Maciej Poćwierz
242736

Analiza algorytmów

AAL-2-LS podróż

Dokumentacja projektu

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.
- Wyjazd poza grupę miast partnerskich powoduje konieczność ponownego uiszczenia opłaty przy ponownym wjeździe do miasta z tej grupy.

Ostatnie z wymienionych założeń sprawia, że problem jest prostszy do rozwiązania, niż gdyby takiego założenia nie przyjęto. Założenie sprawia, że koszt przejazdu z daną drogą jest zawsze taki sam, bez względu na to, które miasta odwiedzono wcześniej. Można dzięki temu wykorzystać algorytm Dijkstry, który zakłada, że wagi krawędzi grafu są stałe.

Metoda rozwiązania

Problem rozwiązano za pomocą algorytmu Dijkstry, traktując miasta jako wierzchołki grafu, a drogi, jako jego krawędzie. Następnie wprowadzono funkcję kosztu, która dla dowolnych dwóch miast połączonych bezpośrednio drogą, wyznacza 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.

Ponieważ graf reprezentujący sieć dróg jest w praktyce prawie zawsze grafem rzadkim, zdecydowano się na implementację z użyciem kolejki priorytetowej. Ponieważ implementacja kolejki priorytetowej z biblioteki języka Scala nie umożliwia zmiany priorytetu elementu obecnego w kolejce, użyto implementacji bazującej na kopcu binarnym, dostępnej w repozytorium github.

Złożoność obliczeniowa:

W dokumentacji wstępnej projektu przedstawiono rozumowanie, zgodnie z którym złożoność obliczeniowa zaimplementowanego algorytmu wynosi: $O(EG\,log V)$, gdzie:

- V – liczba wierzchołków (miast).
- E - liczba krawędzi (dróg).
- G – liczba grup miast partnerskich.

Zgodność rzeczywistej złożoności algorytmu z przewidywaniami można potwierdzić korzystając ze specjalnego trybu działania aplikacji. W tym trybie aplikacja generuje losowe instancje problemu zwiększając stopniowo ich rozmiar. Następnie znajdywane są rozwiązania dla dziesięciu losowo wybranych par miast i wyznaczany jest średni czas obliczania pojedynczego rozwiązania. Dysponując danymi dotyczącymi czasów obliczeń dla instancji problemu o różnych wielkościach, aplikacja wyznacza specjalny współczynnik, umożliwiający ocenę zgodności przewidywanej i rzeczywistej złożoności. Przykładowy wynik działania tego trybu programu przedstawiono poniżej:

n (V/E/G)   / t(n) / q(n)

(250/3125/15) / 9.5 / 1.155025536755219 (265/3511/16) / 13.3 / 1.3352141384560632 (280/3920/16) / 13.6 / 1.210928270115737 (295/4351/17) / 12.4 / 0.9276103770247259 (310/4805/17) / 20.1 / 1.3497848114359765 (325/5281/18) / 18.9 / 1.0817358873018588 (340/5780/18) / 27.6 / 1.4321285387584313 (355/6301/18) / 29.7 / 1.4032755884268884 (370/6845/19) / 23.0 / 0.9410641406584783 (385/7411/19) / 38.4 / 1.4414860370943252 (400/8000/20) / 28.7 / 0.942089545572953 (415/8611/20) / 24.0 / 0.7274407259810515 (430/9245/20) / 38.8 / 1.0889658721108129 (445/9901/21) / 50.0 / 1.2409154819869215 (460/10580/21) / 44.7 / 1.0325675331723172 (475/11281/21) / 46.4 / 1.0	(490/12005/22) / 98.3 / 1.8907421787826013 (505/12751/22) / 66.6 / 1.2002232041697232 (520/13520/22) / 62.1 / 1.0505325251348567 (535/14311/23) / 73.7 / 1.1215446333482124 (550/15125/23) / 80.2 / 1.149716273424811 (565/15961/23) / 81.3 / 1.099750358949715 (580/16820/24) / 94.4 / 1.1564695012770554 (595/17701/24) / 59.0 / 0.6840741170101325 (610/18605/24) / 90.0 / 0.9889459078412887 (625/19531/25) / 93.0 / 0.9309953358437201 (640/20480/25) / 95.2 / 0.9055220745726283 (655/21451/25) / 132.6 / 1.1998685909112297 (670/22445/25) / 134.7 / 1.1608387580969592 (685/23461/26) / 155.0 / 1.224617798030296 (700/24500/26) / 137.2 / 1.0345822380383645

Jak widać nie występuje żadna korelacja między współczynnikiem q(n) a rozmiarem problemu. Wkazuje to, że złożoność problemu została oszacowana poprawnie.

Obsługa programu

Dostępne tryby uruchomienia i parametry opisano w pliku readme.txt. W pliku zawarto także przykładowe wywołania programu. Można też uruchomić program bez żadnych argumentów, w celu wyświetlenia pomocy.