Kraków, 7.01.2018r.

A* Algorithm

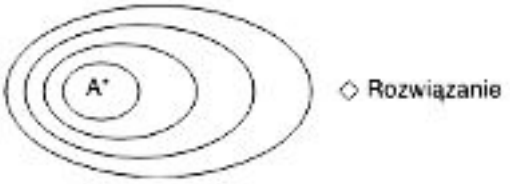
Projekt zaliczeniowy PYTHON

Michał Kucharski

A* jest algorytmem heurystycznym do znajdowania najkrótszych ścieżek w grafie ważonym. Jest to jeden z najlepszych oraz najpopularniejszych algorytmów używanych do znajdowania ścieżek. Stosowany w dziedzinie sztucznej inteligencji oraz w grach komputerowych do imitowania inteligentnego zachowania.

Jest on podobny do algorytmu dijkstry z tą różnicą że do znajdowania kolejnych wierzchołków wykorzystywana jest funkcja **f(x)=g(x)+h(x)**
g(x)- suma wag krawędzi, które należą już do ścieżki plus waga krawędzi łączącej aktualny węzeł z x
h(x) - funkcja przewidująca drogę od x do wierzchołka docelowego.

Algorytm A* tworzy ścieżkę, za każdym razem wybierając wierzchołek x z dostępnych w danym kroku wierzchołków tak, by minimalizować funkcję f(x). Wystarczy do priorytetu kolejki dodać wartość heurystyki, to sprawi że wierzchołki znajdujące się bliżej wierzchołka docelowego będą rozwijane wcześniej. A* przeszukuje przestrzeń stanów szybciej w kierunku rozwiązania niż w innych kierunkach.



W mojej pracy wykorzystałem graf ważony nieskierowany zaimplementowany przy pomocy słownika. Oprócz standardowych danych algorytm A* przyjmuję słownik zawierający informację o współrzędnych każdego z wierzchołków. Informacja ta jest potrzebna dla funkcji h(x) która zwraca odległość w lini prostej od x (aktualnie badanego wierzchołka) do wierzchołka docelowego.

Dane wejściowe funkcji Astar:
-graf reprezentowany poprzez słownik
np: graph = {"A":["B","C"], "B":["C","D"], "C":["D"], "D":["C"], "E":["C"], "F":[]}
-informacja na temat współrzędnych każdego z wierzchołków- reprezentacja słownikowa
np: {'a': [2, 2], 'c': [8, 3], 'b': [5, 4], 'e': [14, 6], 'd': [9, 5]}
-wierzchołek z którego zaczynamy
- wierzchołek na którym kończymy.

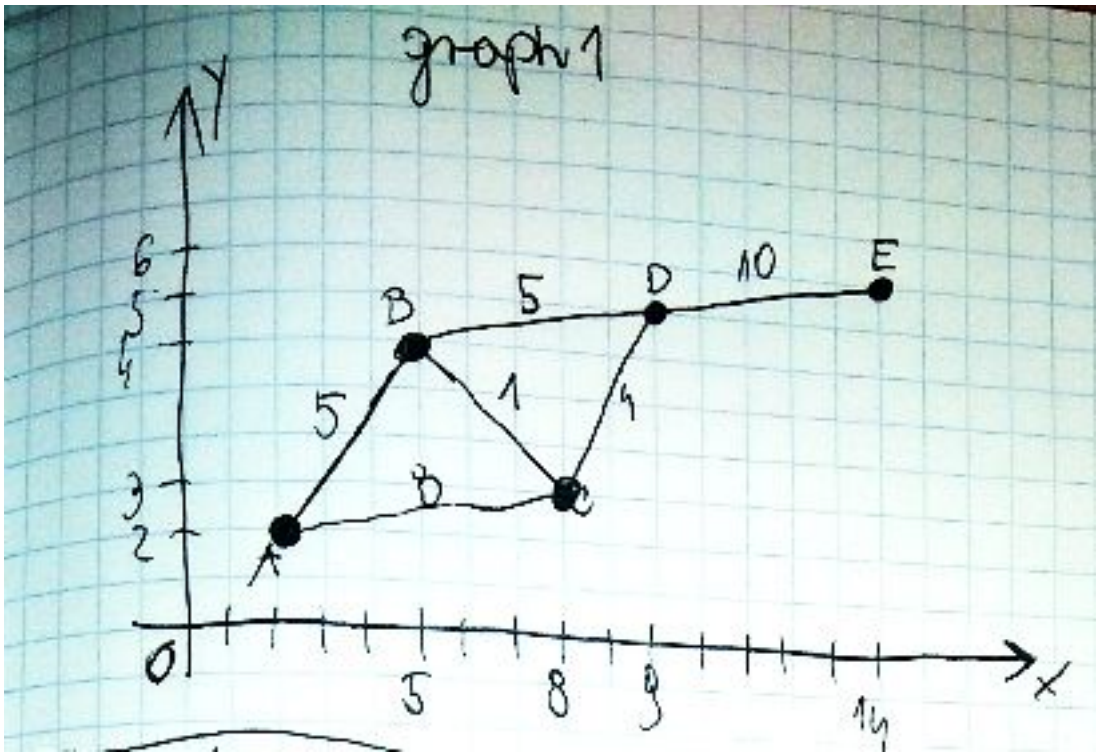
Funkcja **build_proper_path()** na podstawie danych słownikowych visited odbudowywuje właściwą ścieżkę start -> end.

Wyniki pracy programu oraz rysunki grafów dostępne na następnej stronie.

Wyniki pracy programu main_tests.py

////////////////////////////Graph1////////////////////////////

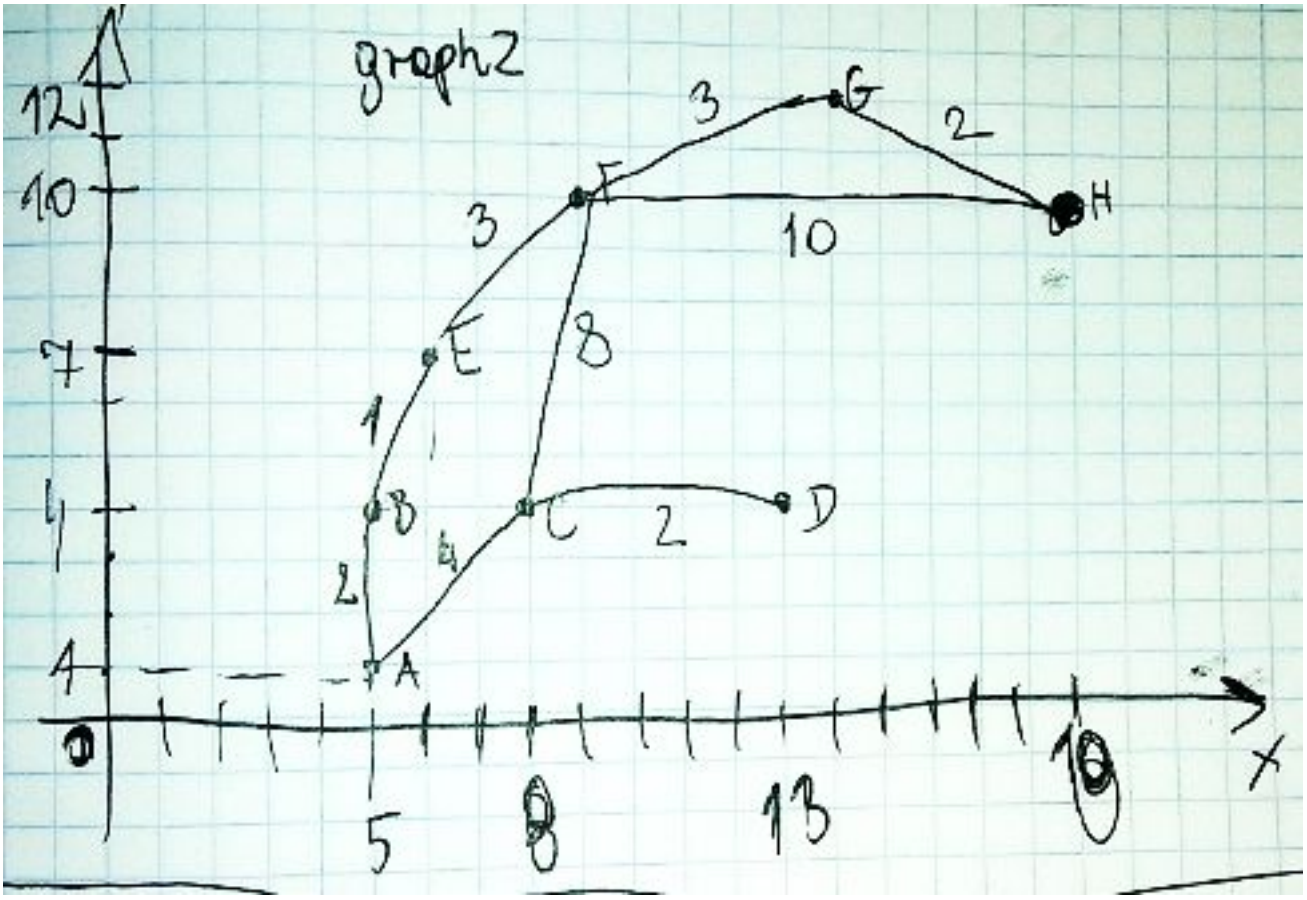
Path a -> e = ['a', 'b', 'd', 'e'] Distance = 20



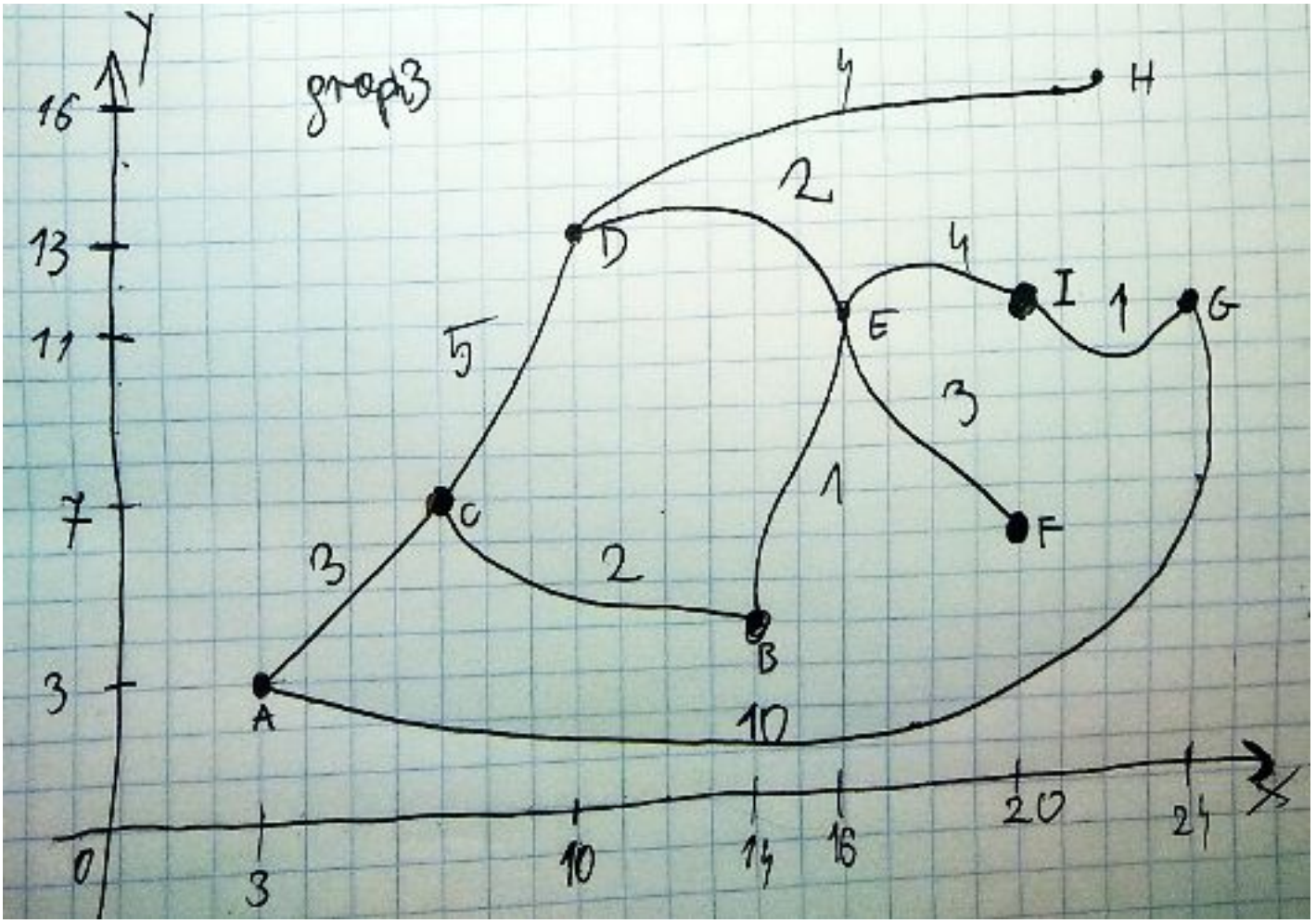
////////////////////////////Graph2////////////////////////////

Path a -> h = ['a', 'b', 'e', 'f', 'g', 'h'] Distance = 11

Path d -> h = ['d', 'c', 'f', 'g', 'h'] Distance = 15



////////////////////////////Graph3////////////////////////////
Path a -> g = ['a', 'g'] Distance = 10
Path b -> h = ['b', 'e', 'd', 'h'] Distance = 7
Path f -> h = ['f', 'e', 'd', 'h'] Distance = 9



Po przeprowadzonych testach widać że algorytm pracuje stabilnie oraz otrzymane wyniki są prawidłowe tz. program znajduję najktrótszą z możliwych ścieżek.

Źródła:
http://web.mit.edu/eranki/www/tutorials/search/
https://xevaquor.wordpress.com/2015/03/09/gwiazda-wieczoru-algorytm-a-a-star/
https://www.redblobgames.com/pathfinding/a-star/implementation.html
https://en.wikipedia.org/wiki/A*_search_algorithm