Maciej Poćwierz
242736

Analiza algorytmów

AAL-2-LS podróż

Koncepcja rozwiązania

Opis problemu

Problem polega na znalezieniu najtańszej opcji przejazdu pomiędzy wyróżnionymi miastami. Miasta są połączone siecią dróg. Kosztami obarczone są:

- Wjazd do każdego miasta.
- Wjazd na każdą drogę.
- Każda godzina podróży – czas przejazdu każdą drogą jest znany.

Opłata za wjazd do danego miasta nie jest pobierana, podczas przejazdu z miasta A do miasta B, jeśli istnieje co najmniej jedna grupa miast partnerskich, do której należą oba te miasta.

Założenia

- Dowolna liczba grup miast partnerskich.
- Każde miasto może należeć do dowolnej liczby grup miast partnerskich.
- Koszty wjazdu do każdego miasta oraz na każdą drogę są nieujemne.
- Każda droga łączy ze sobą dokładnie 2 miasta.
- Cena wjazdu na drogę i czas przejazdu jest taki sam, bez względu na kierunek podróży. Jednak cena przejazdu tej samej drogi może się różnić przez opłatę za wjazd do miasta.

Wprowadzanie danych wejściowych

Użytkownik powinien móc w wygodny sposób wprowadzać do aplikacji wybrane miasto źródłowe i docelowe. Identyfikatory tych miast będą zatem wczytywane z konsoli. Użytkownik nie powinien jednak wprowadzać danych dotyczących miast, dróg i grup miast partnerskich przy każdym uruchomieniu programu. Dane te będą zatem przechowywane w pliku tekstowym w formacie JSON, przykładowo o następującej strukturze:

```
"cities" : [
     {
        id="cityA",
        price=40
    },
    {
        id="cityB",
        price=30
    },
    {
        id="cityC",
        price=15
    }
]
```

```
"roads" : [
     {
        orig="cityA"
        dest="cityB",
        duration=60,
        price=20
    },
    {
        orig="cityB"
        dest="cityC",
        duration=40,
        price=15
    }
]
```

Koncepcja rozwiązania

Problem można rozwiązać z pomocą algorytmu Dijkstry, traktując miasta jako wierzchołki grafu, a drogi, jako jego krawędzie. Następnie wprowadzić należy funkcję kosztu, która dla dowolnych dwóch miast połączonych bezpośrednio drogą, wyznaczy koszt przejazdu pomiędzy nimi. Można zapisać tę funkcję jako:

$$\lambda(A,B)=\lambda_r(A,B)+\lambda_t(A,B)+\lambda_e(A,B) \quad ,$$

gdzie:

$\lambda(A,B)$ - koszt całkowity przejazdu z miasta A do miasta B.

$\lambda_r(A,B)$ - koszt wjazdu na drogę.

$\lambda_t(A,B)$ - koszt związany z czasem przejazdu.

$\lambda_e(A,B)$ - koszt wjazdu do miasta B – zerowy, jeśli A i B są miastami partnerskimi.

Szacowana złożoność rozwiązania

Złożoność czasowa

Złożoność czasowa algorytmu Dijkstry wynosi:

$O(V^2)$ - dla naiwnej implementacji z wykorzystaniem tablicy.

$O(E\,logV)$ - dla implementacji z wykorzystaniem kolejki, zaimplementowanej jako kopiec.

Gdzie: V – liczba wierzchołków, E – liczba krawędzi.

Zaawansowane rozwiązanie mogłoby polegać na uprzednim wyznaczeniu gęstości grafu i wybraniu na tej podstawie odpowiedniej implementacji. Implementacja naiwna jest optymalna dla grafów gęstych, a implementacja z użyciem kolejki dla grafów rzadkich. Nie wydaje się to jednak konieczne, jako że w rzeczywistości sieć dróg jest praktycznie zawsze grafem rzadkim. Zastosowany zostanie zatem wariant z kolejką priorytetową.

Wyznaczenie złożoności czasowej zaproponowanego rozwiązania jest jednak bardziej skomplikowane, gdyż należy dodatkowo uwzględnić złożoność czasową funkcji kosztu λ . Złożoność poszczególnych składowych tej funkcji wynosi:

λ_r - O(1), jeśli odpowiednie koszty przechowamy w tablicy mieszającej.

λ_t - O(1), jeśli odpowiednie koszty przechowamy w tablicy mieszającej.

λ_e - O(G), gdzie G jest liczbą grup miast partnerskich. Dla każdej grupy należy wykonać sprawdzenie, czy należy do niej miasto A i miasto B. Sprawdzenia te dokonywane są w czasie O(1), z użyciem tablic mieszających.

Zatem λ wykonuje się w złożoności $O(G)$.

Skoro zatem przy każdej relaksacji w algorytmie Dijkstry musimy dodatkowo wywołać funkcję o złożoności $O(G)$, to złożoność czasowa algorytmu wzrośnie do: $O(EG\,logV)$.

Złożoność pamięciowa

Na tym etapie dokładne określenie sposobu reprezentacji grafu w pamięci programu jest zadaniem trudnym. Można jednak przypuszczać, że konieczne będzie przechowywanie:

- V wierzchołków (miast).
- E krawędzi (dróg).
- G grup miast partnerskich, maksymalnie po V miast każda.

Zatem złożoność pamięciowa rozwiązania wyniesie: $O(V+E+GV)=O(E+GV)$.