Ćwiczenie 1 Etap 1

Imię i nazwisko: Tomasz Łogisz

Nr. indeksu: 241370

Data: 11.03.2021r.

Z góry przepraszam za duże sprawozdanie, temat sztucznej inteligencji jest dla mnie dość ciekawy tak więc zrobiłem troszę więcej niż było to wymagane.

Została stworzona sama struktura projektu.

Projekt dzieli się na klasy:

1. Point – jest to klasa, która tworzy strukturę punktu – P(x,y) – jest to elementarna klasa, z której będzie się składać projekt.

W skład klasy Point zawierają się podstawowe metody jak gettery, czy metoda toString, a także metoda sumująca: P(x,y) + P(z,c) = P(x+z,y+c), oraz metoda sprawdzająca czy są to te same punkty – możemy powiedzieć, że punkty są takie same gdy mają te same współrzędne czyli P(x,y) == P(z,c) gdy x==z oraz y==c

1. Osobnik – jest to klasa, która przetrzymuje w sobie informację na temat pojedynczego lutowania na płytce PCB – mamy tutaj referencję na punkt początkowy, punkt końcowy oraz listę punktów pośrednich ( punkty pośrednie wyznaczają ścieżkę lutowania).

W skład klasy osobnik wchodzi dość więcej metod więc opisze tylko najważniejsze z nich.

Metoda createConnection - > metoda w sposób losowy tworzy ścieżkę pomiędzy punktem początkowym a końcowym.

Metoda checkConnection -> metoda sprawdzająca czy dany punkt nie powtarza się w ścieżce – mianowicie jest to metoda wprowadzająca ograniczenie, że w ścieżce pomiędzy punktem początkowym i końcowym nie pojawi się ten sam punkt kilkukrotnie

np. P(1,1) -> P(2,1) -> P(1,1) -> P(1,2) ------ błąd!

Uwaga! Jako osobnik uważam pojedyncze lutowanie na płytce PCB.

Nie jest to do końca prawidłowe nazewnictwo, lecz był to sam początek wykonywania zadania, przez co nie miałem jeszcze do końca uformowanej wizji implementacji problemu.

1. PlytkaPCB – jest to klasa będąca implementacją danej plytki PCB.

Posiada takie atrybuty jak wysokość i szerokośc (płytki) orz listę osobników czyli lutowań na tej płytce.

Z ważniejszych metod:

Metoda wczytajDane(String fileName) – jest to pobranie danych z pliku txt, które są podane jako przykładowe problemy do rozwiązania.

Metoda vPrint() -> metoda pokazująca (w konsoli) pokazujące graficznie Plytke PCB wraz z lutowaniami.

Metoda checkLimitations() -> metoda sprawdzająca czy pomiędzy lutowaniami doszło do przecięcia się, ta metoda w implementacji jest wyłączona czyli dopuszczone są przecięcia między lutowaniami na potrzebę przyszłego krzyżowania.

Metoda metodaLosowa() -> Jest to metoda, która generuje w sposób losowy lutowania dla wszystkich osobników na płytce PCB.

Metoda funkcjaOcenyDlugosciSciezki -> metoda obliczająca jak dużo mamy punktów pośrednich ( czyli obliczająca sume długości ścieżek dla wszystkich osobników na płytce PCB).

Metoda funkcjaOcenyIloscPrzeciec -> metoda obliczająca ilosc przeciec pomiedzy osobnikami na plytce PCB.

1. AlgorytmGenetyczny – jest to klasa, która będzie implementować algorytm genetyczny.

Klasa nie jest póki co skończona, lecz na chwilę obecną posiada dwa atrybuty, pierwszy z nich jest nazwa pliku z problemem do pobrania. Drugim natomiast jest lista płytek PCB czyli populacja.

Klasa ta zawiera póki co jedną ważną metodę – init\_population.

Metoda ta tworzy początkową populację w danej ilości. Czyli w praktyce tworzy daną ilość płytek PCB, z tymi samymi danymi początkowymi (czyli wysokość, szerokośc, oraz punkty początkowe i końcowe lutowań, wszystkie są zawarte w pliku tekstowym problemu).

Następnie każda płytka oblicza przy pomocy metody losowej dane lutowania.

W przyszłości zostanie tutaj zamieszczona reszta algorytmu.

Metoda ładowania danych: Klasa PlytkaPCB, Metoda: wczytajDane(String fileName)

Metoda funkcja przystosowania: Klasa PlytkaPCB, Metoda: funkcjaOcenyDlugosciSciezki() / Metoda: funkcjaOcenyIlosciPrzeciec()

Metoda losowa: Klasa PlytkaPCB, Metoda: metodaLosowa()

Poniżej przykłady działania programu.

Teraz zaprezentuje obecne wyniki:

\*\*\* Uwaga \*\*\*

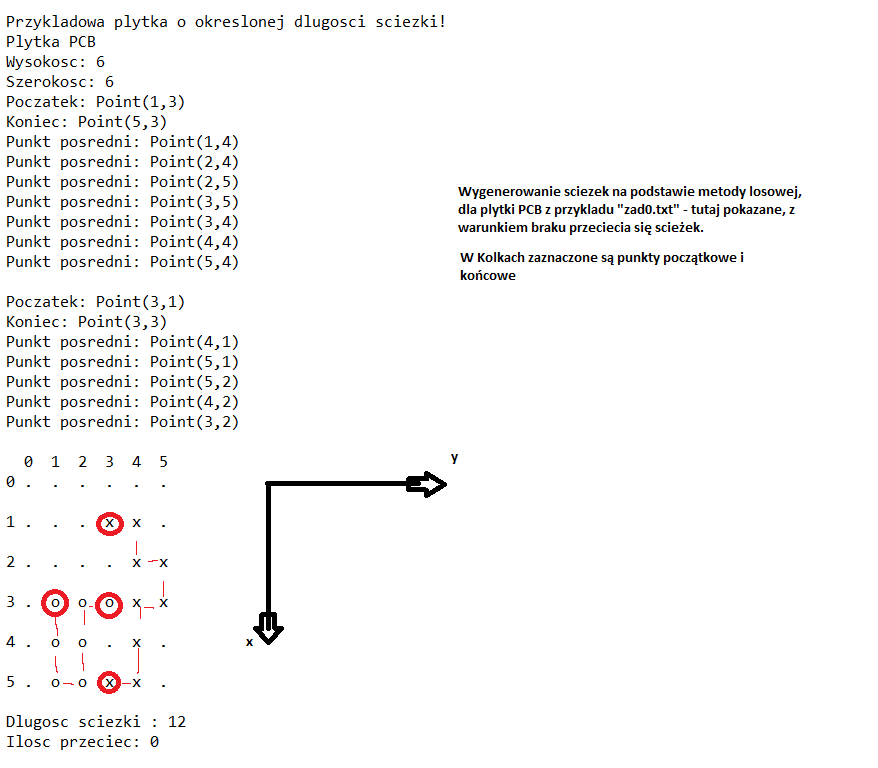
Zaprezentowane będą przykłady zarówno z przecięciami się ścieżek jak i bez.

Wystarczy dodać jedną linię kodu implementującą sprawdzanie warunku przecięcia się ścieżek!

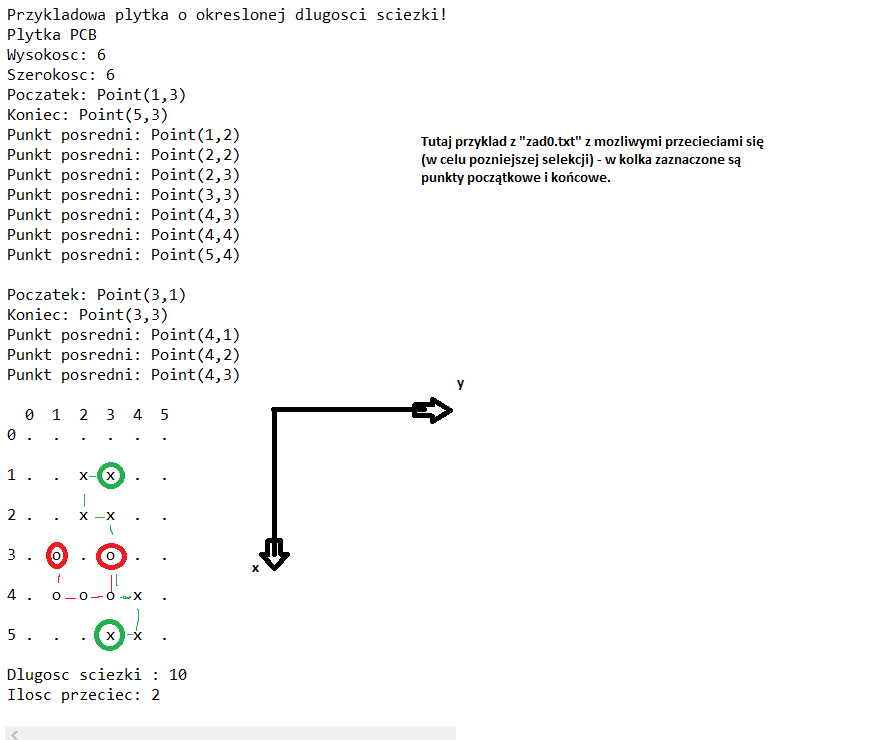
\*\*\* Uwaga\_2 \*\*\*

Poniższe przykłady tyczą się pliku z przykładowym problemem – plik z eportalu lab1\_problemy\_testowe.zip / zad0.txt

1. Przykład 1

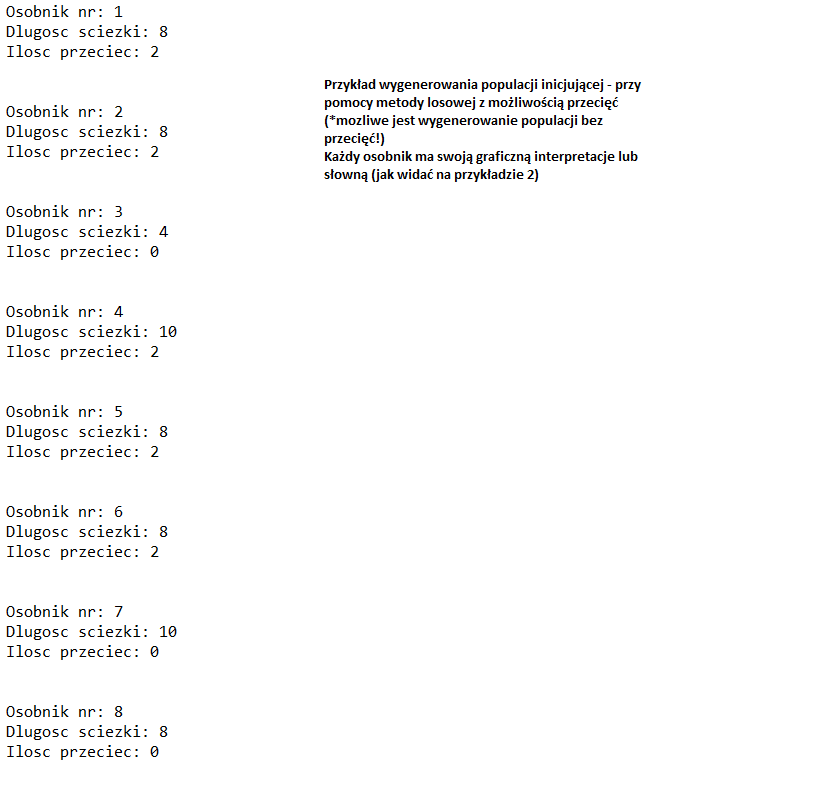


1. Przykład 2



Tutaj niektóre „x” są przysłonięte przez „o” – lecz jak spojrzymy na tekstową interpretacje przejścia zauważymy, że np. P(3,3) znajduje się w obu lutowaniach.

1. Przykład 3



Ograniczenia:

1) W danym osobniku - punkt nie może zostać powtórzony

Np. P(1,1) -> P(1,2) -> P(1,1) -> P(2,1) <<---- Jest to błędne, więc w implementacji nie pojawi się takie przejście

2) Dwa osobne osobniki w jednej płytce mogą się przeciąć ( w celu przyszłej selekcji ) - jednakże jest także zaimplemetowane ograniczenie -

metoda checkLimitations() w klasie PlytkaPCB - sprawdzana (bądź nie) jest ta metoda w metodzie losowej.

Przykład wyników:

Bez możliwości przecięć – przykład1

Z możliwością przecięć - przyklad2

Wszystkie pliki implementacyjne znajdują się w folderze Klasy\_Java.