Zaawansowane algorytmy
Projekt 3
Wyszukiwanie najkrótszej ścieżki do wyjścia z labiryntu.

Adrian Świderek
367867
Grupa 23

Treść zadania:

Na wejściu w kolejnych liniach dostajemy labirynt opisany w następujący sposób:
O - oznacza korytarz
X - oznacza ścianę
S - oznacza punkt startowy
W - oznacza wyjścia (może być więcej niż jedno)

Określ, czy z punktu startowego da się dotrzeć do wyjścia. Jeśli się da, to wypisz długość najkrótszej ścieżki. Znajdź minimalny las rozpinający dla podanego na wejściu labiryntu.
Dokładny opis wejścia (Przyjmujemy, że linie są stałej długości):
ilosc_lini dlugosc_linii
....
LABIRYNT
....
Przykład:
4 8
XXSXXXXX
XXOOOXXX
WOOXOOOW
XXXWXXXX

Odpowiedź: TAK 4

Opis działania algorytmu:

Program został napisany w javie. Dane z pliku w którym zawarty jest labirynt oraz ilość_linii i długość_linii pobrałem za pomocą klasy Scanner odnajdując dwie liczby całkowite oraz następujące po nich linie tekstu. Znaki przedstawiające labirynt przekształciłem na macierz 0 oraz 1 (0 została oznaczona niemożliwa do przejścia ściana a reszta została oznaczona 1, współrzędne wejścia oraz wyjść zostały również zapisane w oddzielnych punktach). W programie utworzyłem dwie pomocnicze klasy czyli „Point" przechowująca współrzędne punktu (x i y) oraz „Node" przechowującą punkt oraz przebytą odległość. Utworzona została również funkcja walidująca która sprawdza czy dana współrzędna punktu jest większa lub równa od zera, czy nie wykracza poza liczbę wierszy/kolumn oraz czy nie była wcześniej odwiedzana. Następnie przechodzimy do właściwej funkcji szukającej ścieżek która za argumenty przyjmuje wcześniej opisaną macierz, punkt wejścia oraz punkt wyjścia. W funkcji tej na początku oznaczamy nasz punkt wejścia jako odwiedzony, tworzymy kolejkę węzłów i dodajemy do niej nasz węzeł początkowy oraz przebyty „dystans" czyli 0. Następnie zaczynamy w pętli przeszukiwać naszą kolejkę pobierając pierwszy z elementów funkcją poll() (element zostanie usunięty). Jeżeli pobrany z kolejki punkt okaże się wyjściem dodaję przebyty do tej pory „dystans" do listy i opuszczam pętlę. Jeśli nie to przechodzę do kolejnej pętli - iterowanej do 4 ponieważ tyle jest możliwych ruchów z danego punktu – w górę, w dół, w prawo, w lewo i sprawdzam czy dany ruch jest możliwy wcześniej opisanym walidatorem. Jeśli jest możliwy to dodaję węzeł zawierający punkt (po danym ruchu) i zwiększony o jeden dystans do kolejki. Po tym rozpoczyna się kolejny przebieg pętli. Funkcję tą wywołuję tyle razy ile jest wyjść z labiryntu a następnie wybieram najkrótszy możliwy dystans.