

Package Genetic

Volanie metody genetic – vstup String zasifrovany text, Hasmap mapa frekvencie Ngramov, keySize odhad dlzky kluca

Vystup – kluc s najlepsou finess funkciou po 1000 iteraciach

Triedy:

**Chromosome** =reprezentacia kluca ako pole Integerov

=odsahuje Double Score pre fitness hodnotu daneho retazca

**Population**= ArrayList<Chromosome> populacia dlzky n jednotlivych Chromosomov

Metody: **genRandPop()**: generuje nahodnu populaciu klucov

**mergePop()**: spaja dokopy 2 populacie

**Selection**= trieda na vyber najlepsich jedincov alebo nahodnych jedincov

Vstupy: Zasifrovany text, mapa frekvencii n gramov

Metody: **setScoreToPop**: ohodnoti jednotlive chromosomy podla fitness funckie

**sortPop()**: zotriedi ohodnotenu populaciu od najnizsej fitness po najvyssiu

**selBest()**: vyberie z utriedenej populacie n jedincov

Vstupy: criterium {1,2,2}

Vystupy: nova populacia, kde sa najlepsi jedinec nachadza raz, 2. Najlepsi 2krat a 3.najlepsi 2krat

**selRand()**: do novej populacie vlozy n jedincov z utriedeneho pola



**Mutation**= trieda pomocou ktorej sa robia vybrane permutacne operacie na populacii

Metody: **muteSwap()**: prehodi 2 nahodne prvky v permutacii

Vstup: Double percentage: percento kolko jedincov zmutuje

1 2 3 4

Vystup: zmutovana populacia

2 1 3 4

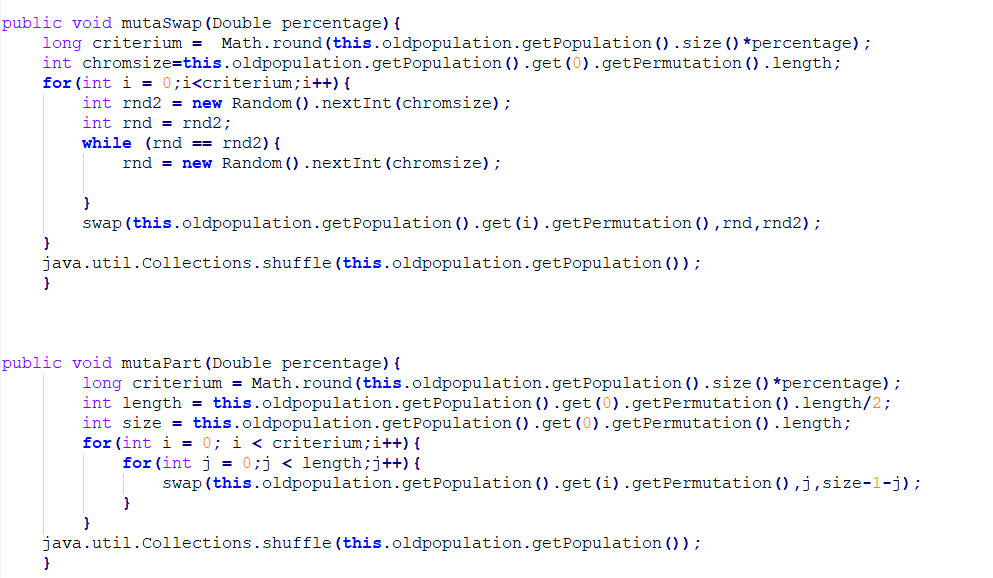
**mutePart()**: preusporiada poradie prvej a druhej polovice v permutacii

Vstup: Double percentage: percento kolko jedincov zmutuje

1 2 3 4

Vystup: zmutovana populacia

3 4 1 2

(existuje pre to aby ked kluc je 1 2 3 4 a nam dava dobry fitness 3 4 1 2 aby sa vyskusala aj tato moznost)

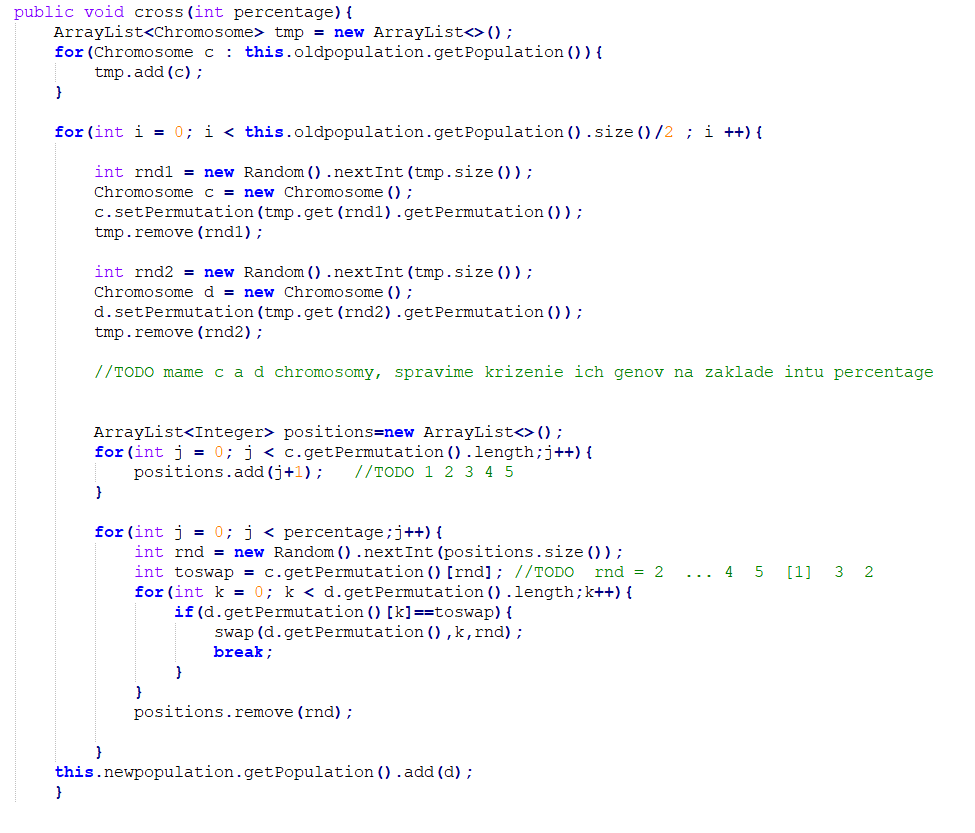
**CrossBreed** = krizenie medzi jedincami populacie

Vstupy: Double percentage: kolko prvov permutacie sa zkrizi

Vystupy: Nova populacia skrizenych jedincov o velkosti n/2

Metody: **cross()**

Vstup: nohodne 2 permutacie z pop. permuacia 1 4 **3** 2 a **3** 2 **1** 4, percentage 1

Vystup: jeden jedinec 1 2 3 4 swapne prvok **3** v druhom jedincovi na miesto, kde je v prvom jedincovi

**Genetic**= hlada kluc transpozicnej sifry

**removeDuplicates()**  = duplikaty v novovytvorenej populacii nahradza nahodnymi permutaciami, snaha o odstranenie lokalneho minima

genetic() hlada najlepsieho jedinca v 1000 iteraciach v populaciach o velkosti 20 jedincov, 2 najlepsi ostanu nezmeneni a vstupuju do dalsej iteracie, 24 jedincov sa skrizi a vytvori 12 novych unikatnych. 8 jedincov zmutuje 12+8+2 je nova populacia o velkosti 20. Program bol testovany fitness funkciou na mapach ngramovych frekvencii **listngramDec,** ktora pracuje s idealnym pripadom zhody ngramov, nie s referencnymi hodnotami. Aj napriek tomu, casto krat nie je mozne najst vysledok a to z dovodov: **Prepisovanie minimalneho jedinca**

**Zla implementacia volania fitness funkcie**

Samotny geneticky algoritmus pracuje predpokaldam tak ako by mal, chyba sa vyskytuje v nespravnej implementacii volania **fitness funkcie.**

**Potencialne vyuzite v pripade správneho fungovania (program budem este optimalizovat):**

Hladanie kluca transpozicnej sifri najma pre velke texty a velke kluce, kde referencne hodnoty ngramov su priblizne rovnake ako hodnoty ngramov zasifrovaneho textu. Kedze program nema metodu ako zastavit na zaklade uspesnosti alebo percentualnej zhody, musi sa vzdy na konci iteracii najlepsim klucom text prelozit a skontrolovat tak ci nam dava zmysel. V idealnom pripade by mal geneticky algoritmus najst priblizne podobny kluc akym sme text zasifrovali v case kratsom ako Hill Climbing, ale je dost pravdepodobne ze sa to tak nemusi stat vobec.