**1. Tworzenie struktury programu**

* generowanie przystanków na osi współrzędnych - warunki, żeby nie były od siebie za blisko:

*Jednak są dwie funkcje - jedna generuje punkty i zwraca dikt (klucz to id, a wartość to (xy)) DIKT KOORDYNAT. // 1*

*Druga funkcja dostaje na wejscie coordynaty i zwraca tablice nxn z odległościami między każdym przystankiem - można to zrobić używając dicta dict (DIKT ODLEGŁOŚCI). Np w pierwszy klucz to id pierwszego punktu który ma jako wartość dikt z odległościami tego pierwszego punktu do każdego innego. // 1*

* zapisujemy w tablicy odległości między każdymi dwoma przystankami i wybieramy pary najbardziej odległych od siebie

*// do zastanowienia jak dokładnie*

*Będzie pobierać (?????) a zwracać liste idików przystanków które zostaną pętlami // 2/3*

**2. Generowanie pierwotnych połączeń:**

* liczba nitek (wektorów połączeń między pętlami) generowana losowo z przedziału <1, n(n-1)/2>
* wybieramy przykładowe połączenia między pętlami - losowo

*Upychamy to do jednej funkcji. Dostajemy listę z idikami pętli zwraca dikt gdzie kluczem będzie id nitki i para pętli (początkowej i końcowej) (DIKT LINII) // 1*

* losujemy długości wektorów - nitek i wybieramy losowo przystanki (ale koniecznie bez powtórzeń w danej nitce!) -> to tworzy pierwszego członka pierwszej generacji, jeśli f, celu jest zerowa dla niego (tzn - nie objął wszystkich przystanków) to powtarzamy wybieranie, robimy tak np 10 razy i to nam daje 10 członków pierwotnej populacji dla których będziemy sprawdzać i porównywać f. celu.

ANEKS DO TEGO:

W takim razie po rozmowie z Chmielem nie uwzględniam czy nitki objęły wszystkie przystanki. Dopisze funkcję która sprawdza ile przystanków nie zostało zawartych w żadnej nitce.

*Przekazujemy dikta z końcówkami pętli -DIKT LINII i klucze z DIKTU KOORDYNAT -nasze id przystanków. Zwraca DIKT (klucze to id nitki a wartości to listy przystanków) - DIKT NITEK. // 1*

*--------------------------------------------------------------------------------------------------------------------------------*

**3. Sprawdzamy i generujemy tablice ilości przesiadek między każdymi dwoma przystankami**

- używamy algorytmu Gołębiowskiego.

Liczymy sumę z kwadratów z macierzy trójkątnej z wartości tam wpisanych.

Alogrytm:

* mamy wypisane nitki w wektorach
* rozpisujemy macierz - na przekątnych zera
* wszykujemy pierwszy przystanek w pierwszym wektorze i we wszystkich innych wektorach też i przy wszystkich przystankach które znajdują się w tych wektorach zapisujemy w kolumnie A jako 0. Inne miejsca pozostają puste.
* zapisujemy symetrycznie do wierszy to samo
* robimy to dla kolejnego przystanku w pierwszej nitce. Kończymy i przechodzimy tak po każdej. W wyniku mamy tablice z zerami i miejscami pustymi.
* w kolejnym kroku sprawdzamy miejsca puste - jeśli na przykład dla AB jest puste to bierzemy dwa wiersze i sprawdzamy czy w jakiejś kolumnie mają gdzieś oba 0, jeśli tak to dla AB i BA wpisujemy 1. Robimy tak dla wszystkich miejsc pustych, jak nie znajdziemy to zostawiamy puste. W wyniku mamy tabele z X 0 1 lub miejscem pustym.
* zaczynamy od nowa dla miejsc pustych. Np jeśli w CD jest puste to sprawdzamy wiersze D i C i jeśli w jakiejś kolumnie przyjmują w sumie 1 - czyli mają 0 i 1 to zapisujemy w DC i CD 2. I tak dalej :D

*Dostaje DIKT NITEK (można mu też przekazać klucze DIKTU ODLEGŁOŚCI w liście na wykład zeby sobie zbudował tablice) i zwraca sumę kwadratów z wartości z tablicy czyli jedną liczbę. // 3*

**4. Funkcja celu**

* sprawdza czy są objęte wszystkie przystanki - warunek konieczny

*Dostaje DIKT NITEK i liste kluczy DIKTU ODLEGŁOSCI zwraca 1 lub 0 (true lub false) // 1*

* sumuje czas przejazdu przez wszystkie nitki (z wektorów i tablicy odległości między przystankami - liczymy odchylenie standardowe - zeby wszystkie nitki były mniej więcej takiej samej długości - minimalizujemy to :D

*To są dwie osobne. Dostaje DIKT ODLEGŁOŚCI i DIKT NITEK i sumuje długości wszystkich nitek. Zwraca liczbę. // 1*

*Druga funkcja dostaje to samo i liczy odchylenie standardowe i zwraca liczbę.*

*Obie wartości będą się wliczały do funkcji celu. // 1*

*---------------------------------------------------------------------------------------------------------------------------*

* obliczamy wartość tablicy przesiadek - minimalizujemy

// już wcześniej uwzględnione

Oceniamy wartości f celu i wybieramy najlepszych 5.

Tu się de facto tworzy funkcję celu która sumuje wszystkie wartości przemnożone przez współczynniki które będziemy potem doświadczalnie dobrać.

Przyjmuje te wartości i zwraca liczbę. // 3

|

V

Robimy takie maina który będzie tą całą funkcją EWOLUCJI. Najpierw będzie wywoływał te wszystkie funkcje inicjalizujące a potem przeprowadzał kolejne pokolenia i porównywał funkcje celu i usuwał i wariował i mutował :D AAA i uwzględniał warunki stopu (patrz punklt 6). //

**5. Dokonujemy wariacje między najlepszymi osobnikami**

* Jeśli w jakichś dwóch rodzicach pojawia się nitka o takich samych końcach i początkach - uwzględniamy ze nie mogą one znaleźć sie w tym samym osobniku potomnym. Wyrzucamy je z puli nitek i przyporządkowujemy do pierwszego i drugiego potomka.
* Losowo przydzielamy nitki z puli do osobników potomnych tak żeby liczba nitek danego dziecka była potomna
* Wprowadzamy mutacje jednego dziecka - losujemy mu od nowa losową liczbę nitek (pętle są bez zmian - albo jak chcemy :D)

*Wariująca - ostaje 5 najlepszych DIKTÓW NITEK RODZICÓW (w liście) i zwracała 5 nowych dzieci w postali DIKTÓW NITEK DZIECI // 3*

*Mutująca - dostaje losowy DIKT NITEK DZIECI i lista idkow pętli, zwraca DIKT NITKI nowego dziecka mutanta. // 3*

6. Warunki stopu

* jak któryś będzie miał dostatecznie niską f celu
* jak osiągnie ileś tam populacji (generacji)
* jeśli na przestrzeni np 10 generacji nie będzie znacznej zmieny między populacjami :D

Notatki z zajęć:

Dodać funkcję kary dla nieobjęcia wszystkich przystanków (niedopuszczalnych rozwiązań)

Modyfikacja wyboru rodziców (dodanie prawdopodobieństwa - wprowadzenie rankingu, czyli posortowania rozwiązań według wartości funkcji celu, a na podstawie rankingu przypisujemy prawdopodobieństwo)

Tworzenie nowej populacji z populacji potoków i populacji wstępnej rodziców:

1. Tworzenie populacji tylko na podstawie potomków - bezpośrednie wrzucenie dzieci do nowej populacji
2. Do jednego worka wrzucamy rodziców i dzieci i na podstawie ich wybieramy nową populacji na podstawie najlepszych z tego zbioru (dużo większa zbieżność) Żeby zrobic tą metoda to dla dzieci trzeba policzyć jeszcze funkcje celu!!!
3. Bierzemy część najlepszą z populacji rodziców i dodajemy do populacji dzieci tworząc nową populację

Co do mutacji to nie trzeba zamieniać nitek, można też całkowicie dodać albo usunąć nitkę

Można też wymienić dwie nitki między sobą (w sensie ich części zamienić na krzyż) - a ro bardziej krzyżowanie. Można dla mutacji wydłużyć jakąś albo skrócić, dodatnie jakiejś losowej do danego potomka

PZDR.