Dokumentacja

Miłosz Dulewicz, Aleksandra Wasiak, Sabina Majewska

January 2023

1 Opis Problemu

Kurier każdego dnia musi dostarczyć paczki do wybranych lokalizacji i wrócić do bazy. Lokalizacje, które odwiedza kurier zmieniają się z dnia na dzień. W interesie firmy kurierskiej leży, aby trasa, którą pokonuje kurier pociągała za sobą jak najniższe koszty. Kurier powinien poruszać się więc po cyklu Hamiltona o najmniejszej sumie wag krawędzi. Mamy więc do czynienia z problemem komiwojażera, w którym codziennie należy rozpatrywać inny zestaw lokalizacji.

2 Sposób działania

Program stworzy graf pełny, którego wierzchołki są reprezentowane przez punkty na układzie współrzędnych, a wagi krawędzi to odległości między punktami. Następnie na takim grafie jest uruchamiany algorytm Christofides'a. Kolejne kroki algorytmu obejmują:

1. znalezienie, za pomocą algorytmu Kruskala, minimalnego drzewa rozpinającego MST
2. wybranie zbioru wierzchołków V drzewa MST, których stopnie są nieparzyste
3. wyznaczenie minimalnego skojarzenia doskonałego M na podzbiorze V zbioru wierzchołków oryginalnego grafu
4. stworzenie multigrafu zawierającego krawędzie grafów MST oraz M
5. znalezienie cyklu Eulera w powyższym multigrafie
6. zamianę cyklu Eulera na cykl Hamiltona, poprzez pomijanie powtarzających się wierzchołków.

Cykl Hamiltona znaleziony w algorytmie Christofides'a ma koszt nie większy niż 1.5 kosztu cyklu optymalnego.

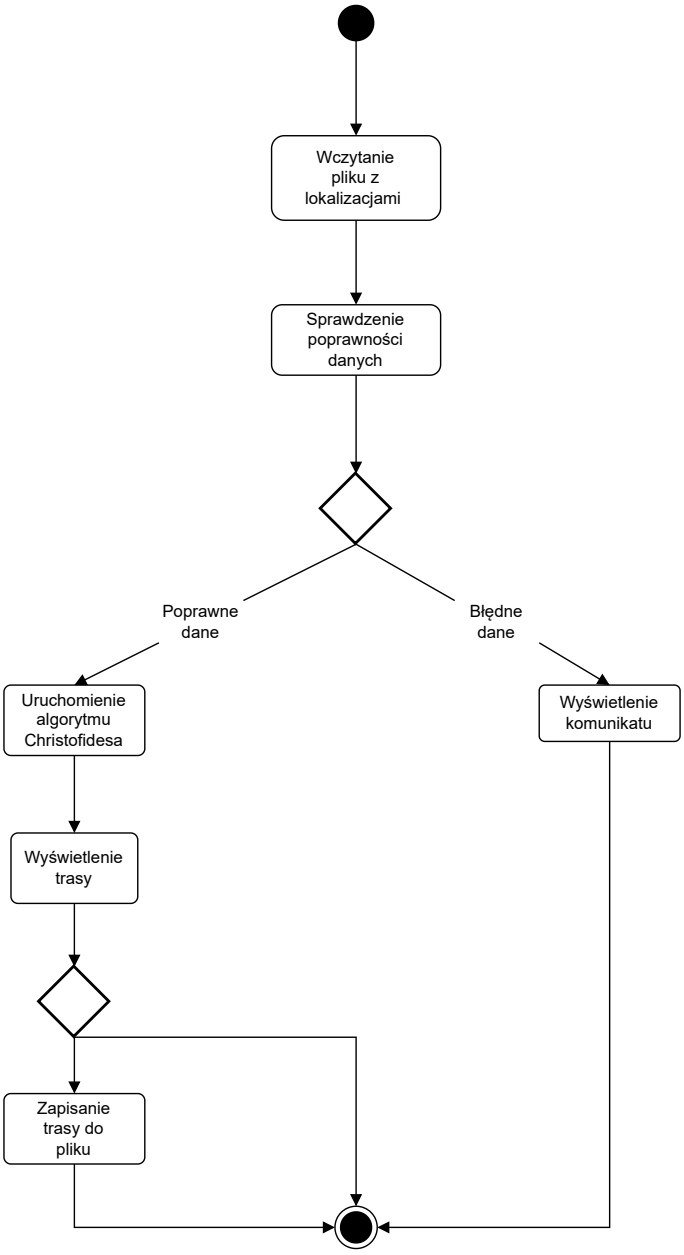

Figure 1: Diagram aktywności

3 Instrukcja

Aby znaleźć przybliżone rozwiązanie problemu komiwojażera dla grafu pełnego o wierzchołkach o współrzędnych $[x, y]$ należy:

- Odpowiednio zmienić plik "App/generate_graph_2.txt." W każdym wierszu są współrzędne jednego wierzchołka oddzielone spacją.
- Skompliować plik "App/app.py"
- Nacisnąć przycisk "Save christofides graph to file"
- Otworzyć plik "App/example_out.txt" w którym jest podane przybliżone rozwiązanie problemu.

4 Czas Wykonania

W pliku "Tests/time_it.py" można znaleźć skrypt wyliczający czas wykonania się algorytmu w zależności od wielkości danych. Dostaliśmy następujący output:

n	czas [s]
$n = 25$	0.011
$n = 50$	0.0539
$n = 100$	0.3501
$n = 150$	1.0666
$n = 200$	1.8941
$n = 300$	7.277
$n = 500$	38.4046
$n = 1000$	220.1227
$n = 1500$	1029

W pracy Christofides'a można wyczytać, że algorytm ma złożoność $O(n^3)$. Za pomocą metod graficznych przybliżyliśmy czas działania t naszego algorytmu w zależności od wielkości danych n wielomianem $t(n) \approx 10^{-6.5} n^3$.