Grafy, czyli najszybsza droga do celu

Krzysztof Konieczny - nr albumu 130323

30 czerwca 2020, Politechnika Krakowska im. Tadeusza Kościuszki w Krakowie

Spis treści

1 Wstęp

Grafy to coś co kazdy z nas pamięta możliwie jeszcze ze szkoły podstawowej jako kółeczka, w których wpisywano cyfrę i nad strzałką podawało się znak działania i rozwiązywało poprzez wpisanie odpowiedniego wyniku w kółko obok. Jednakże te grafy, które będę chciał przedstawić w swojej pracy będą troszkę podobne do owych wspomnianych, lecz z bardziej szerokim ich zastosowaniem. W 1736 roku szwajcarski matematyk, Leonhard Euler rozwiązał za pomocą grafów zagadnienie mostów królewieckich, czyli takich które można przejść tylko jeden raz.

System GPS jest w obecnych czasach bardzo powszechnym urządzeniem, znajdującym się na większości przedmiotów, który pozwala na znalezienie danej osoby, rzeczy bądź najszybszej trasy. Pierwsze testy urządzeń nawigacyjnych odbyły się w 1977 roku jeszcze przed wprowadzeniem satelitów na orbity okołoziemskie. Próby i testy przeprowadzone zostały za pomocą pseudosatelitów, którymi były naziemne stacje nadawcze imitujące satelity. Jego działanie polega na pomiarze czasu dotarcia sygnału radiowego z satelitów do odbiornika. Gdy znamy prędkość fali elektromagnetycznej oraz dokładny czas wysłania danego sygnału to możemy obliczyć odległość odbiornika od satelity.

2 Cel pracy

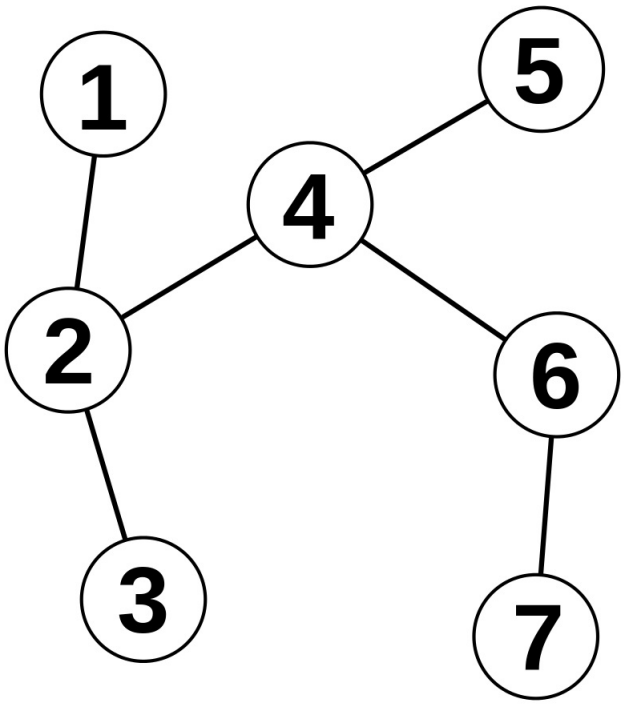
Głównym celem tej pracy jest pokazanie jak wykorzystać grafy do znalezienia najszybszej drogi w labiryncie w języku C++, który także umożliwi wykonanie owej czynności i pokażą w ten sposób jak owa funkja grafu może ułatwić pracę w dziedzinach takich jak: topografia, gry kompterowe czy też sieć.

Przedstawię również sam wygląd i krótki opis labiryntu, kod programu, a także wyniki, które pokażą czy program zadziałał i znalazł drogę bądź gdzieś się zgubił. Praca składa się z ośmiu rozdziałów, włącznie ze wstępem. Rozdział trzeci przybliży nam czym są grafy, jakie możemy wyróżnić, definicję, a także ich przykłady. Rozdział piąty zawiera nasze "sedno" całej pracy, w którym po krótce opiszę działanie i operację jakie będą wykonywane przez program. W rozdziale szóstym pokaże rezultat jaki otrzymamy po wykonaniu funkcji przez program. Następnie mamy krótkie podsumowanie i na końcu pracy znajduje się bibliografia.

3 Pojęcia grafów, definicje, przykłady

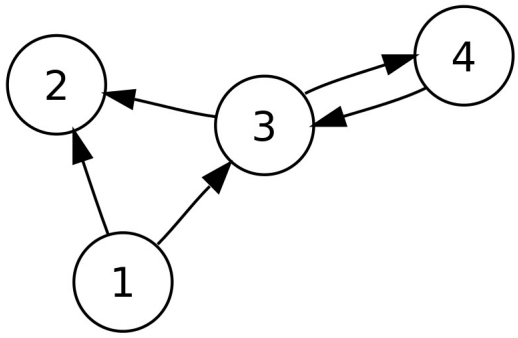
Grafem nazywamy strukturę złożoną z wierzchołków i krawędzi łączących owe wierzchołki. Mogą być one numerowane i czasem stanowią reprezentację jakichś obiektów, natomiast krawędzie mogą wówczas obrazować relacje między takimi obiektami. W notacji matematycznej graf oznacza się następująco:

- G = (V, E) - uporządkowana para wierzchołków i krawędzi
- V = 1, 2, ... n - skończony zbiór wierzchołków grafu
- E = $\{i, j\} : i \neq j \quad i \quad \{i, j\} \in V$ - zbiór krawędzi grafu



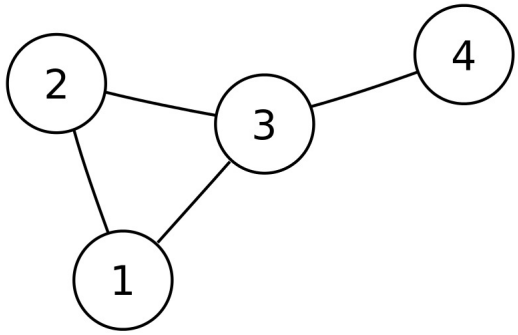
Rysunek 1: Przykład grafu

Graf skierowany lub inaczej digraf składa się z dwóch zbiorów – niepustego zbioru wierzchołków V oraz rodziny A par uporządkowanych elementów zbioru V, zwanych krawędziami lub łukami grafu skierowanego. Kolejność wierzchołków w parze wyznacza kierunek krawędzi – w przypadku pary v , u łuk biegnie z wierzchołka v do wierzchołka u.



Rysunek 2: Graf skierowany

Przeciwieństwem grafu skierowanego jest **graf nieskierowany.** Nie występują w nim połączenia, które zabraniają przejścia w którąś ze stron, są one "dwukierunkowe". Aby przejść z wierzchołka A do B wystarczy jedno połączenie między nimi. Można zauważyć, że jedno połączenie w grafie nieskierowanym, zastępuje dwa w grafie skierowanym. Warto pamiętać, że **z każdego grafu nieskierowanego można zrobić skierowany ale nigdy odwrotnie.**



Rysunek 3: Graf nieskierowany

Jedną ze struktur danych, umożliwiających przedstawienie skomplikowanego grafu lub jego przechowywanie w pamięci komputera, jest **macierz sąsiedztwa**, zawierająca dane na temat połączeń między wierzchołkami. Macierz jest rozmiaru V(G) na V(G), wyraz, leżący z i-tego wiersza i j-tej kolumny, zawiera wartość będącą liczbą krawędzi łączących x-owy i y-owy wierzchołek. Macierz sąsiedztwa pozwala na prezentację grafów, zawierających krawędzie wielokrotne oraz pętle własne. Do jej wad należy duża ilość potrzebnej pamięci (asymptotyczne tempo wzrostu) oraz fakt, że czas potrzebny do przejrzenia zbioru krawędzi jest proporcjonalny do kwadratu liczby wierzchołków zamiast do liczby krawędzi.

	I	II	III	IV	V
I	X	1	1	0	1
II	1	X	1	1	1
III	1	1	X	1	0
IV	0	1	1	X	1
V	1	1	0	1	X

Rysunek 4: Przykład macierzy sąsiedztwa

Graf może być reprezentowany również poprzez **listy sąsiedztwa.** Sposób ten polega na utworzeniu dla każdego wierzchołka listy zawierającej wszystkich jego bezpośrednich sąsiadów. W tej metodzie wierzchołki nie muszą być posegregowane.

I	II,III,V
II	I,III,IV,V
III	I,II,IV
IV	II,III,V
V	I,II,IV

Rysunek 5: Przykład listy sąsiedztwa

4 Zastosowanie

Grafy służą do praktycznych zastosowań w wielu różnorodnych problemach, **np.: sieć dróg z wierzchołkami reprezentującymi skrzyżowania i krawędziami przedstawiającymi ulice, sieć pomieszczeń i korytarzy w budynkach**, dzięki czemu możliwe jest komputerowe wynajdywanie najlepszej drogi ze swojego obecnego położenia do pożądanego celu.

Algorytmy grafowe stanowią istotną część programów obsługujących urządzenia GPS. **System sztucznej inteligencji w niektórych grach komputerowych musi odszukać dla sterowanych przez program postaci najlepszą drogę, która pozwoli zaatakować przeciwnika.** Przedstawienie w formie grafów sieci komputerowych pozwala na stworzenie oprogramowania usprawniającego trasowanie w Internecie.

W dużych organizacjach realizację zlecanych przez klientów zadań można przedstawić w postaci grafów, dzięki czemu możliwe jest zwiększenie wydajności: **wierzchołki mogą reprezentować pracowników, a krawędzie grafu – przepływ zadań.**

Za pomocą związanych z grafami pojęć można opisywać też m.in. **rysunki obwodów, schematy blokowe, ponieważ przedstawiają one połączenia lub relacje, zachodzące między różnymi fragmentami wykresu.**

Drzewa grafowe przydatne są w praktycznej reprezentacji różnego rodzaju hierarchii. Mistrzostwa sportowe czy drzewo genealogiczne mogą być w przejrzysty sposób opisane przez drzewa binarne.

W projekcie wykorzystamy właśnie wspomniane funkcję grafów, by znaleźć najszybszą możliwą drogę ucieczki z naszego labiryntu.

5 Szukanie drogi w labiryncie

Przechodzimy teraz do meritum całej pracy, której celem jest wyjście z labiryntu przez algorytm napisany w języku C++. Skupię się tutaj głównie na opisie działania kodu, co robi i jak wygląda labirynt - z czego się składa. Natomiast samym kodzie zawarłem także komentarze ospisujące w ważnych miejscach co będzie robił program. Kody programów dołączę w notatniku w celu weryfikacji.

Przejdźmy zatem do krótkiego opisu labiryntu. Będziemy go reprezentować w postaci **dwuwymiarowej tablicy znaków ASCII o rozmiarze m wierszy na n kolumn.** Tablica będzie zawiera następujące znaki:

- # - ściana
- . - korytarz
- S - start
- W - wyjście
- * - koniec danych

Każdy znak tablicy będzie odpowiadał **"wierzchołkowi" grafu.** Znak "#" to **wierzchołek izolowany**, do którego nie prowadzą **żadne** krawędzie. Pozostałe znaki oznaczają wierzchołki i będą połączone ze sobą krawędziami, które istnieją jedynie pomiędzy sąsiadującymi ze sobą wierzchołkami. Te z kolei sąsiadują ze sobą jedynie **w poziomie i w pionie** (a nie np. po przekątnej). Sama informacja o istnieniu krawędzi jest zawarta wewnątrz tablicy znakowej. Zatem do sąsiedniego wierzchołka możemy przejść, **jedynie gdy zawiera on znak korytarza, czyli kropkę.** Jeżeli jest to inny znak, przejście w tym kierunku jest **niemożliwe.**

Do wyszukania ścieżki wykorzystamy **przejście BFS**, które gwarantuje znalezienie najkrótszej ścieżki. Zasada pracy algorytmu polega na tym, iż po odczytaniu tablicy, szukamy w niej położenia znaków S i W. Położenie to zapamiętujemy w odpowiednich zmiennych. Znak S pozostawimy bez zmian, natomiast znak W zastąpimy znakiem korytarza "kropką", aby algorytm wiedział że ma podążać w tym kierunku.

Następnie uruchomiamy algorytm do rozpoczęcia szukania ścieżki przejściem BFS – kolejka będzie przechowywała współrzędne kolejno odwiedzanych wierzchołków, czyli numery wiersza i kolumny, w których znajduje się przetwarzany wierzchołek/znak labiryntu. Na początku w kolejce umieszczamy współrzędne znaku S znalezione na początku. Wyszukiwanie będzie trwać, aż algorytm dojdzie do współrzędnych znaku W lub wyczerpie wszystkie dostępne przejścia.

W trakcie przeglądania labiryntu algorytm umieszcza w komórkach tablicy informację o tym, skąd do danej komórki przyszedł. Opisane jest to za pomocą jednej z czterech małych liter:

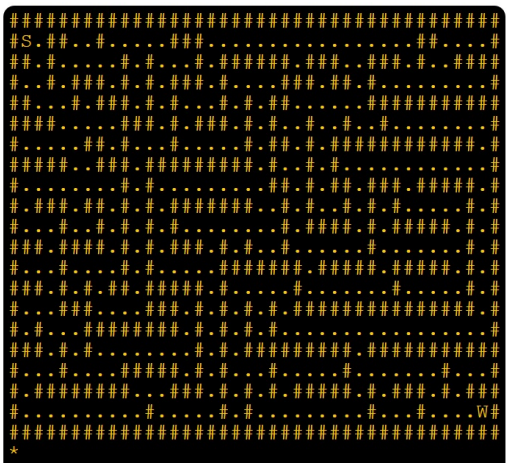
- g - ze znaku leżącego w wierszu powyżej
- d - ze znaku leżącego w wierszu poniżej
- l - ze znaku leżącego po lewej stronie
- p - ze znaku leżącego po prawej stronie

Gdy wyszukiwanie ścieżki się zakończy, to algorytm sprawdza zawartość lokacji w tablicy, która początkowo zawierała znak W (który został zastąpiony przez znak korytarza "kropkę"). **Jeśli w tym miejscu mamy wciąż "kropkę", to algorytm nie odnalazł ścieżki.** Zatem program wraca się po znakach g, d, p, l do pozycji S. Aby pokazać ścieżkę jaką przebył zastąpiłem kropki znakiem "+".

Końcowo usuwamy z tablicy pozostałe znaki kierunków, odtwarzamy znak W na jego pierwotnej pozycji i wyświetlamy wynik, który przedstawię w następnym rozdziale.

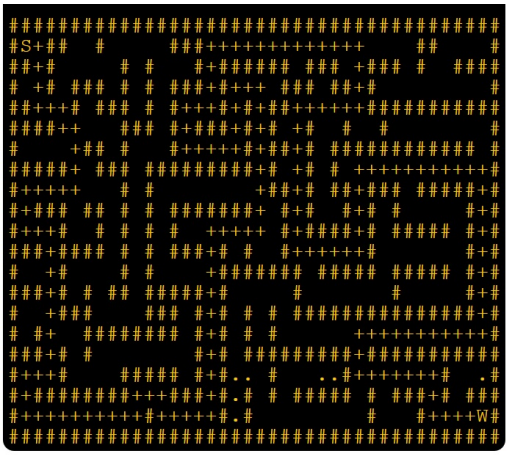
6 Podgląd otrzymanego wyniku

W tym rozdziale przedstawię, za pomocą rysunków, wynik jakie uzykał program. Na początku przedsatwiam na rysunku poniżej wygląd naszego labiryntu przed wykonaniem operacji poszukiwania ścieżki wyjścia:



Rysunek 6: Wygląd naszego labiryntu przed znalezieniem drogi ucieczki

Natomiast tak prezentuje się efekt wykonanej przez program pracy:



Rysunek 7: Program znajduje najszybszą drogę ucieczki, którą oznaczyłem znakiem "plus"

7 Podsumowanie

Końcowo jak możemy zobaczyć w rozdziale poprzednim, założony cel aby znaleźć drogę do wyjścia z labiryntu przebieł pomyślnie. Praca ta poruszyła również wiele zagadnień dotyczących grafów jak i sam algorytm przybliżył obraz na to, jak duży wpływ mają grafy w naszym życiu codziennym - mowa tutaj o systemie GPS, dzięki któremu bez problemu możemy znaleźć najszybszą trasę do punktu naszego celu - jak i w dziedzinach informatyki - rozwój gier; logistyki - nowe sieci połączeń dróg, linii kolejowych, linii metra; matematyki, topologii itp i wciąż będzie się rozwijać co może skutkować w przyszłości sporym postępem technologicznym.

8 Bibliografia

- http://fizyk.ifpk.pk.edu.pl/~rkycia/master/MW.html
- https://eduinf.waw.pl/inf/alg/001_search/0128.php
- https://pl.wikipedia.org/wiki/Graf_(matematyka)
- https://pl.wikipedia.org/wiki/Global_Positioning_System