Sprawozdanie z miniprojektu 4

Mateusz Kamieniecki

SPRAWOZDANIE	**Data wykonania: 28.12.2025**
Tytuł miniprojektu:	*Labirynt*
Wykonał:	*Mateusz Kamieniecki*
Sprawdził:	*dr inż. Konrad Markowski*

Spis treści

1 Cel Projektu **1**

2 Rozwiązanie Problemu **1**
2.1 Tworzenie losowego labiryntu 1
2.1.1 Algorytm DFS 2
2.1.2 Algorytm Kruskala 2
2.1.3 Algorytm Wilsona 2
2.2 Znajdywanie Najkrótszej Ścieżki 2
2.3 Znajdywanie Pozostałych Ścieżek 2

3 Szczegóły Implementacji **2**
3.1 Reprezentacja Labiryntu w Ascii 2
3.2 3

1 Cel Projektu

Celem projektu było stworzenie lowosego labiryntu w którym można dostać się do każdego wierzchołka, zrobioć z niego graf, znaleźć najkrótszą ścieżkę między początkiem i końcem labiryntu oraz znaleźć pozostałe ścieżki.

2 Rozwiązanie Problemu

2.1 Tworzenie losowego labiryntu

Aby stworzyć losowy labirynt jest kilka różnych algorytmów. Ja do tego problemu wybrałęm generację za pomocą dfs ale krótko opiszę na czym polegają. Po zakończeniu algorytmu dfs losowo są niszczone ściany, aby stworzyć więcej ścieżek w labiryncie.

2.1.1 Algorytm DFS

Algorytm DFS (Depth-First Search) polega na tym, że zaczynamy od losowego punktu i idziemy w losowym kierunku aż do momentu, gdy nie możemy iść dalej. Wtedy wracamy się do ostatniego punktu, gdzie mieliśmy jeszcze nieodwiedzone sąsiednie komórki i kontynuujemy. Powtarzamy ten proces aż odwiedzimy wszystkie komórki. Ten algorytm ma tendencję do tworzenia długich, krętych ścieżek.

2.1.2 Algorytm Kruskala

Algorytm Kruskala polega na traktowaniu każdej ściany labiryntu jako krawędzi w grafie. Losowo wybieramy krawędzie i usuwamy je, jeśli usunięcie nie spowoduje powstania cyklu w grafie. Powtarzamy ten proces, aż wszystkie komórki będą połączone.Ten algorytm tworzy labirybty, które mają dużo krótkich ścieżek i rozgałęzień.

2.1.3 Algorytm Wilsona

Algorytm Wilsona polega na tworzeniu losowych ścieżek z nieodwiedzonych komórek do już odwiedzonych komórek. Gdy ścieżka napotka odwiedzoną komórkę, łączymy ją z labiryntem. Powtarzamy ten proces, aż wszystkie komórki będą odwiedzone. Ten algorytm tworzy labirynty z równomiernym rozkładem ścieżek.

2.2 Znajdywanie Najkrótszej Ścieżki

Początkowo używałem tego samego algorytmu DFS do znajdywania wszystkich ścieżek oraz najkrótszej ścieżki. Jednak DFS nie jest optymalnym wyborem do znajdywania najkrótszej ścieżki w grafie o równych wagach krawędzi, ponieważ może prowadzić do eksploracji długich ścieżek zanim znajdzie krótszą. Zamiast tego, lepszym wyborem jest algorytm BFS (Breadth-First Search). BFS eksploruje wszystkie sąsiednie wierzchołki na danym poziomie przed przejściem do następnego poziomu, co gwarantuje znalezienie najkrótszej ścieżki w grafie o równych wagach krawędzi.

2.3 Znajdywanie Pozostałych Ścieżek

Do znajdywania wszystkich ścieżek między dwoma punktami w labiryncie użyłem algorytmu DFS (Depth-First Search). DFS jest odpowiedni do tego zadania, ponieważ pozwala na eksplorację wszystkich możliwych ścieżek poprzez rekurencyjne przechodzenie przez każdy możliwy kierunek z aktualnego wierzchołka, aż do momentu dotarcia do punktu docelowego lub wyczerpania wszystkich opcji.

Ponieważ liczba wszystkich możliwych ścieżek może być bardzo duża, postanowiłem wprowadzić limit na maksymalną liczbę ścieżek do znalezienia. W mojej implementacji ustawiłem limit na 1000 ścieżek.

3 Szczegóły Implementacji

3.1 Reprezentacja Labiryntu w Ascii

Ciężko jest przedstawić ściany labiryntu oraz pole w jednym znaku dlatego postanowiłem rozszerzyć labirynt, aby oddzielne komórki miały swoje ściany. Labirynt o rozmiarze NxN

jest reprezentowany jako tablica o rozmiarze (2N+1)x(2N+1).

3.2