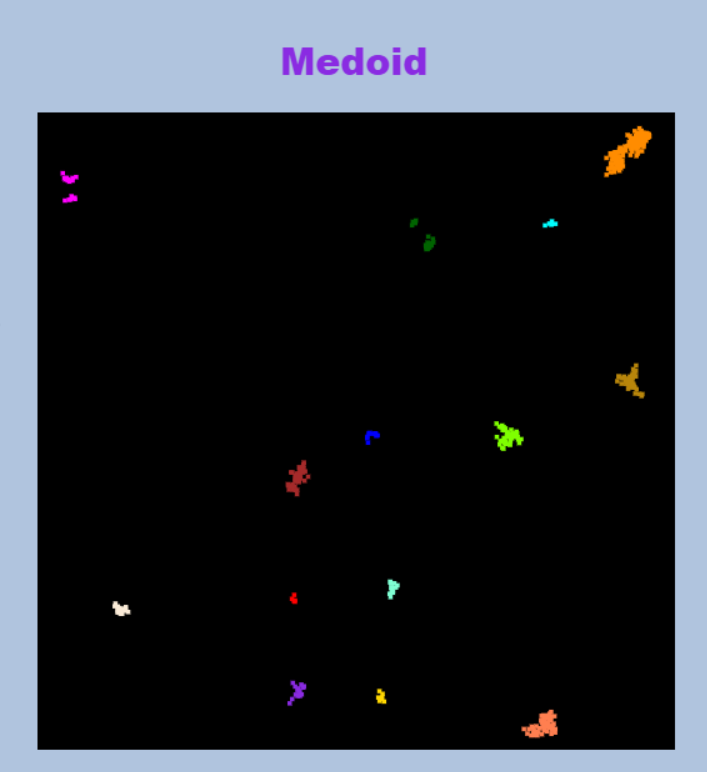
Opäť, ak je najmenšia vzdialenosť zostávajúca v matici väčšia ako 500, cyklus ukončujeme a vyfarbujeme body podľa toho, pod ktorý medoid patria. Ak nepatria pod žiaden, vygenerujeme im novú farbu.

1. **Testovanie**

Nakoľko v zadaní bolo písané, že testovanie musí prebehnúť na 20 000 prvkoch pri centroide a aspoň 5 000 pri medoide, a vzhľadom na to, že časová náročnosť tohto zadania bola veľmi vysoká, testovanie som spustil len raz.



Pri vygenerovaní 20 počiatočných bodov, a následnom vygenerovaní 20 000 dodatočných bodov vyzerá môj výstup nejak takto. Všetkých 20 020 bodov sa mi zredukovalo do 18 klastrov a výpočtový čas celého tohto programu bol 5 hodín aj 32 minút.



Po vygenerovaní 20 počiatočných bodov a následnom vygenerovaní dodatočných 5 000 bodov som získal vyššie uvedený výstup. Celkových 5 020 bodov sa mi zredukovalo len do 14 klastrov a a celkový čas bežania programu bol 3 hodiny a 26 minút.

1. **Zhodnotenie**

Dovolím si spätne konštatovať, že vzhľadom na vysokú časovú náročnosť zadania musím priznať, že som si zvolil nevhodné implementačné prosttredie. Jazyk JavaScript nie je prispôsobený na takúto analýzu dát, a preto by som v budúcnosti volil na takúto prácu Python a knižnicu PyPy.

1. **Záver**

Napriek tomu, že som nad týmto zadaním strávil viac času ako som pôvodne mienil, môžem skonšatovať, že ma to bavilo. Z celkohvého hľadiska tak toto zadanie, rovnako ako celý tento predmet hodnotím veľmi pozitívne.