Dokumentacja wstępna

Miasto Borgów

Autor:
Artur Wyrozębski

Problem:
Rozbudowywany jest rekurencyjnie acykliczny graf składający się początkowo z jednego wierzchołka. Rozbudowa polega na stworzeniu trzech duplikatów aktualnego stanu grafu i połączeniu tych duplikatów dwoma węzłami oraz pięcioma krawędziami, gdzie nowo dodany węzeł łączy się z drugim nowo dodanym węzłem oraz z dwoma duplikatami grafu. Każda krawędź ma własną wagę, która jest identyczna dla wszystkich nowych krawędzi w danym kroku powiększania grafu.
Celem jest obliczenie sumy odległości pomiędzy wszystkimi węzłami w grafie powstałym po krokach rozbudowy opisanych na wejściu.

Analiza problemu:
W każdym kroku rozbudowy liczba wierzchołków rośnie czterokrotnie oraz dodawane są dwa nowe wierzchołki, więc po ***n*** krokach liczba wierzchołków będzie wynosiła $|V| = (5/3)*4^n-(2/3)$ (rozwiązanie równania rekurencyjnego).
Liczba krawędzi natomiast jest równa liczbie wierzchołków pomniejszonej o jeden ($|E| = |V| - 1$), co wynika z faktu, że graf jest zawsze drzewem.
Zawsze występują maksymalnie trzy krawędzie na wierzchołek, bo nowo dodany wierzchołek w danym kroku zawsze łączy się z liśćmi zduplikowanych grafów oraz z drugim dodanym wierzchołkiem. Liść grafu ma tylko jedną krawędź, a po połączeniu będzie miał dwa krawędzie.
Wagi krawędzi każdego wierzchołka, który ma dwa krawędzie, są różnowartościowe. Inaczej jest w przypadku wierzchołków, które posiadają trzy krawędzie. U nich są to identyczne wartości.
W każdym grafie rozbudowanym jak w opisie problemu występuje symetria „względem środka grafu”.
Po obliczeniu sumy odległości w grafie dla dowolnego wierzchołka i potem dodaniu wierzchołka do tego grafu tak, aby był on liściem połączonym z tym wierzchołkiem, to suma odległości dla niego będzie równa sumie sumy odległości dla sąsiadującego wierzchołka oraz poprzedniej ilości wierzchołków pomnożonej przez wagę krawędzi między dodanym wierzchołkiem, a sąsiadującym wierzchołkiem ($S_{|V|} = S_{|V|-1} + (|V|-1)*W(|V|;|V|-1)$; S – suma odległości do reszty wierzchołków dla danego wierzchołka; |V| - liczba wierzchołków; W – funkcja zwracająca wagę krawędzi między jednym wierzchołkiem a drugim). Wynika to z faktu, że każda ścieżka nowo dodanego wierzchołka-liścia do innych wierzchołków prowadzi przez wierzchołek sąsiadujący.

Wstępna propozycja rozwiązania problemu:
Najpierw następuje całkowita rozbudowa grafu o określoną liczbę kroków.
Należy następnie ponumerować wszystkie wierzchołki i iterować od wierzchołka o numerze najmniejszym do tego o numerze największym.
W danej iteracji należy obliczyć sumę odległości od wierzchołka którego dotyczy iteracja do wierzchołków o numerach większych poprzez wykorzystanie algorytmu Depth First Search.
Przy przejściu z wierzchołka do następnego, którego się jeszcze nie odwiedziło, należy dodać wagę krawędzi do dotychczasowej odległości, która początkowo wynosi zero dla danej iteracji, a następnie dodać tę odległość do ich sumy, która również początkowo wynosi zero. Dodawanie odległości do sumy należy dokonywać tylko, gdy numer wierzchołka z którego się przeszło jest mniejszy niż numer wierzchołka będącego celem. Jeżeli z danego wierzchołka nie ma krawędzi do tych, których się nie odwiedziło, to należy odjąć wagę połączenia z poprzednim wierzchołkiem od dotychczasowej odległości i cofnąć się do niego.

Opis sposobu testowania rozwiązania problemu:
Do testowania poprawności algorytmu wykorzystuję testy jednostkowe, które mają sprawdzać poprawność działania poszczególnych funkcji algorytmu, grafu oraz rozbudowy grafu.
Do testowania czasu działania algorytmu oraz wymagań pamięciowych korzystam z narzędzi do profilowania. Jako, że używam języka Python, to tymi narzędziami są odpowiednio cProfile oraz memory-profiler.
cProfile pozwala na kalibrację profilowania, przez co ograniczany jest wpływ profilowania na wyniki czasu wykonywania algorytmu. Przedstawia dodatkowo wpływ czasowy poszczególnych funkcji na działanie algorytmu.
Memory-profiler analogicznie przedstawia wpływ pamięciowy poszczególnych funkcji algorytmu.

Analiza jakości rozwiązania problemu:
Złożoność czasowa algorytmu DFS jest liniowa względem liczby wierzchołków. Algorytm DFS jest w danej propozycji rozwiązania problemu wykonywany dla każdego wierzchołka. Z tych faktów wynika, że złożoność czasowa algorytmu wynosi $O(k) = k^2$, gdzie k jest liczbą wierzchołków.
Na podstawie treści problemu można zapisać równanie rekurencyjne $k_n = 4*k_{n-1} + 2$ – w tym przypadku n jest numerem kroku rozbudowy, a natomiast k jest liczbą wierzchołków, tym samym uzależniając liczbę wierzchołków od kroku rozbudowy. Rozwiązaniem tego równania rekurencyjnego jest $k_n = (5/3)*4^n-(2/3)$. Z niego wynika liczba wierzchołków po powiększeniu grafu n razy.
Dzięki temu rozwiązaniu można uzależnić złożoność czasową od liczby rozbudowań. Ta złożoność rozwiązania problemu wynosi $O(n) = (4^n)^2$.

Złożoność pamięciowa algorytmu jest uzależniona jedynie od algorytmu DFS – najdłuższej ścieżki jaką znajdzie. Tę najdłuższą ścieżkę można określić rekurencyjnym wzorem $a_n = 2*a_{n-1}+2$ (a to długość ścieżki, natomiast n to numer kroku rozbudowy). Rozwiązaniem tego równania jest $a_n = 3*2^n-2$. Ostatecznie wynika, że złożoność pamięciowa tego rozwiązania problemu wynosi $O(n) = 2^n$

Kres dolny złożoności czasowej można określić jako $O(n) = 4^n$. Wynika to z potrzeby budowy grafu, aby go przeanalizować. Jak wiadomo z treści problemu, trzeba 3 razy wykonać klonowanie w każdym kroku rozbudowy oraz dodać 2 wierzchołki do grafu. Złożoność czasowa tworzenia grafu do kroku n jest szeregiem $3*k_i+2$ od i=0 do i=n-1, k_i to liczba wierzchołków w kroku i. Poprzednio obliczono, że $k_i = (5/3)*4^i - (2/3)$. Z szeregu wynika więc, że $O(n) = 4^n$.