| Politechnika Świętokrzyska w Kielcach<br>Wydział Elektrotechniki, Automatyki i Informatyki<br>Katedra Informatyki, Elektroniki i Elektrotechniki | | |
|---|---|---|
| **Kierunek:**<br><br>Informatyka | **Laboratorium:**<br><br>Systemów Dynamicznych | |
| **Grupa dziekańska:**<br><br>2IZ12A | **Temat ćwiczenia:**<br>Zapoznanie się z zastosowaniem Programowania Dynamicznego w problemach Systemów Dynamicznych. | **Wykonał <u>zespół</u> w składzie:**<br>A. Szymkiewicz, |
| **Data wykonania:**<br><br>10.06.2024 | **Data oddania:** | **Ocena i podpis:** |

## Opis problemu

Projekt dotyczy wyznaczania optymalnej trajektorii sterowań w siatce punktów o zadanych kosztach przejścia. Celem jest znalezienie ścieżki o minimalnym koszcie przejścia od lewego dolnego rogu do prawego górnego rogu siatki. Problem ten może mieć zastosowanie w dziedzinach takich jak robotyka, logistyka czy planowanie tras.

## Przyjęte założenia

1. **Siatka punktów**: Jest reprezentowana przez macierz dwuwymiarową, gdzie każdy element oznacza koszt przejścia przez dany punkt.
2. **Punkty początkowy i końcowy**: Start znajduje się w lewym dolnym rogu, a koniec w prawym górnym rogu siatki.
3. **Możliwe ruchy**: Dozwolone są tylko ruchy w górę i w prawo.

## Omówienie funkcji programu

### Struktura `Point`

```
struct Point {
    int x;
    int y;
};
```

Reprezentuje punkt na siatce z koordynatami `x` i `y`.

### Funkcja `printGridWithNames`

```
void printGridWithNames(int rows, int cols);
```

Wyświetla siatkę z nazwami punktów w formie liter i liczb, co ułatwia użytkownikowi orientację.

### Funkcja `printGrid`

```
void printGrid(const vector<vector<int>>& grid);
```

Wyświetla wartości siatki, prezentując koszty przejścia przez poszczególne punkty.

### Funkcja `printPath`

```
void printPath(const vector<vector<int>>& grid, const vector<Point>& path);
```

Wyświetla trajektorię na siatce oraz sumaryczny koszt przejścia. Początek oznaczony jest literą 'P', a koniec literą 'K'.

### Funkcja `findPaths`

```
void findPaths(const vector<vector<int>>& grid, int x, int y,
vector<Point>& currentPath, int currentCost, vector<vector<Point>>&
allPaths, vector<int>& allCosts);
```

Rekurencyjnie znajduje wszystkie możliwe ścieżki od punktu początkowego do końcowego. Wykorzystuje technikę backtrackingu do eksploracji ścieżek. Funkcja findPaths korzysta z techniki rekurencyjnej do przeszukiwania wszystkich możliwych ścieżek od punktu początkowego do końcowego. Algorytm ten eksploruje wszystkie możliwe ruchy (w górę i w prawo) i cofa się (backtracks), gdy ścieżka nie prowadzi do celu. Jest to metoda brute-force, która może być bardzo kosztowna obliczeniowo dla dużych siatek.

### Funkcja `main`

```
int main();
```

Główna funkcja programu:

- Pobiera od użytkownika wymiary siatki i wartości kosztów punktów.
- Wyświetla siatkę z nazwami punktów.
- Znajduje wszystkie możliwe ścieżki i ich koszty.
- Wyświetla wszystkie ścieżki.
- Wybiera i wyświetla najlepszą ścieżkę o minimalnym koszcie.

## Testy (przykładowe zadania)

### Przykład 1

- **Wejście**: Siatka 3x3 z kosztami:

```
1 2 3      | A1 A2 K     A - pierwszy rząd, B,C... - kolejne rzędy
4 5 6      | B1 B2 B3    1 - piersza kolumna, 2,3.. - kolejne kolumny
7 8 9      | P  C2 C3    P - początek, K- Koniec
```

- **Oczekiwany wynik**: Najlepsza ścieżka z minimalnym kosztem: `P -> B1 -> A1 -> A2 -> K`.

**Przykład 2**

- **Wejście**: Siatka 2x3 z kosztami:

```
1 2 3
4 5 6
```

- **Oczekiwany wynik**: Najlepsza ścieżka z minimalnym kosztem: `P -> A1 -> A2 -> K`.

## Analiza uzyskanych wyników

Program prawidłowo znajduje wszystkie możliwe ścieżki oraz wybiera tę o minimalnym koszcie. Implementacja rekurencyjna jest poprawna, jednak może być nieefektywna dla dużych siatek ze względu na dużą liczbę możliwych ścieżek do przeszukania. W praktycznych zastosowaniach należałoby rozważyć zastosowanie bardziej zaawansowanych algorytmów, takich jak algorytm Dijkstry lub A*.