Od startu k cíli trochu jinak

Zpětné trasování pomocí algoritmu BFS

Krokování algoritmu

3. března 2026

Zadání a princip řešení

Zadání úlohy

- Máme dánu mapu ve tvaru pravoúhlé mřížky.
- Každé políčko obsahuje buď znak směru (Sever, Jih, Východ, Západ), kterým se z něj musíme vydat, nebo cíl (C).
- Úkolem je pro každé políčko zjistit, zda se z něj po šipkách dostaneme do cíle, nebo nikoliv (vyjdeme mimo mapu, nebo se zacyklíme).

Princip řešení (Zpětný chod)

- Namísto zkoušení cest ze všech startů začneme v **cíli**.
- Klademe si inverzní otázku: *Z kterých sousedních políček se dostanu na moje aktuální políčko?*
- Pro procházení mapy využijeme datovou strukturu **fronta** a algoritmus **BFS**.

Ukázková mapa a souřadný systém

sloupec $c \longrightarrow$

řádek $r \downarrow$

J	Z	V	V	J
Z	V	J	V	J
V	J	J	S	Z
J	C	Z	Z	J
S	S	S	V	Z

Pohyb po mapě:
Písmena na políčkách určují směr, kterým z daného políčka musíme odejít:

- **S** (Sever) – o řádek výše
- **J** (Jih) – o řádek níže
- **V** (Východ) – o sloupec doprava
- **Z** (Západ) – o sloupec doleva
- **C** – Cíl, zde cesta končí

Upozornění: Odlišný souřadný systém!

Všimněte si orientace os. První souřadnice (r) určuje řádek a roste směrem dolů. Druhá souřadnice (c) určuje sloupec a roste doprava. Jde o maticový zápis (r, c), který funguje jinak, než klasická kartézská soustava (x, y), na kterou jste běžně zvyklí z matematiky!

Trasování algoritmu – Vlna 0 a 1

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	V	J	J	S	Z
3	J	**C**	Z	Z	J
4	S	S	S	V	Z

Komentář:

Hledáme na mapě cíl (**C**).

Fronta ke zpracování

(prázdná)

Trasování algoritmu – Vlna 0 a 1

$r \setminus c$	0	**1**	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	V	J	J	S	Z
3	J	**0**	Z	Z	J
4	S	S	S	V	Z

Komentář:

Našli jsme cíl na souřadnicích $(3, 1)$. Nahradíme jej symbolem **0** (vzdálenost nula kroků) a tyto souřadnice zařadíme do fronty ke zpracování.

Fronta ke zpracování

(3, 1)

Trasování algoritmu – Vlna 0 a 1

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	V	**J**	J	S	Z
3	J	**0**	**Z**	Z	J
4	S	**S**	S	V	Z

Komentář:

Z fronty vyjmeme první prvek **(3, 1)** a na mapě najdeme všechny jeho sousedy, ze kterých se do něj po šipce dostaneme (ukazují na něj).

Jsou to políčka **J** (shora), **S** (zdola) a **Z** (zprava). Jejich souřadnice zařadíme na konec fronty.

Fronta ke zpracování

(3, 1), (2, 1), (4, 1), (3, 2)

Trasování algoritmu – Vlna 0 a 1

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	V	**1**	J	S	Z
3	J	**0**	**1**	Z	J
4	S	**1**	S	V	Z

Komentář:

Políčka, která jsme v předchozím kroku přidali do fronty, nyní v mapě označíme číslem **1**.

Představují takzvanou **první vlnu** – tedy všechna políčka, ze kterých se po šipkách spolehlivě dostaneme do cíle přesně za 1 krok.

Fronta ke zpracování

(3, 1), (2, 1), (4, 1), (3, 2)

Trasování algoritmu – Vlna 2

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	**V**	**1**	J	S	Z
3	J	0	1	Z	J
4	S	1	S	V	Z

Komentář:

Z fronty odebíráme **(2, 1)**. Z neobjevených sousedů na něj ukazuje pouze **V** zleva. Přidáme **(2, 0)** do fronty.

Fronta ke zpracování

(3, 1), **(2, 1)**, (4, 1), (3, 2), **(2, 0)**

Trasování algoritmu – Vlna 2

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	**V**	1	J	S	Z
3	J	0	1	Z	J
4	S	**1**	S	V	Z

Komentář:

Odebíráme **(4, 1)**. Žádný z jeho neobjevených sousedů k němu nevede.

Fronta ke zpracování

(3, 1), (2, 1), **(4, 1)**, (3, 2), (2, 0)

Trasování algoritmu – Vlna 2

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	**V**	1	**J**	S	Z
3	J	0	**1**	**Z**	J
4	S	1	**S**	V	Z

Komentář:

Odebíráme **(3, 2)**. Na toto políčko ukazují sousedé **J** (shora), **S** (zdola) a **Z** (zprava). Zařadíme je do fronty.

Fronta ke zpracování

. . . (4, 1), **(3, 2)**, (2, 0), **(2, 2)**, **(4, 2)**, **(3, 3)**

Trasování algoritmu – Vlna 2

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	**2**	1	**2**	S	Z
3	J	0	1	**2**	J
4	S	1	**2**	V	Z

Komentář:

Všechna políčka objevená v předchozích krocích dostávají na tomto snímku číslo **2**. Tvoří **druhou vlnu** (do cíle dojdou za 2 kroky).

Fronta ke zpracování

. . . (3, 2), (2, 0), (2, 2), (4, 2), (3, 3)

Trasování algoritmu – Vlna 3

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	J	V	J
2	**2**	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Zpracováváme vlnu 2. Odebíráme **(2, 0)**. Žádný soused na něj neukazuje.

Fronta ke zpracování

... (3, 2), **(2, 0)**, (2, 2), (4, 2), (3, 3)

Trasování algoritmu – Vlna 3

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	**J**	V	J
2	2	1	**2**	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Odebíráme **(2, 2)**. Horní soused **J** k němu vede, přidáme **(1, 2)** do fronty.

Fronta ke zpracování

. . . (2, 0), **(2, 2)**, (4, 2), (3, 3), **(1, 2)**

Trasování algoritmu – Vlna 3

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	**J**	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	**2**	V	Z

Komentář:

Odebíráme **(4, 2)**. Žádný nový soused na něj neukazuje.

Fronta ke zpracování

. . . (2, 2), **(4, 2)**, (3, 3), (1, 2)

Trasování algoritmu – Vlna 3

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	**J**	V	J
2	2	1	2	S	Z
3	J	0	1	**2**	J
4	S	1	2	V	Z

Komentář:

Odebíráme **(3, 3)**. Žádný nový soused.

Fronta ke zpracování

. . . (4, 2), **(3, 3)**, (1, 2)

Trasování algoritmu – Vlna 3

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	V	**3**	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Zpracování vlny 2 je hotovo. Nové políčko označíme číslem **3** (třetí vlna).

Fronta ke zpracování

$\ldots (3, 3), (1, 2)$

Trasování algoritmu – Vlna 4 a konec

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	**V**	**3**	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Zpracováváme vlnu 3. Odebíráme **(1, 2)**. Levý soused **V** k němu vede, přidáme **(1, 1)** do fronty.

Fronta ke zpracování

. . . (3, 3), **(1, 2)**, **(1, 1)**

Trasování algoritmu – Vlna 4 a konec

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	**4**	3	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Nové políčko získá číslo **4**.

Fronta ke zpracování

$\ldots (1, 2), (1, 1)$

Trasování algoritmu – Vlna 4 a konec

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	**4**	3	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Zpracováváme vlnu 4. Odebíráme poslední prvek **(1, 1)**. Žádný neobjevený soused k němu již nevede.

Fronta ke zpracování

. . . (1, 2), **(1, 1)**

Trasování algoritmu – Vlna 4 a konec

$r \setminus c$	0	1	2	3	4
0	J	Z	V	V	J
1	Z	**4**	3	V	J
2	2	1	2	S	Z
3	J	0	1	2	J
4	S	1	2	V	Z

Komentář:

Fronta je prázdná, algoritmus končí!

U všech políček s čísly známe nejkratší cestu do cíle. Z libovolného zbylého políčka s písmenem buď vyjdeme mimo mapu, nebo se zacyklíme.

Fronta ke zpracování

... (1, 1) – *(fronta je prázdná)*

Pseudokód algoritmu – 1. část

Algoritmus 1 Zpětné prohledávání do šířky – Inicializace

Vstup: Mapa (mřížka políček) *mapa*
Výstup: Tabulka minimálních vzdáleností *vzdalenost*

```
 1: funkce NajdiCesty(mapa)
 2:     fronta ← vytvoř prázdnou frontu
 3:     vzdalenost ← tabulka plná hodnot ∞
 4:                                  ▷ Krok 1: Nalezení cíle a inicializace
 5:     pro každé políčko (r, c) v mapa dělej
 6:         pokud mapa[r][c] == 'C' pak
 7:             fronta.push((r, c))
 8:             vzdalenost[r][c] ← 0
 9:         konec pokud
10:     konec pro
```

Pseudokód algoritmu – 2. část

Algoritmus 2 Zpětné prohledávání do šířky – Šíření vln

```
11:                                  ▷ Krok 2: Samotné zpracování fronty (vlny)
12:     dokud fronta není prázdná dělej
13:         (r, c) ← fronta.popleft()
14:         pro každé souseda (nr, nc) políčka (r, c) dělej
15:             pokud (nr, nc) leží v mapa a vzdalenost[nr][nc] == ∞ pak
16:                 pokud šipka z mapa[nr][nc] ukazuje na (r, c) pak
17:                     vzdalenost[nr][nc] ← vzdalenost[r][c] + 1
18:                     fronta.push((nr, nc))
19:                 konec pokud
20:             konec pokud
21:         konec pro
22:     konec dokud
23:     vrať vzdalenost
24: konec funkce
```

Časová složitost algoritmu

Časová složitost tohoto algoritmu je $\mathcal{O}(N)$, kde N je celkový počet políček v mřížce, protože každé políčko přidáme do fronty nejvýše jednou a při jeho zpracování se podíváme na čtyři (tedy konstantní počet) sousedů.