Lidia Sobońska 310903
Arkadiusz Kojtek 310746

AISDI

Grafy

Sprawozdanie

Oświadczamy że poniższa praca, będąca podstawą do ocenienia wyników laboratorium z AISDI została wykonana przez nas samodzielnie.

Biblioteki wykorzystane w projekcie

Do stworzenia naszego projektu korzystaliśmy z następujących modułów:

1. **sys** służący do przyjęcia informacji o pliku źródłowym oraz do utworzenia wartości reprezentującej maksymalną wartość unsigned int..
2. **random**, który posłużył do wygenerowania planszy z losowymi ścieżkami

Algorytm tworzenia najkrótszej ścieżki

W pliku **dijkstra.py** znajdują się funkcję odpowiedzialne za odczytanie danych, zapisanie ich oraz znalezienie najkrótszej ścieżki.
generate_random_map(size) - tworzy planszę z losowymi wartościami oraz zwraca w formie krotki informacje potrzebne do działania algorytmu, czyli planszę, wymiar oraz pozycje obu zer.
read_from_file(file_name) - generuje te same informacje co **generate_random_map()** ale zamiast losować planszę przyjmuje jej wartości z pliku o nazwie podanej jako argument
prepare_dictionary(values_list, size) - generuje i zwraca słownik na podstawie planszy, który reprezentuje każde pole mapy wraz z jego sąsiadami oraz kosztami dostania się do nich.
shortest_path(start,end,map,distance,parents) - znajduje najkrótszą ścieżkę pomiędzy **start** i **end** jednocześnie uzupełniając lub edytując słownik **parents** który odpowiada temu sąsiadowi danego węzła przez który poprowadzona zostanie najkrótsza ścieżka do **starts**.
create_path(**start,end,shortest**) - generuje listę elementów na podstawie zaktualizowanego słownika **parents**, które reprezentują pola które należą do najkrótszej ścieżki.
create_map(path,values,size) - generuje nową planszę przedstawiającą najkrótszą ścieżkę na podstawie listy elementów ścieżki, starej planszy oraz jej rozmiaru.
find_path(data) - zajmuje się obsługą generowania najkrótszej ścieżki na podstawie krotki zawierającej pozycję zer, rozmiar i planszę.
get_map_str(data) - z krotki zawierającej pozycję zer, rozmiar i planszę w postaci listy tworzy stringa przedstawiającego planszę w celu wypisywania
open_and_write_to_file(file_path, data) - zapisywanie stringu ‘data’ do pliku pod wskazaną ścieżką

Podział prac

Obsługa plików, tworzenie i odczytywanie map, zapisywanie i wczytywanie danych zostało napisane przez **Lidię Sobońską,** tworzenie ścieżki na podstawie danych zostało napisane przez **Arkadiusza Kojtek**

Sposób uruchomienia

Plikiem obsługującym projekt jest **dijkstra.py**. Uruchomienie projektu polega na wywołaniu pliku **dijkstra.py** z poziomu terminala z dopisanym opcjonalnym argumentem z ścieżką do pliku zawierającego planszę. W przypadku, gdy nazwa pliku z danymi nie zostanie podana, wygenerowana zostanie losowa plansza o wymiarach 6x6.

Przebieg działania programu

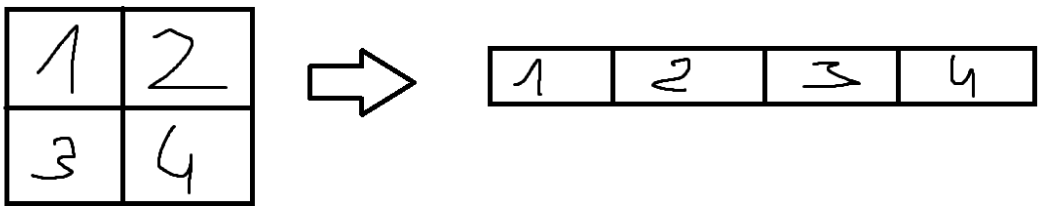
Jeżeli został podany plik wejściowy, to program wczytuje planszę i znajduje jej podstawowe dane (rozmiar, położenie zer). W przeciwnym wypadku generowana jest losowa plansza, która zapisywana jest do pliku i wypisywana na ekran. Następnie puszczany jest algorytm wyszukujący najkrótszą ścieżkę, która jest następnie zapisywana do pliku i wypisywana na ekran.

Pliki wynikowe

Jako plik wynikowy zostanie utworzony plik:
sciezka.txt - znaleziona najkrótsza ścieżka
plansza.txt - jeżeli plik wejściowy nie został podany, to wygenerowana losowo plansza zostanie zapisana w tym plik

Analiza działania programu (algorytm)

Zaimplementowany został algorytm Dijkstry.
Algorytm przyjmuje listę reprezentującą pola bez podziału na wiersze tzn. Plansza jest reprezentowana jako ponumerowane od lewej do prawej, od góry do dołu pola początkowej mapy.



Następnie na podstawie przyjętej listy generowany jest słownik reprezentujący każde pole wraz z kluczami jego sąsiadów i przypisanymi kosztami:
Dictionary = {Pole1:{Pole2: kosztPole2, Pole3:kosztPole3}, Pole2:{Pole1:kosztPole1, Pole4:kosztPole4} itd...}
Generowanie słownika opiera się na sprawdzaniu sąsiedztwa gdzie wiadomo, że:
Sąsiad z lewej jest polem o 1 mniejszym, musi być polem większym lub równym zero, oraz aktualne pole nie może być podzielne przez wymiar planszy.
Sąsiad z góry jest polem mniejszym o wymiar oraz musi być większy równy zero.
Sąsiad z prawej jest polem o 1 większym, musi być polem mniejszym od kwadratu wymiaru oraz reszta z dzielenia aktualnego pola musi być różna od wymiar - 1
Sąsiad z dołu jest większy od aktualnego pola o wymiar planszy oraz musi być mniejszy od kwadratu wymiaru planszy.

Następnie tworzony jest pomocniczy słownik zawierający koszt ścieżki do pola startowego (pole z pierwszym zerem). Dla ułatwienia wartość kosztu każdej ścieżki jest ustawiana jako maksymalna wartość unsigned int (zakładając, że koszt ścieżki nigdy nie przekroczy tej wartości) oprócz wartości ścieżki do pierwszego zero która jest ustawiana na zero.

Następnie generowany jest pusty słownik **parent_path**, w którym przechowywane są klucze każdego pola wraz z kluczem sąsiada przez którego poprowadzona może zostać najkrótsza ścieżka do pola startowego.
Na przykład załóżmy, że najkrótsza ścieżka z **1** do **4** zostaje poprowadzona przez **2.**
Powinien zostać utworzony następujący słownik:
parent_path = {4:2, 2:1}

Na podstawie słownika **parent_path** generowana jest list pól **shortest** które reprezentują najkrótszą ścieżkę.

Z wykorzystaniem tej listy tworzona jest nowa mapa, która dla pól nieznajdujących się w **shortest** wstawia pusty znak **“ “** a dla pozostałych wstawia wartość tego pola.

Przykłady działania programu