Jozef Waldhauser

Úloha 8

Hľadanie cesty v 2D mape

Obsah

[Úloha 8 – zadanie 2](#_Toc197772713)

[Úloha 8 – zdroje 2](#_Toc197772714)

# Úloha 8 – zadanie

1. Naimplementujte metodu path\_planner(self, start, goal), která vezme dvojici souřadnic start a dvojici souřadnic cíl a v mapě block.py vrátí dvě proměnné - frontu obsahující souvislou posloupnost dvojic souřadnic ze startu do cíle a pole expandovaných uzlů k vybarvení
2. V metodě implementujte algoritmy Greedy best first search, Dijkstra a A\*
3. Díky vizualizaci porovnejte množství expandovaných uzlů u jednotlivých algoritmů

# Úloha 8 – zdroje

<https://www.youtube.com/watch?v=GVvN0ikNekw> – greedy bfs

<https://www.geeksforgeeks.org/greedy-best-first-search-in-ai/> - greedy bfs

<https://www.youtube.com/watch?v=6TsL96NAZCo> – a\*

<https://colab.research.google.com/github/cloudpedagogy/data-science-programming/blob/main/python-programming/07_Introduction_to_File_I_O.ipynb/#scrollTo=s_LodqCfrYTV> – nacitanie suborov

<https://en.wikipedia.org/wiki/Taxicab_geometry> - manhattan distance

<https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm> - dijkstra

**Čo je Greedy Best-First Search**

Greedy Best-First Search je vyhľadávací algoritmus v umelej inteligencii, ktorý sa pokúša nájsť najsľubnejšiu cestu od štartovacieho bodu k cieľu. Používa heuristickú funkciu na ohodnotenie každého uzla a vždy expanduje uzol, ktorý vyzerá ako najbližší k cieľu.

**Základný princíp**

Algoritmus vždy vyberá najbližší uzol k cieľu podľa heuristickej funkcie **bez ohľadu na to, koľko krokov už prešiel**. Preto sa nazýva "greedy" (chamtivý) - zaujíma ho len odhadovaná vzdialenosť k cieľu, nie celková dĺžka cesty.

**Návrh algoritmu (pseudokód)**

|  |
| --- |
| funkcia GreedyBestFirstSearch(graf, start, ciel):  // Inicializácia  1. Vytvor prázdny zoznam navštívených uzlov (CLOSED)  2. Vytvor prioritný front (OPEN) a vlož doňho štartovací uzol  3. Prirad každému uzlu heuristickú hodnotu (odhadnutá vzdialenosť k cieľu)  // Hlavný cyklus  kým OPEN nie je prázdny:  1. Vyber uzol s najmenšou heuristickou hodnotou z OPEN  2. Ak je to cieľový uzol, vráť nájdenú cestu a skonči  3. Pridaj uzol do CLOSED  4. Pre každého suseda aktuálneho uzla:  a. Ak je v CLOSED, ignoruj ho  b. Ak nie je v OPEN, pridaj ho do OPEN a vypočítaj jeho heuristickú hodnotu  c. Ak je už v OPEN, aktualizuj cestu, ak je nová lepšia  // Ak sme sa dostali sem, cesta neexistuje  vráť "Cesta neexistuje" |

## Greedy Best-First Search (Greedy BFS) - vysvetlenie po slovensky

### Čo je Greedy Best-First Search

Greedy Best-First Search je vyhľadávací algoritmus v umelej inteligencii, ktorý sa pokúša nájsť najsľubnejšiu cestu od štartovacieho bodu k cieľu. Používa heuristickú funkciu na ohodnotenie každého uzla a vždy expanduje uzol, ktorý vyzerá ako najbližší k cieľu.

### Základný princíp

Algoritmus vždy vyberá najbližší uzol k cieľu podľa heuristickej funkcie \*\*bez ohľadu na to, koľko krokov už prešiel\*\*. Preto sa nazýva "greedy" (chamtivý) - zaujíma ho len odhadovaná vzdialenosť k cieľu, nie celková dĺžka cesty.

### Návrh algoritmu (pseudokód)

```

funkcia GreedyBestFirstSearch(graf, start, ciel):

// Inicializácia

1. Vytvor prázdny zoznam navštívených uzlov (CLOSED)

2. Vytvor prioritný front (OPEN) a vlož doňho štartovací uzol

3. Prirad každému uzlu heuristickú hodnotu (odhadnutá vzdialenosť k cieľu)

// Hlavný cyklus

kým OPEN nie je prázdny:

1. Vyber uzol s najmenšou heuristickou hodnotou z OPEN

2. Ak je to cieľový uzol, vráť nájdenú cestu a skonči

3. Pridaj uzol do CLOSED

4. Pre každého suseda aktuálneho uzla:

a. Ak je v CLOSED, ignoruj ho

b. Ak nie je v OPEN, pridaj ho do OPEN a vypočítaj jeho heuristickú hodnotu

c. Ak je už v OPEN, aktualizuj cestu, ak je nová lepšia

// Ak sme sa dostali sem, cesta neexistuje

vráť "Cesta neexistuje"

```

### Hlavné body algoritmu

1. \*\*Prioritný front\*\* - Používa sa na uschovanie uzlov, ktoré ešte neboli preskúmané. Uzly sú zoradené podľa heuristickej hodnoty.

2. \*\*Heuristická funkcia\*\* - Odhaduje vzdialenosť od aktuálneho uzla k cieľu. Čím nižšia hodnota, tým lepšie.

3. \*\*Chamtivý výber\*\* - Vždy expanduje uzol s najnižšou heuristickou hodnotou.

4. \*\*Neúplnosť\*\* - Greedy BFS nie je úplný algoritmus - môže sa "zaseknúť" v lokálnom minime a nenájsť riešenie, aj keď existuje.

### Výhody

- Rýchlejší ako slepé vyhľadávanie (BFS, DFS)

- Spotrebuje menej pamäte ako A\*

- Jednoducho implementovateľný

### Nevýhody

- Negarantuje nájdenie najkratšej cesty

- Môže sa zaseknúť v lokálnom minime

- Nie je úplný (nemusí nájsť riešenie, aj keď existuje)

### Príklad použitia

Vhodný pre problémy, kde:

- Rýchlosť je dôležitejšia ako optimalita

- Heuristická funkcia je relatívne presná

- Lokálne minimum nie je problém

## Design Algoritmu

* Funkcia: Načítanie 2D mapy reprezentovanej ako maticu
  + Naša mapa je mapa v danej klase self.map
* Class: PathFinder
  + Method: heuristic
    - Manhattan distance current node a cieľa
  + Method: Get\_neighbors
    - Kontroluje stĺpce, riadky a podľa toho rozhoduje či je „platný“ sused (keď je v prvom stĺpci nepriradí ľavého suseda atď.)
    - Je potrebne kontrolovať riadky a stĺpce kvôli tomu že ak dáme do matice matrix[x][y] kde x = -1 tak vráti posledný riadok čo je nechcené chovanie
  + Method: greedy\_best\_first\_search
    - Vstupy štart, cieľ (self, start, goal)
    - Načíta heuristickú hodnotu štartovacieho bodu
    - Dokým nie je heuristika 0 (tzn. nie sme v cieli) opakuje while loop:
      * Zisti susedov bodu, v ktorom sme (metóda get\_neighbors)
      * Vypočítaj iteratívne heuristiku susedov (metóda heuristic)
      * Vyber náhodne z susedov, ktorí majú nižšiu heuristiku
        + Ak nie sú susedia ukončím hľadanie
      * Nastav vybraného suseda ako bod, v ktorom sme
      * Pridaj ho do path array (array of tuples)
    - Return path