Sprawozdanie z ćwiczenia laboratoryjnego „Grafy”

Karolina Morawska

11 maja 2014

1 Czym wogóle jest graf ?

To taka struktura danych, która składa się z wierzchołków i krawedzi, przy czym poszczególne wierzchołki (zwane też węzłami) mogą być połączone krawędziami (skierowanymi lub nieskierowanymi) w taki sposób, iż każda krawędź zaczyna się i kończy w którymś z wierzchołków. Wierzchołki i krawędzie mogą być numerowane, etykietowane i nieść pewną dodatkową informację - w zależności od potrzeby modelu, do którego konstrukcji są wykorzystane. W porównaniu do drzew w grafach mogą występować pętle i cykle. Krawędzie mogą mieć wyznaczony kierunek (wtedy graf nazywamy skierowanym), mogą mieć przypisaną wagę (pewną liczbę), kolor, etykietę, np. odległość pomiędzy punktami w terenie, rodzaj połączenia.

Istnieje wiele rodzajów grafów, które mogą mieć wiele interesujących właściwości. Grafy mogą być, np.:

- Skierowane gdy możliwe jest przejście pomiędzy wierzchołkami tylko w jedną stronę (krawędź wtedy oznaczamy strzałką).
- Nieskierowane gdy możliwe jest przejście pomiędzy wierzchołkami w obydwie strony.

Naszym zadaniem było zaimplementowanie grafu nieskierowanego.

1.1 Opis implementacji grafu nieskierowanego wybranego przeze mnie.

Wybrałam implememtacje grafu za pomocą listy sąsiedztwa.Reprezentacja grafu za pomocą list sąsiedztwa jest podobna do reprezentacji macierzą sąsiedztwa. Mamy tablicę n-elementową, gdzie n oznacza liczbę wierzchołków w grafie. Każdy element tej tablicy jest skojarzony z jednym wierzchołkiem grafu - numer wiersza jest numerem wierzchołka. Elementy tablicy są listami. Listy te zawierają numery wierzchołków w grafie, do których prowadzi z danego wierzchołka krawędź.
Zaletą takiej implementacji jest:

- Oszczędność pamięci komputera, ponieważ odwzorywane są tylko istniejące krawędzie.
- Dostęp do sąsiadów danego wierzchołka jest szybszy niż w przypadku tablicy sąsiedztwa, ponieważ nie musimy sprawdzać kolejnych wierzchołków - lista od razu zawiera gotowych do odczytu sąsiadów.

2 Algorytmu służące do przeszukiwania grafu

2.1 Breadth-first search, czyli przeszukiwanie wszerz

Jest to jeden z najprostszych algorytmów przeszukiwań służacy do odnajdywania najkrótszej drogi w grafie. Przechodzenie grafu rozpoczyna się od zadanego wierzchołka i polega na odwiedzeniu wszystkich dostępnych z niego wierzchołków.

Złożoność pamięciowa algorytmu uzależniona jest od sposobu implementacji grafu.W moim przypadku czyli implementacji grafu za pomocą listy sąsiedztwa dla każdego wierzchołka przechowywana jest lista wierzchołków dostępnych bezpośrednio z niego.Złożoność pamięciowa wynosi $O(|V| + |E|)$ gdzie$|V|$ to liczba węzłów a $|E|$ to liczba krawędzi w grafie.

Ponieważ w najgorszym przypadku przeszukiwanie wszerz musi przebyć wszystkie krawędzie prowadzące do wszystkich węzłów, złożoność czasowa tego przeszukiwania wynosi$O(|V| + |E|)$, gdzie $|V|$ to liczba węzłów, a $|E|$ to liczba krawędzi w grafie.

2.2 Depth-first search, czyli przeszukiwanie w głąb

Polega na badaniu wszystkich krawędzi wychodzących z podanego wierzchołka.Po zbadaniu wszystkich krawędzi wychodzących danego wierzchołka algorytm powraca do wierzchołka, z którego dany wierzchołek został odwiedzony.

Złożoność pamięciowa algorytmu w przypadku drzewa jest o wiele mniejsza niż przeszukiwania wszerz, gdyż algorytm w każdym momencie wymaga zapamiętania tylko ścieżki od korzenia do bieżącego węzła, podczas gdy przeszukiwanie wszerz wymaga zapamiętywania wszystkich węzłów w danej odległości od korzenia, co zwykle rośnie wykładniczo w funkcji długości ścieżki.

Złożoność czasowa przeszukiwania jest uzależniona od liczby wierzchołków oraz liczby krawędzi. Algorytm musi odwiedzić wszystkie wierzchołki oraz wszystkie krawędzie, co oznacza, że złożoność wynosi $O(|V| + |E|)$.

3 Wyszukiwanie ścieżek w grafie

3.1 „Algorytm Dijkstry".

Aby wyznaczyć najkrótszą ścieżkę między s a wszystkimi innymi węzłami, w algorytmie Dijkstry wymaga się, aby w każdym węźle zapisywane były kolor i oszacowanie najkrótszej ścieżki. Początkowo wszystkim węzłom przypisujemy kolor biały, wszystkie oszacowania ścieżki ustawiamy na ∞ . Oszacowanie najkrótszej ścieżki dla węzła początkowego ustawiamy na 0. W miare działania algorytmu, wszystkim węzłom poza poczatkowym przypisujemy rodziców z drzewa najkrótszych ścieżek. Rodzic węzła może zmieniać się przed zakończeniem działania algorytmu wielokrotnie. Dalej algorytm działa nastepująco: najpierw sposród wszystkich białych węzłów grafu wybieramy węzeł u z najmniejszym oszacowaniem najkrótszej ścieżki. Wstępnie będzie to węzeł początkowy, którego ścieżka została oszacowana na 0. Po wybraniu węzła zaczerniamy go. Następnie, dla każdego białego węzła v przylegającego do u zwalniamy krawędź (u, v). Kiedy zwalniamy krawędź, sprawdzamy, czy przejście z u do v poprawi wyznaczoną dotąd najkrótszą ścieżkę do v. W tym celu dodajemy wagę (u, v) do oszacowania najkrótszej ścieżki do u. Jeśli wartość ta jest mniejsza lub równa oszacowaniu najkrótszej ścieżki do v, przypisujemy tę wartość v jako nowe oszacowanie najkrótszej scieżki i ustawiamy v jako rodzica u. Proces ten powtarzamy dotąd, aż wszystkie węzły będa czarne. Kiedy wyliczone zostanie już drzewo najkrótszych ścieżek, najkrótszą ścieżkę z węzła s do danego węzła t można wybrać poprzez przejście po tym drzewie od węzła t przez kolejnych rodziców, aż do s. Ścieżka o odwrotnej kolejności do uzyskanej jest ścieżką szukaną.

3.2 „Algorytm Forda-Bellmana"

Algorytm ten służy do wyznaczania najmniejszej odległości od ustalonego wierzchołka s do wszystkich pozostałych w skierowanym grafie bez cykli. Warunek braku cykli jest spowodowany faktem, że w grafie posiadajacym cykl najmniejsza odległość między niektórymi wierzchołkami jest nieokreślona, ponieważ zależy od liczby przejść w cyklu. Macierz A dla każdej pary wierzchołków u i v zawiera wagę krawędzi (u,v), przy czym jeśli krawędź (u,v) nie istnieje, to przyjmujemy, że jej waga wynosi nieskończoność. Algorytm Forda-Bellmana w każdym kroku oblicza górne oszacowanie S(vi) odległości od wierzchołka s do wszystkich pozostałych wierzchołków vi. W pierwszym kroku przyjmujemy S(vi)=A(s,vi). Gdy stwierdzimy, że S(v)>S(u)+A(u,v), to każdorazowo polepszamy aktualne oszacowanie i podstawiamy S(v):=S(u)+A(u,v). Algorytm kończy się, gdy żadnego oszacowania nie można już poprawić, macierz S(vi) zawiera najkrótsze odległości od wierzchołka s do wszystkich pozostałych.

Bibliografia

[1] $http://pl.wikipedia.org/wiki/Algorytm_Dijkstry,$

[2] $http://pl.wikipedia.org/wiki/Algorytm_Bellmana-Forda$

[3] $http://zasoby1.open.agh.edu.pl/dydaktyka/informatyka$

[4] $http://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz$

[5] $http://pl.wikipedia.org/wiki/Przeszukiwanie_w$

[6] $http://www.algorytm.org/algorytmy-grafowe/przeszukiwanie-grafu-wszerz-bfs-i-w-glab-dfs.html$