Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Dokumentacja projektu zespołowego

Wizualizacja wyszukiwania drogi w labiryncie

Filip Stępień Nr indeksu: 094117
Rafał Grot Nr indeksu: 094046

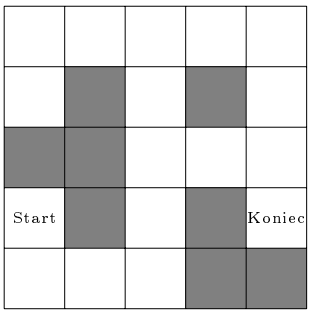
Informatyka, grupa 3ID11B

26 maja 2025

Spis treści

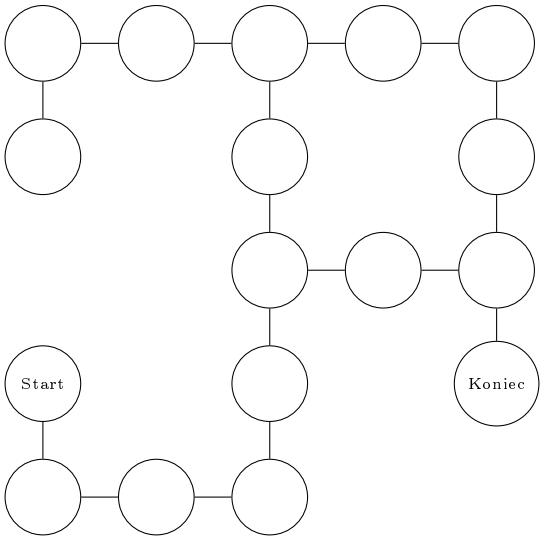
1 Wstęp

Celem projektu jest stworzenie aplikacji umożliwiającej generowanie dwuwymiarowego labiryntu oraz wizualizację procesu wyszukiwania ścieżki pomiędzy dwoma punktami. Labirynt w kontekście projektu to struktura siatki, gdzie każde pole może stanowić przejście lub ścianę.



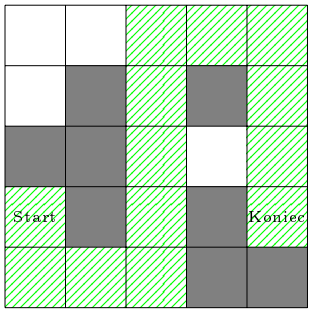
Rysunek 1: Przykładowy labirynt 5x5 z zaznaczonym startem i końcem, gdzie białe pole - przejście, czarne - ściana.

Można zauważyć, że taka struktura jest reprezentacją grafu, gdzie pola odpowiadają wierzchołkom, a krawędzie łączą sąsiadujące pola przejściowe.

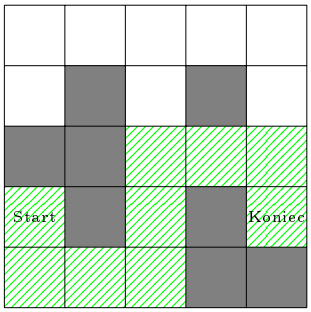


Rysunek 2: Graf reprezentujący labirynt z rys. 1.

W projekcie istotne jest porównanie różnych algorytmów wyszukiwania ścieżki, które pozwalają znaleźć trasę między dwoma punktami. W najlepszym przypadku celem jest znalezienie ścieżki *optymalnej*, czyli takiej, która minimalizuje liczbę kroków, co w kontekście grafu o jednakowych wagach krawędzi sprowadza się do znalezienia drogi o minimalnej długości.



(a) Ścieżka nieoptymalna



(b) Ścieżka optymalna

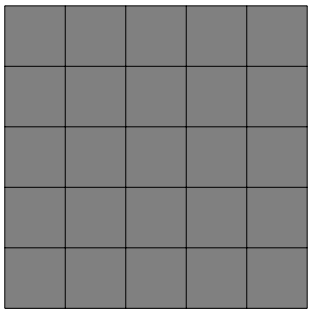
Rysunek 3: Porównanie ścieżek w labiryncie.

2 Generowanie labiryntu

2.1 Algorytm Prima

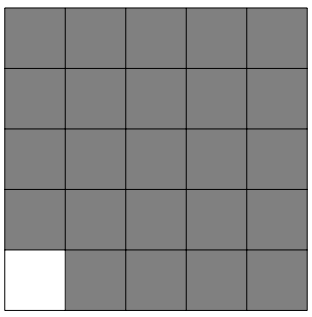
Algorytm Prima to metoda generowania labiryntów wykorzystująca technikę tworzenia minimalnego drzewa rozpinającego (MST) dla grafu reprezentującego planszę labiryntu. W kontekście generowania labiryntu działanie algorytmu przebiega następująco:

1. Na początku tworzona jest plansza, w której wszystkie pola są oznaczone jako ściany.



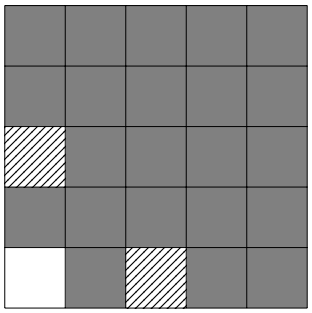
Rysunek 4: Cała plansza stanowi ściany.

2. Następnie wybierane jest losowe pole i oznaczane jako przejście.



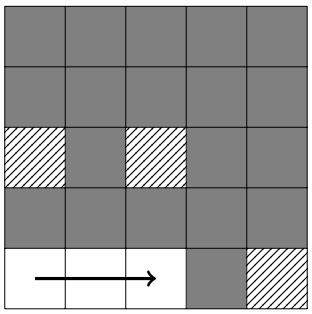
Rysunek 5: Losowe pole startowe.

3. Do zbioru krawędzi dodawane są sąsiednie komórki, do których można przejść bezpośrednio z miejsca startowego. Za sąsiednie uznaje się komórki oddalone o jedno pole w pionie lub poziomie.



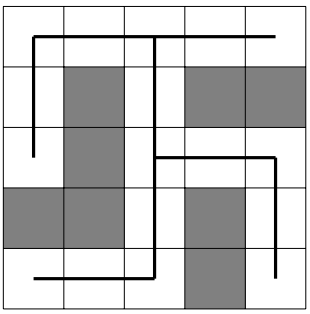
Rysunek 6: Wybór sąsiednich pól.

4. Losowana jest jedna krawędź ze zbioru. Jeśli prowadzi ona do nieodwiedzonego pola, to tworzy się przejście pomiędzy bieżącym polem a nowym (usuwana zostaje ściana między nimi), a nowe pole zostaje oznaczone jako przejście. Do zbioru krawędzi dodawani są sąsiedzi nowego pola.

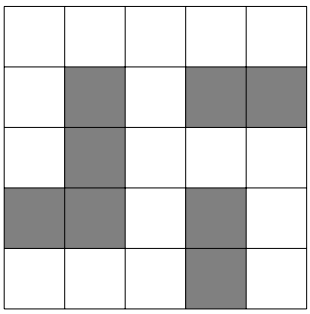


Rysunek 7: Utworzenie krawędzi do sąsiedniego pola.

5. Proces powtarza się, dopóki zbiór krawędzi nie będzie pusty.



(a) Wyznaczanie kolejnych krawędzi labiryntu.



(b) Labirynt powstały na podstawie wyznaczonych krawędzi.

Rysunek 8: Kolejne kroki działania algorytmu.

Algorytm ten gwarantuje, że utworzony labirynt nie będzie zawierał zamkniętych pętli, a pomiędzy dowolnymi dwoma punktami zawsze będzie istniała możliwa ścieżka.

2.2 Algorytm Kruskala

Algorytm Kruskala podobnie jak algorytm Prima jest oparty na tworzeniu minimalnego drzewa rozpinającego. Algorytm traktuje ściany między polami labiryntu jako potencjalne krawędzie (przejścia w labiryncie). Działanie algorytmu w kontekście labiryntu przebiega następująco:

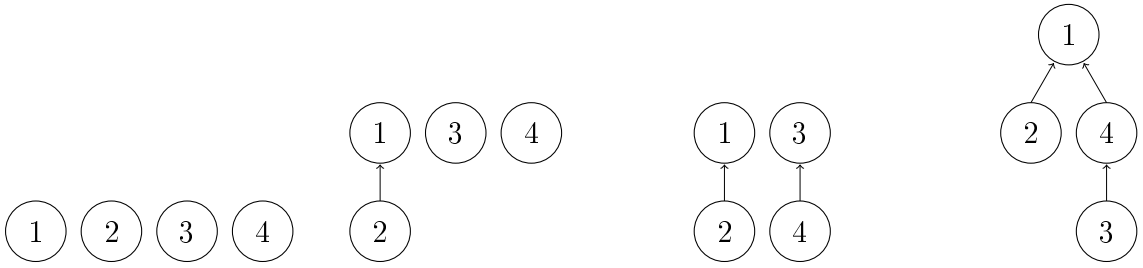
1. Na początku tworzona jest plansza, w której wszystkie pola są oznaczone jako przejścia.

2. Tworzony jest zbiór rozłącznych zbiorów *Disjoint Set*, w którym każda komórka należy do własnego zbioru — oznacza to, że na początku żadna komórka nie jest połączona z inną.

   Struktura *Disjoint Set* reprezentuje rozłączne zbiory elementów za pomocą drzew. Na początku każdy element tworzy pojedynczy, jednoelementowy zbiór, którego reprezentantem jest korzeń drzewa.

   Kluczową operacją na tej strukturze jest *Union*, która łączy dwa zbiory, tworząc jedno drzewo, w którym korzeń jednego zbioru staje się potomkiem korzenia drugiego. W ten sposób zbiory są scalane.

   Schematyczne przedstawienie scalania zbiorów znajduje się na Rysunku 9.



(a) Krok 1 (b) Krok 2 (c) Krok 3 (d) Krok 4

Rysunek 9: Kolejne kroki scalania zbioru

3. Zbierane są wszystkie możliwe krawędzie, czyli miejsca między dwiema bezpośrednio sąsiadującymi komórkami, które można potencjalnie zamienić na ściany. Następnie są one tasowane w losowej kolejności. W odróżnieniu od algorytmu Prima, za sąsiadujące uznaje się tutaj pola przylegające bezpośrednio.

4. Iterujemy po każdej ścianie ze zbioru:

   (a) Dla danej ściany sprawdzane są dwie komórki, które ta ściana oddziela.

   (b) Jeśli te komórki należą do różnych zbiorów (nie są jeszcze połączone), to:

      i. Usuwana jest ściana między nimi - tworzy się przejście.

      ii. Obie komórki zostają połączone w jednym zbiorze.

   (c) Jeśli komórki są już w tym samym zbiorze (czyli istnieje już droga między nimi), ściana nie jest usuwana — zapobiega to tworzeniu cykli.

5. Proces trwa do momentu, gdy wszystkie komórki zostaną połączone w jeden zbiór — czyli zostanie utworzone jedno spójne drzewo bez cykli.