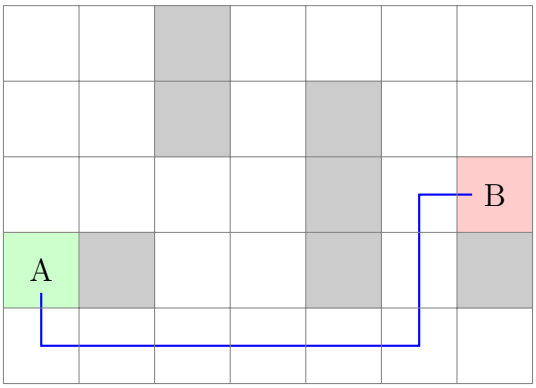
ZAAWANSOWANE PROGRAMOWANIE KOMPUTEROWE WNE (2022)

KRZYSZTOF ZIEMIAŃSKI

11. Szukanie najkrótszej drogi

Labirynt składa się z kwadratowych pól ułożonych w prostokątną siatkę. Niektóre pola labiryntu są wolne i można się po nich poruszać a niektóre sa zajęte i nie ma na nie wstępu. Z danego pola możemy przechodzić bezpośrednio tylko na pola, które stykają się bokami. Naszym celem jest znalezienie najkrótszej drogi pomiędzy dwoma zadanymi polami. Na przykład najkrótsza droga z pola A na pole B na rysunku poniżej ma długość 9:



Wykorzystamy do tego celu algorytm przeszukiwanie grafów wszerz.

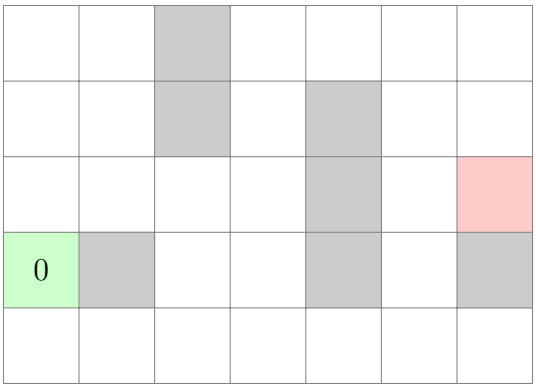
11.1. **Przeszukiwanie grafów.** Graf składa się ze zbioru wierzchołków. Każde dwa wierzchołki mogą (ale nie muszą) być połączone krawędzią. Każda krawędź może mieć przypisaną długość, ale zazwyczaj zakłada się, że wszystkie krawędzie mają długość 1.

W naszym przykładzie:

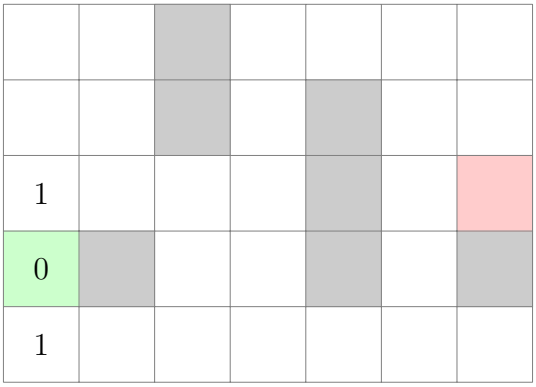
- wierzchołki to wolne pola,
- dwa pola są połączone krawiędzią jeśli stykają się bokami.

Mamy dwa wyróżnione pola, oznaczone A i B. Chcemy znaleźć najkrótszą drogę z A do B, ale tak naprawdę będziemy szukać najkrótszych dróg z A do wszystkich możliwych pól, bo nie jesteśmy w stanie stwierdzić, które drogi prowadzą "w kierunku" B.

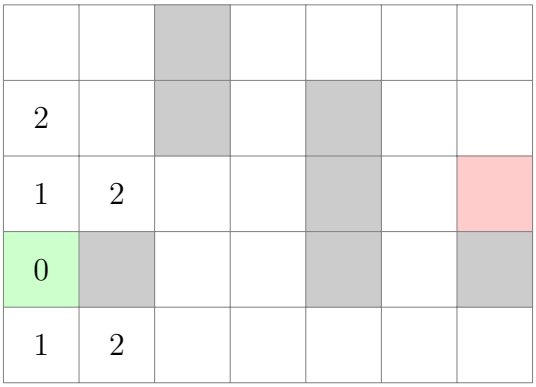
Będziemy kolejno wyszukiwać pola, do których znamy długość najkrótszej drogi z A. Na początek znamy jedno takie pole: jest to A, oczywiście najkrótsza droga z A do A ma długość 0. Zapisujemy to więc w naszym labiryncie:



Oczywiście A jest jedynym polem, do którego odległość wynosi 0. Szukamy teraz i oznaczamy pola, do których odległość wynosi 1. Są to pola bezpośrednio sąsiadujące z A:



Pola o odległości 2 to te, które sąsiadują z polami o odległości 1 (i oczywiście nie mają odległości 0 ani 1). Oznaczamy więc te pola:



W taki sam sposób znajdujemy pola o odległościach 3,4, itd., aż dojdziemy do pola B. Może się też zdarzyć, że skończą się nowe pola, którym możemy oznaczyć odległość — oznacza to, że z A do B nie ma żadnej drogi.

3	4		6	7	8	9
2	3		5		9	
1	2	3	4		8	9
0		4	5		7	
1	2	3	4	5	6	7

Widzimy więc, że odległość do punktu B wynosi 9. Jak znaleźć najkrótszą drogę? Próba znalezienia drogi od punktu A może się nie powieść: jeśli pierwszy krok wykonamy do góry, najkrótszej drogi nie znajdziemy. Ale możemy to zrobić konstruując drogę "od końca". Zaczynamy od B (gdzie jest liczba 9), cofamy się na dowolne pole z liczbą 8 (na pewno takie jest, być może więcej niż jedno ale to nieistotne), a potem na pole z liczbą 7, itd. W końcu dotrzemy do punktu A.

3	4		6	7	8	9
2	3		5		9	
1	2	3	4		8	9
0		4	5		7	
1	2	3	4	5	6	7

Podobnie można szukać najkrótszych dróg w innych grafach, niekoniecznie pochodzących od kwadratowej siatki.

11.2. **Zadania.**

(1) Dodać następujące metody do klasy `Plansza`:

- `int ileZajetych()`
  zwraca liczbę zajętych pól na planszy
- `bool wolnaKolumna()`
  zwraca `true` jeśli pewna kolumna jest wolna
- `void zmienRozmiar(int nowaSzer, int nowaWys)`
  zmienia rozmiar planszy, nowe pola powinny być puste

(2) Dodać do klasy `Plansza` metodę

```
int odleglosc(int x1, int y1, int x2, int y2)
```

która zwraca długość najkrótszej drogi z pola (`x1`,`y1`) do pola (`x2`,`y2`), lub -1 jeśli taka droga nie istnieje.