

Hledání nejkratší cesty v
grafu - A^*

Algoritmus

- Pro nalezení nejkratší cesty v grafu z daného výchozího uzlu do všech ostatních uzlů
- Vznik 1968 - Peter Hart, Nils Nilsson, a Bertram Raphael
- Vylepšení Dijkstrova algoritmu
 - Upřednostňuje hrany s nižším ohodnocením
 - Průběžně aktualizuje vzdálenosti k nenavštíveným sousedním uzlům
 - Prohledává graf systematicky bez ohledu na cílový uzel (rozdíl mezi A^*)
- Informovaný algoritmus – využívá heuristickou funkci
 - Odhad vzdálenosti z daného uzlu do cíle
- $f(n) = g(n)$ (cesta od počátečního uzlu k aktuálnímu) + $h(n)$ (heuristika)

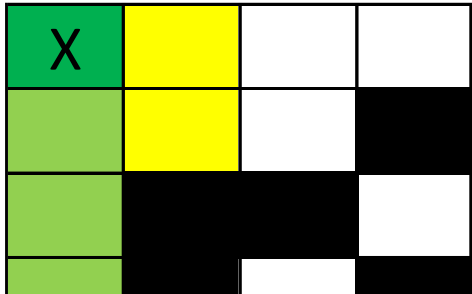
Heuristika

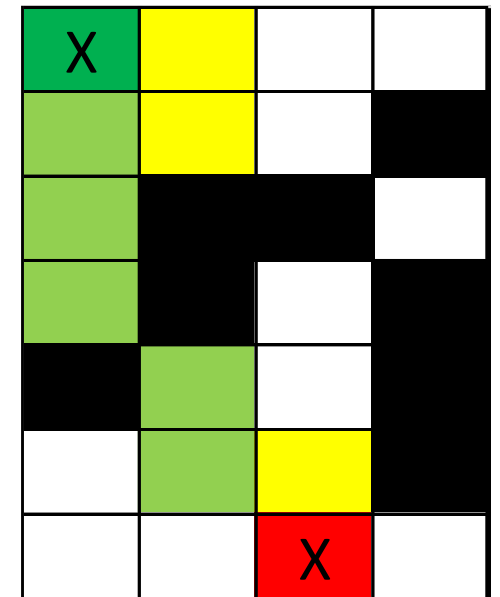
- Pouze chytrý odhad, více způsobů:
 - Manhattan Distance – využíváno pro pohyb ve čtyřech směrech
$$h(n) = |n.x - goal.x| + |n.y - goal.y|$$
 - Euclidean Distance – založeno na Pythagorově větě
$$h(n) = \sqrt{(n.x - goal.x)^2 + (n.y - goal.y)^2}$$
 - Diagonal Distance – pohyb včetně diagonálních směrů

Složitost

- Složitost $O(b^d)$ – nejhorší případ
 - b je maximální počet potomků uzlu (branching factor)
 - d je hloubka nejkratší cesty (depth)
- Nejlepší případ -> lineární složitost $O(n)$, n počet uzlů
- X Dijkstrův algoritmus
 - Složitost $O((V + E) \log V)$
 - V je počet uzlů (vrcholů) v grafu, E je počet hran (hran) v grafu

Metody

- search(Cell start, Cell end) - najde nejkratší cestu od počátečního uzlu k cíli
 - Začnu s počátečním uzlem, hodnoty na inf, vložím do prioritní fronty
 - Dokud není fronta prázdná nebo nejsem v cíli
 - Vyberu vrchol curr s nejmenším f pomocí prioritní fronty
 - Projdu všechny sousedy curr, pokud má smysl ho zkoumat
 - Spočítám vzdálenost od curr ($\rightarrow g$) a heuristikou vzdálenost od cíle (f)
 - Vrchol curr přesunu z prvků na hranici do prozkoumaných
 - Pokud nedojdu do cíle a fronta je prázdná \rightarrow neexistuje řešení
- 



Metody

- `make_path(Cell start, Cell end)` - vytvoří cestu zpětně z cíle do počátečního uzlu
 - Dokud nejsem zpátky na startu:
 - Přidám aktuální buňku do vektoru `path`
 - Nové souřadnice procházené buňky budou rodiče předchozí
 - Na konci přidá počáteční uzel do vektoru `path`
- `write_path()` – vypíše nalezenou cestu od startu do cíle
 - Prochází vektor `path` pomocí zpětné iterace

Reference

- A* Pathfinding Algorithm | Baeldung on Computer Science
- A* Search Algorithm – GeeksforGeeks
- A* pathfinding algorithm - Growing with the Web