Funkcje:

1. **GenerateBusStops**

Funkcja generuje dict (słownik) przystanków, zgodnie z podanymi parametrami.

**przyjmuje** :

* amount - liczba przystanków które ma wygenerować
* max\_x - maksymalna współrzędna x
* max\_y - maksymalna współrzędna y
* dist - minimalną odległość między każdymi dwoma przystankami

**zwraca:**

* **DictCoordinate** - słownik przystanków, gdzie klucze zawierają ID każdego przystanku (iterowane od 1) a wartości zawierają tablicę dwuelementową współrzędnych x i y na płaszczyźnie

**uwagi:**

Funkcja korzysta z **CheckDistance** i sprawdza czy wylosowane współrzędne nowego przystanku spełniają wymagania co do odległości od już istniejących - czy leżą w nie bliżej niż zadany ‘dist’. Jeśli tak, nowy punkt zostaje dołączony do słownika.

1. **CheckDistance**

Sprawdza czy podane współrzędne są nie bliżej niż zadana odległość od każdego innego punktu w słowniku DictCoordinate.

**przyjmuje** :

* DictCoordinate - słownik istniejących przystanków
* x - nowo znaleziona wsp.
* y - nowo znaleziona wsp
* dist - minimalny dystans

**zwraca:**

* True/False - w zależności od wyniku sprawdzenia

**uwagi:**

Funkcja iteruja po wszyskich elementach słownika i sprawdza każdy element. Pierwsza napotkana niezgodność (za blisko) powoduje powrót z funkcji.

1. **PointsDistance**

Funkcja obliczająca odległość między dwoma punktami na podstawie ich współrzędnych (wykorzystywana w **GenerateDistanceDict** ).

**przyjmuje** :

* A - dwuelementowa lista pierwszego punktu
* B - dwuelementowa lista drugiego punktu

**zwraca:**

* odległość między nimi (liczba zmiennoprzecinkowa)

1. **GenerateDistanceDict**

Funkcja generująca słownik słowników odległości, gdzie kluczami ogólnego słownika są ID przystanków (iterowane od 1) a wartościami są słowniki, których klucze zawierają ponowne ID przystanków a wartości - odległości między nimi. Funkcja zwraca zatem coś na kształt tablicy dwuwymiarowej z obliczonymi odległościami między każdym punktem.

**przyjmuje** :

* DictCoordinate - słownik istniejących przystanków

**zwraca:**

* **DictDistance** - słownik słowników odległości

1. **ChooseTerminals**

Funkcja wybiera przystanki, które zostaną potencjalnymi pętlami.

**przyjmuje** :

* DictDistance - słownik słowników odległości
* rate - współczynnik (0,1), który zmniejsza pulę dostępnych do losowania przystanków, aby zmniejszyć szansę na wylosowanie za dużej liczby pętli // możliwe że do usunięcia

**zwraca:**

* LoopsId - listę zawierającą ID przystanków, które wezmą udział w losowaniu do par krańcowych danej nitki

**uwagi:**

Funkcja losuje z puli wszystkich przystanków (zmniejszoną odpowiednio współczynnikiem rate) ilość przystanków które zostaną wybrane jako kandydatki na pętle. Następnie dokonuje się losowanie kolejnych par najbardziej odległych od siebie przystanków - przechodząc kolejno po całym słowniku DictDistance. Następnie każda kolejna, znaleziona para najbardziej odległych od siebie punktów jest weryfikowana czy nie znajduje się już w tablicy LoopsId.

1. **GenerateLines**

Funkcja paruje pętle i zapisuje je w słowniku razem z id danej nitki (nitki są puste, mają początek i koniec)

**przyjmuje** :

* LoopsId - listę zawierającą ID przystanków, które wezmą udział w losowaniu do par krańcowych danej nitki

**zwraca:**

* **DictLines** - słownik par wylosowanych pętli, gdzie kluczami są ID przyszłych nitek a wartościami dwuelementowe listy z ID początkowych oraz końcowych przystanków.

**uwagi:**

Funkcja najpierw generuje listę wszystkich możliwych połączeń między pętlami - PossibPairs. Następnie losuje ile z nich ma zostać wybranych (ile będzie nitek). Ostatnim krokiem jest wylosowanie tych par, nadanie im ID i wpisanie do słownika.

1. **GenerateThreads**

Funkcja dobiera do poszczególnych nitek odpowiednie przystanki i ustawia je w kolejności, w jakiej ma przejechać autobus.

**przyjmuje** :

* DictLines - słownik par wylosowanych pętli
* DictCoordinate - słownik istniejących przystanków

**zwraca:**

* **DictThreads** - słownik pełnych nitek z przystankami i pętlami na brzegach

**uwagi:**

Funkcja przechodzi po kolejnych nitkach. Dla każdej z nich losuje ile będzie przystanków w i-tej nitce - BusStopAmount\_i (w ilosci jest odjęte 2 bo uwzględniamy, że pętle są już określone). Następnie tworzy się lista pomocnicza - helfulLIst z ID wszystkich istniejących przystanków - pętla iterowana po ‘p’. Dalej, z listy pomocniczej usuwane są przystanki zdefiniowane jako pętle w aktualnej nitce (nie będą one się już powtarzać wśród przystanków pośrednich danej nitki). Tworzona jest lista, w której pierwszym elementem staje się pętla początkowa - ListBus\_i. W forze iterowanym po ‘j’ następuje losowanie przystanków z listy pomocniczej, które są od razu umieszczane w ListBus\_i a usuwane z helpfulList, aby więcej się w danej nitce nie powtórzyły. Po zakończeniu przydzielania nitka otrzymuje na koniec ID pętli końcowej i trafia do słownika, na odpowiednie miejsce. Tak się dzieje z każdą kolejną nitką.

1. **AmountOfMissedBusStops**

Funkcja liczy ile przystanków nie znalazło się w rozwiązaniu - DictThread. Na podstawie jej wyniku, będzie uwzględniana kara do funkcji celu.

**przyjmuje** :

* DictThreads -słownik pełnych nitek z przystankami i pętlami na brzegach
* DictCoordinate - słownik istniejących przystanków
* DictLines - słownik par wylosowanych pętli

**zwraca:**

* missed - liczbę przystanków, które nie zostały zawarte w żadnej z nitek

**uwagi:**

Funkcja dla każdego przystanku ‘bst’ przechodzi po wszystkich nitkach ‘thr’ i sprawdza, czy pojawił się on w którejś z nich. Jeśli tak - inkrementuje zmienna missed. Na końcu missed jest odejmowana od wszystkich przystanków, co daje liczbę pomiętych.