Projekt AAL

Michał Pestka

Opis problemu

(Numer zadania 9)

Treść zadania:

> Firma kurierska Koniczynka próbuje określić maksymalną wagę paczek, którą może dostarczać w danym regionie. Ponieważ firma posiada pojazdy o imponującej masie dopuszczalnej, prezes zakłada, że głównym czynnikiem ograniczającym działalność są dopuszczalne na poszczególnych wagi pojazdów.
> Opracuj algorytm, który na podstawie mapy połączeń drogowych danego regionu wyznacza największy ciężar, który da się przetransportować pomiędzy dwoma dowolnymi miastami, przy zachowaniu ograniczeń.

Problem postawiony w zadaniu polega więc na stworzeniu specyficznego rodzaju drzewa rozpinającego.

Klasyczne drzewo rozpinające
Problem drzewa rozpinającego polega na znalezieniu sieci połączneń nie zawierającej cykli która łączy wszystkie wierzchołki, dodatkowo spełniając pewnie kryterium. Zwykle szukamy drzewa o jaknajniższej sumie wag krawędzi.

Moje drzewo rozpinające
W zadaniu chodzi o znalezienie drzewa rozpinającego które maksymalizuje wagę krawędzi o najniższej wadze.

Algorytm Kruskala

Okazuje się że Algorytm kruskala, który służy do znajdowania klasycznego drzewa rozpinającego, rozwiązuje problem dokładnie odwrotny do zadanego. Wystarczy więc zamienić znajdowanie najtańszej krawędzi na szukanie najdroższej.

Metody rozwiązania

Rozwiązanie zostało napisane w języku Python3. Dzieli się na 3 części:

- Cześć zawieracjąca algorytm
- Część zawierająca algorytm
- Część zawierającą strukturę danych - zbiór podzbiorów rozłącznych.

Algorytm rozwiązania

Zgodnie z tym, co zostało napisane powyżej, użyłem niznacznie zmodyfikowanego algorytmu Kruskala.

Algorytm Kruskala
Algortm Kruskala jest bardzo prosty. Kroki algorytmu:

- Utworzenie lasu (zbioru drzew) w którym każdy z wierzchołków jest osobnym drzewem.
- Posortowanie krawędzi
- Dla każdej krawędzi w zbiorze zaczynając od krawędzi o najmniejszej wadze:

- Wybranie pierwszej krawędzi oraz usunięcie jej ze zbioru krawędzi.
- znalezienie najwyższego parenta wierzchołka 1 w poddrzewie do którego należy (operacja FIND)
- znalezienie najwyższego parenta wierzchołka 2 w poddrzewie do którego należy (operacja FIND)
- Porównanie parenta wierzchołka 1 oraz 2:
  - Jeśli Parent(1) == Parent(2) -> nic nie robimy
  - Jeśli Parent(1) != Parent(2) -> łączymy drzewa ze sobą (opracja UNION)

Algorytmy generatora danych

Powstały dwa generatory danych.

Generator1
Zadaniem generatora 1 jest stworzenie zadania o zadanych parametrach którego wynik znamy.

Generator2
Generator drugi znajduje zastosowanie w profilowaniu algorytmu. Zasada działania opiera się na wylosowaniu macierzy gęstej. Po otrzymaniu macierzy losowej o rozmiarach podyktowanych przez zadane rozmiary grafu, obliczany jest punkt odcięcia x_{ths}. Wyznaczany jest tak, aby w macierzy znajdowała się określona liczba liczb mniejszych od x_{ths}. Ponieważ znamy rozkład liczb (jest on jednorodny), możemy ustalić X_{ths} "na oko", czyli tak żeby liczba liczb zgadzała się mniej więcej z zadaną wartością. Następnie zapewniana jest jednolitość grafu poprzez dodanie krawędzi.

Analiza złożoności

Niech zbiór E stanowi zbiór krawędzi, a zbiór V zbiór wierzchołków.

Złożoność poszczególnych kroków jest następująca:

- Sortowanie - $O(|E| * log(|E|))$
- dla każdej krawędzie ($|E|$ razy):
  - FIND - $O(log(n))$
  - FIND - $O(log(n))$
  - UNION - $O(1)$

Wynika z tego że całkowita złożoność algorytmu to:

$$O(|E|log(|E|)) + |E| * (O(1) + O(log(|V|)) = O(|E|log(|E|)) + O(|E|log(|V|))$$

Wiemy również że $O(|E|) = O(|V|^2)$ dlatego możemy stwierdzić że pesymistyczna zlożoność algorytmu to

$$O(|E|log(|E|))$$

Testowanie

Algorytm rozwiązanie był testowany za pomocą generatora zdolnego określić jaka powinna być wynikowa wartość programu. Zostały wygenerowanie 3 grafy. Zostały one następnie wrzucone do programu rozwiązującego.

Badanie

Zostały również przeprowadzone empiryczne badania złożoności obliczeniowej algorytmu. Badania te były bardzo proste i składały się z następujących kroków:

- wygenerowanie grafu
- Sprawdzenie liczby krawędzi oraz zapisanie jej
- Zapamiętanie obecnego czasu
- Rozwiązanie zadanego problemu
- Porównanie obeznego czasu z zapamiętaną wartością.

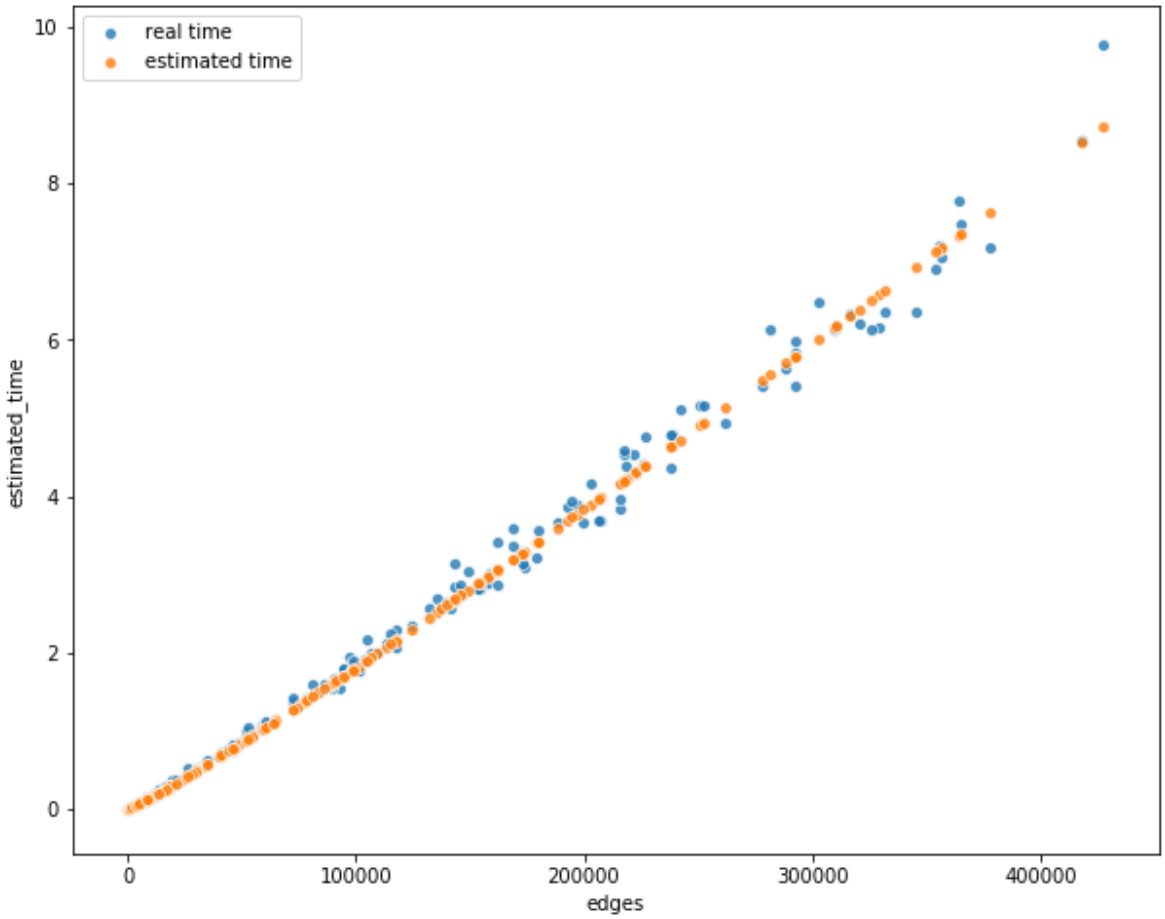
Dane:

liczba krawędzi	**czas**	**czas teoretyczny**
363	0.004564	0.003365
26809	0.433799	0.429867
44448	0.773114	0.748037
48406	0.852666	0.821141
77328	1.374956	1.368725
292220	5.983289	5.783288

Wyniki analizy:

Wnioski z analizy:
Empiryczna analiza złożoności pokazała że złożoność czasowa jest taka sama jak zakładana.

Wykres porównujący spodziewany czas z czasem uzyskanym:


real time
estimated time
estimated_time
edges
0
2
4
6
8
10
0
100000
200000
300000
400000