Algorytm A*

Izabela Czarny, Michał Kreft

Wydział Inżynierii Materiałowej i Fizyki
Politechnika Krakowska

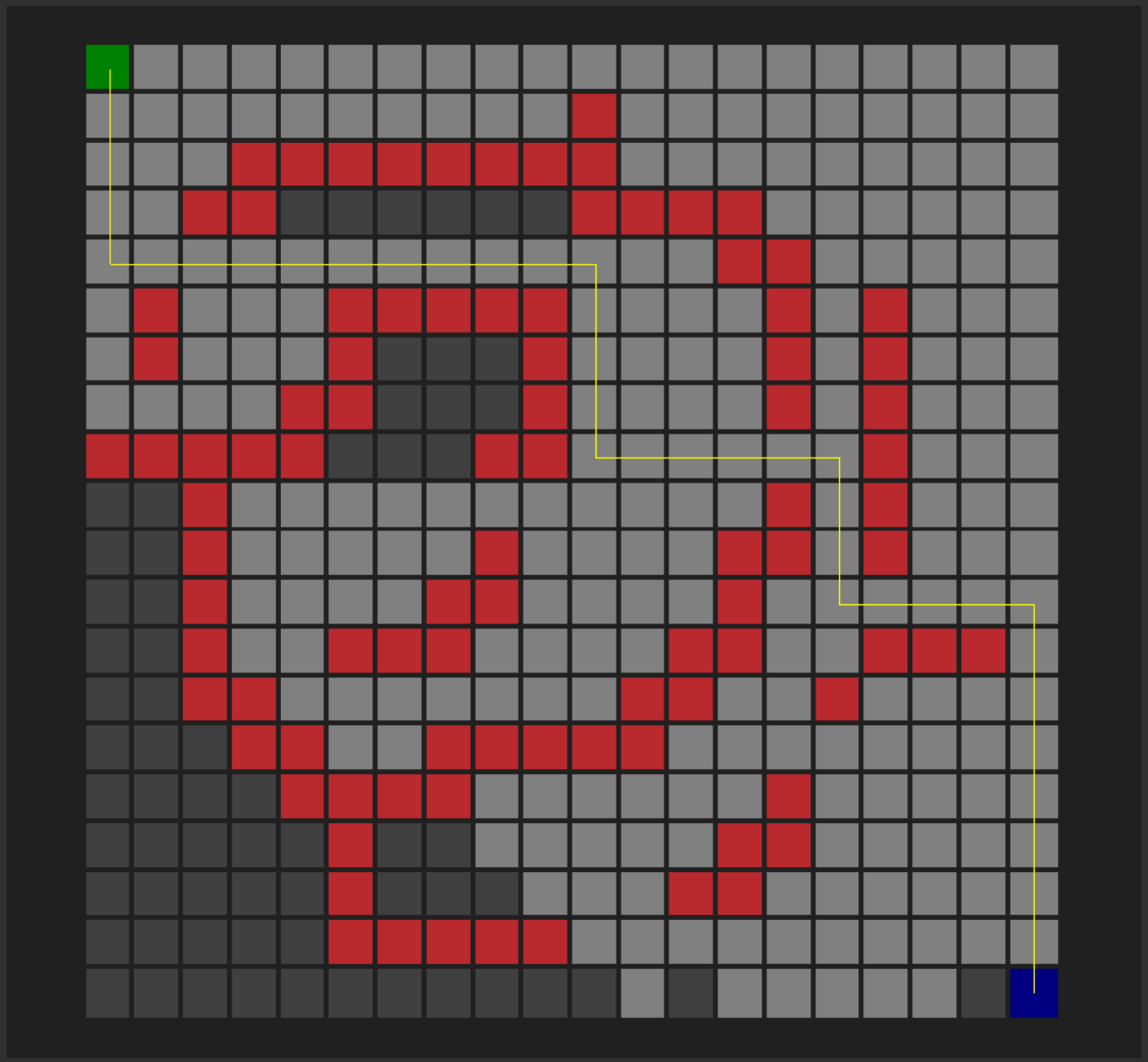
11 czerwca 2021

Spis treści

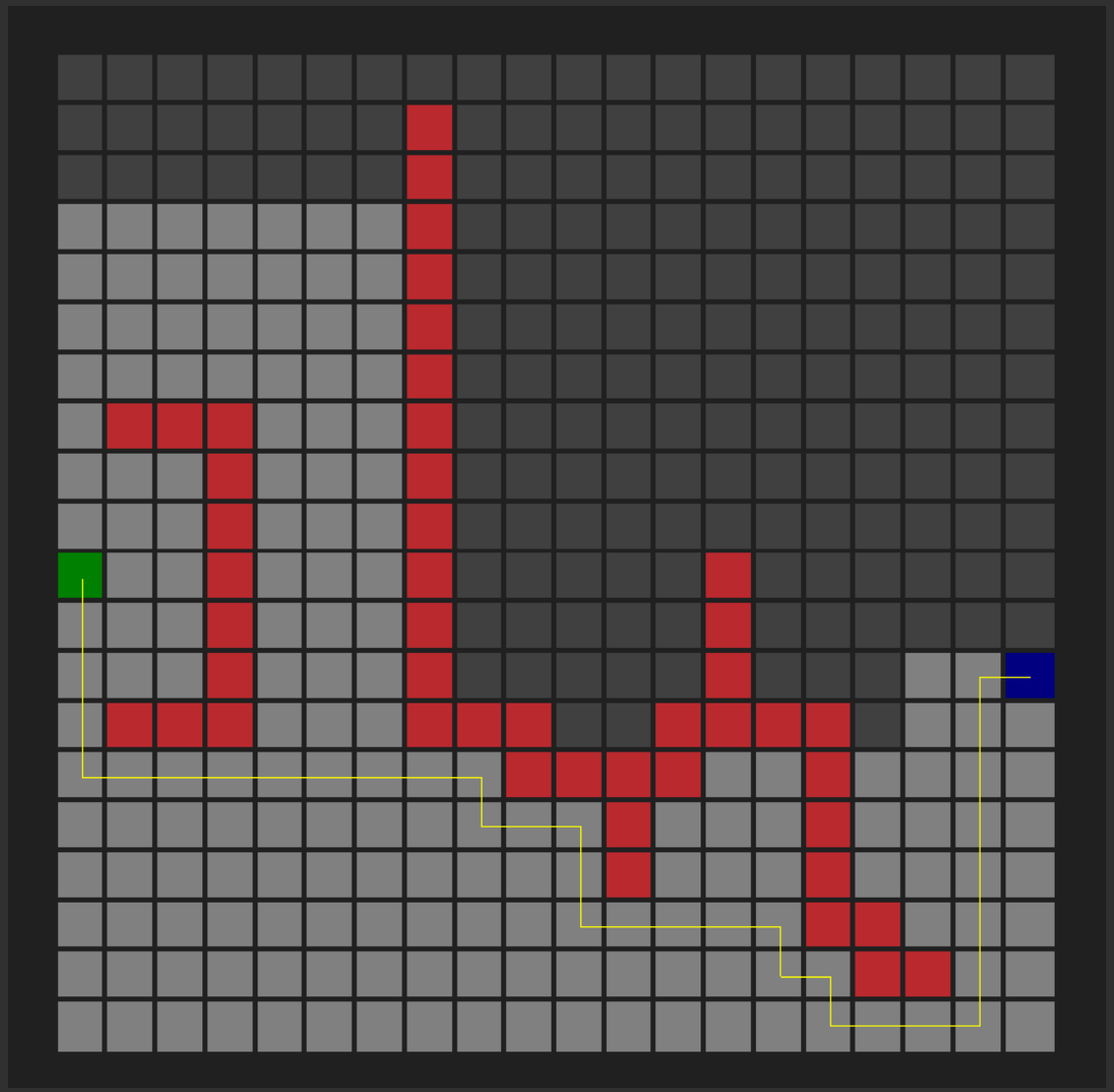
Algorytm A* odpowiada za znalezienie najkrótszej drogi z punktu A do punktu B, priorytetyzuje potencjalnie najlepszą drogę jako pierwszą do sprawdzenia. Ścieżka zostanie zawsze znaleziona pod warunkiem że istnieje. Jest stosowany głównie w dziedzienie sztucznej inteligencji i w grach komputerowych do imitowania inteligentnego zachowania.

W naszym programie ściany(przeszkody) są oznaczone kolorem czerwonym, punkt startowy kolorem zielonym, a punkt końcowy kolorem niebieskim.

Przykład 1

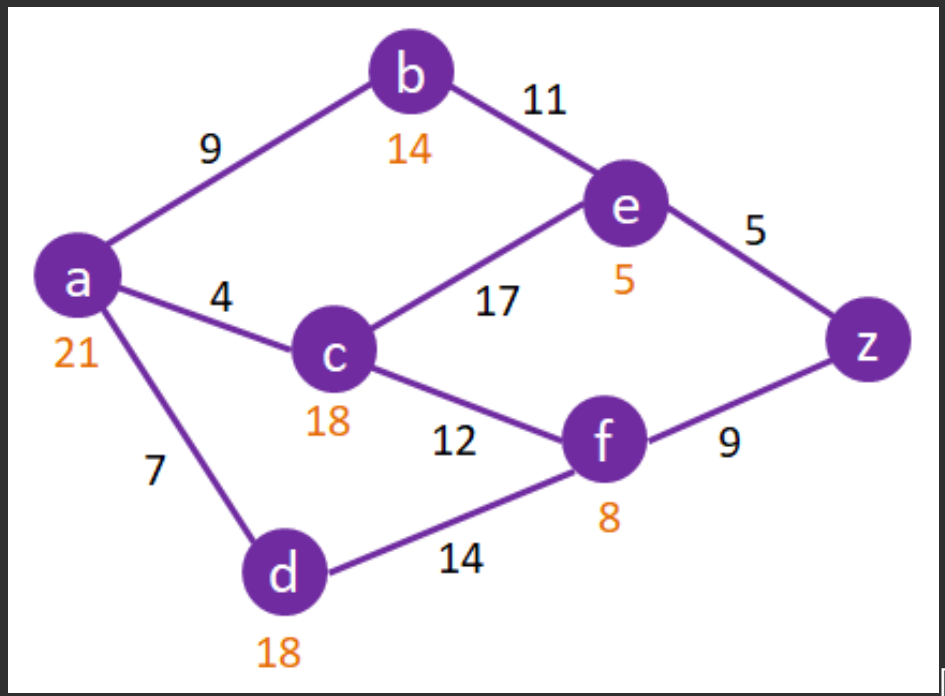


Przykład 2



Czym jest graf?

Graf jest to nieliniowa struktura matematyczna danych, służy do przedstawiania i badania relacji między obiektami. W uproszczeniu graf to zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków[1].



[2]

Figure: Graf z kosztami wykonania ruchu

Algorytm Dijkstry

Innym algorytmem grafowym jest algorytm Dijkstry, znajduje on wszystkie drogi do celu, mając wszystkie drogi można łatwo wybrać tą najkrótszą. Algorytm A* jest uogólnieniem algorytmu Dijkstry, który pozwala przeszukiwać tylko część grafu i znaleźć jedną najkrótszą drogę. Wymaga dodatkowej informacji wstępnej - heurystyki.

Heurystyka (gr. heuresis „odkryć”, heureka „znalazłem”)

Aby algorytm wiedział jak dobry jest kierunek poszukiwań, potrzebujemy funkcji określającej odległość stanu do rozwiązania. Taką funkcje nazywamy funkcją huerestyczną, ideał takiej funkcji zwraca wartość dokładnego kosztu przejścia z aktualnego punktu do rozwiązania. Dzięki takiemu rozwiązaniu praca algorytmu była by bezbłędna, lecz uzyskanie dokładnej wartości może być skomplikowane i czasochłonne, a w praktyce niemożliwe do zastosowania. Do znalezienia rozwiązania wystarczy aby funkcja nie zwracała wartości większej niż jej faktyczny koszt przejścia z danego punktu do końca[3].

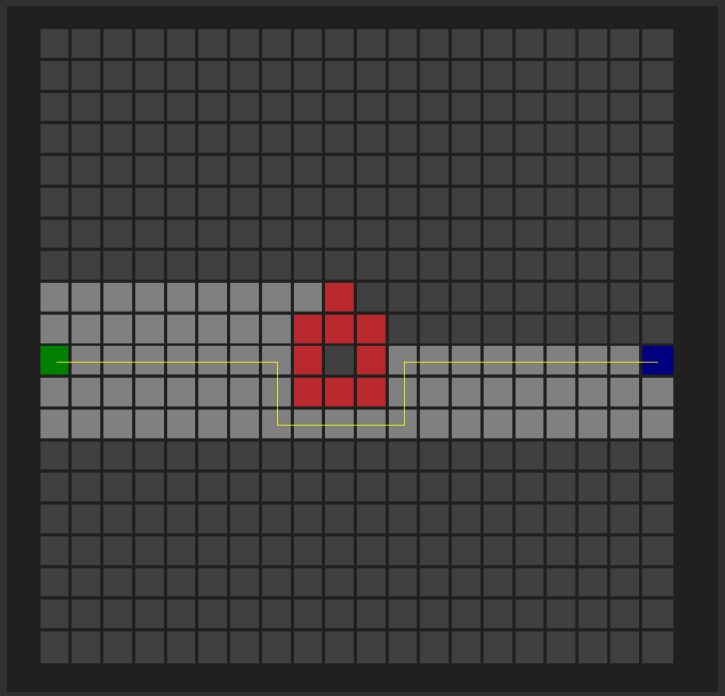
```
float Distance2Vecs(sf::Vector2i a, sf::Vector2i b)
{
    return sqrtf(float(a.x - b.x) * float(a.x - b.x) + float(a.y - b.y) * float(a.y - b.y));
}

float NodeDistance(GridNode* a, GridNode* b)
{
    return Distance2Vecs(a->nodePosition, b->nodePosition);
}

float Hueristic(GridNode* a, GridNode* b)
{
    return NodeDistance(a, b);
}
```

Zasada działania

Algorytm buduje graf, odkrywa pobliskie możliwości ruchu, kolejno bada i nadaje punkty posunięciom. Na podstawie nadanych punktów, wybiera to które zbliży go najbardziej do celu. Zdarza się że pomija niektóre pola, jeżeli obrał już prawidłową drogę. Pominięte pola są oznaczane ciemniejszym szarym kolorem.



Czas obliczeń

Czas potrzebny na wykonanie jednej kalkulacji przy siatce 20x20

```
Calculation took: 0.000147 seconds
Calculation took: 0.000140 seconds
Calculation took: 0.000140 seconds
Calculation took: 0.000139 seconds
Calculation took: 0.000143 seconds
Calculation took: 0.000134 seconds
Calculation took: 0.000137 seconds
Calculation took: 0.000136 seconds
Calculation took: 0.000137 seconds
Calculation took: 0.000136 seconds
Calculation took: 0.000142 seconds
Calculation took: 0.000144 seconds
Calculation took: 0.000148 seconds
Calculation took: 0.000134 seconds
```

Figure: t $= 1 * 10^{-4}[s]$

Bibliografia

- https://pl.wikipedia.org/wiki/Graf_(matematyka).
- https://www.101computing.net/a-star-search-algorithm/.
- https://xevaquor.wordpress.com/2015/03/09/gwiazda-wieczoru-algorytm-a-a-star/.
- https://pl.wikipedia.org/wiki/Algorytm_A*.
- https://pl.wikipedia.org/wiki/Algorytm_Dijkstry.

Koniec