



ZŁOŻONOŚĆ OBliczeniowa algorytmów

PRZESZUKIWANIE GRAFU

Maciej Łacwik

WYKORZYSTANE ALGORYTMY

DJIKSTRA

- ZACHŁANNY

BELLMAN-
FORD

- ZACHŁANNY

A*

- HEURYSTYCZNY

DJIKSTRA

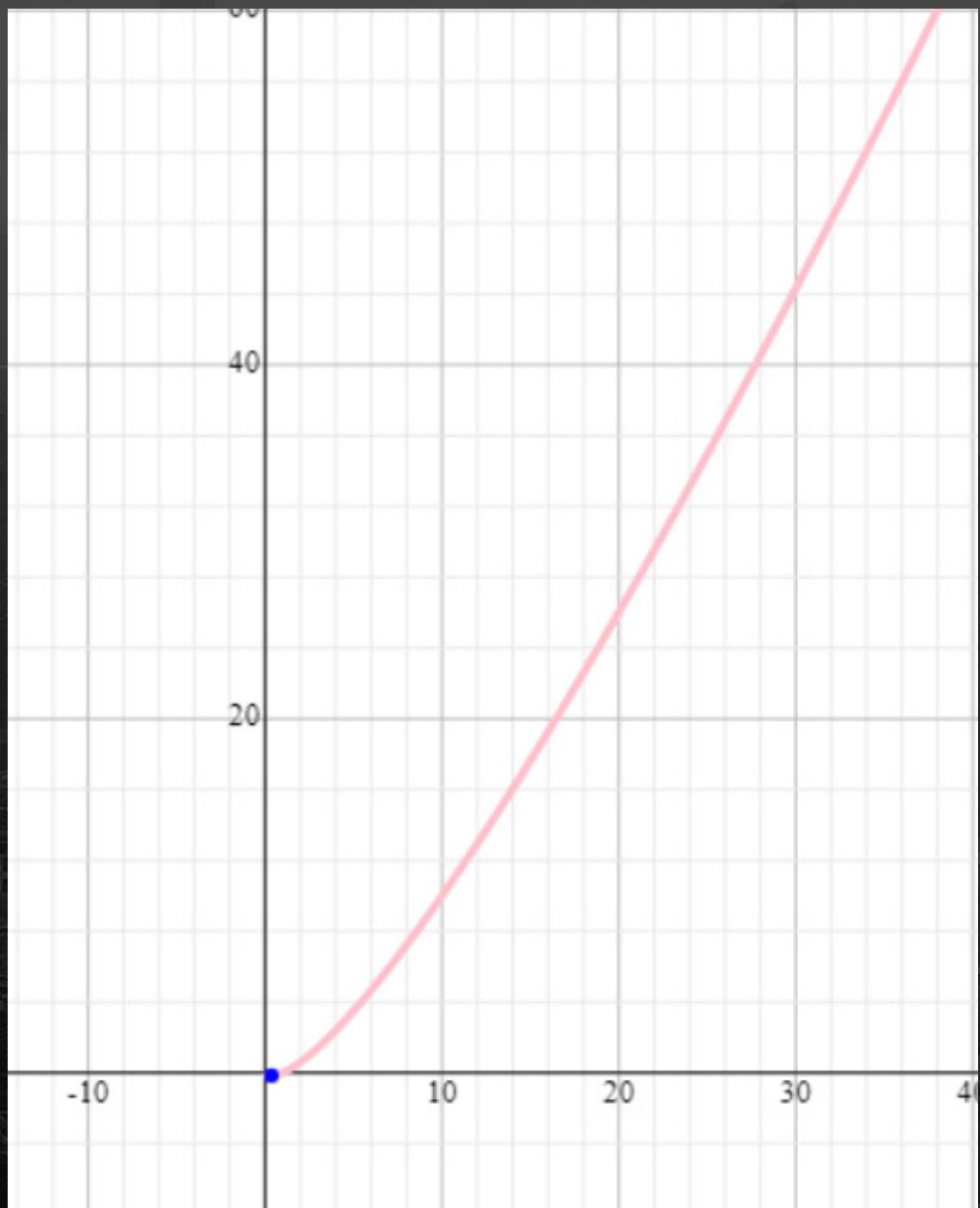
Algorytm Dijkstry jest przykładem algorytmu zachłannego.

Mając dany graf z wyróżnionym wierzchołkiem, algorytm znajduje odległości od źródła do wszystkich pozostałych wierzchołków. W tym jednak wypadku przerywa on działanie po znalezieniu wierzchołka oznaczonego jako końcowy

Złożoność:

TIME: $O(E \log V)$

MEMORY: $O(V \log V)$



BELLMAN-FORD



Algorytm Bellmana-Forda również jest przykładem algorytmu zachłanego.

Idea algorytmu opiera się na metodzie relaksacji.
W odróżnieniu od algorytmu Dijkstry, algorytm Bellmana-Forda działa poprawnie także dla grafów z wagami ujemnymi.
Za tę ogólność płaci się jednak wyższą złożonością czasową.

Złożoność:

TIME: $O(|V|^*|E|)$

MEMORY: $O(E+V*\log V)$

A*

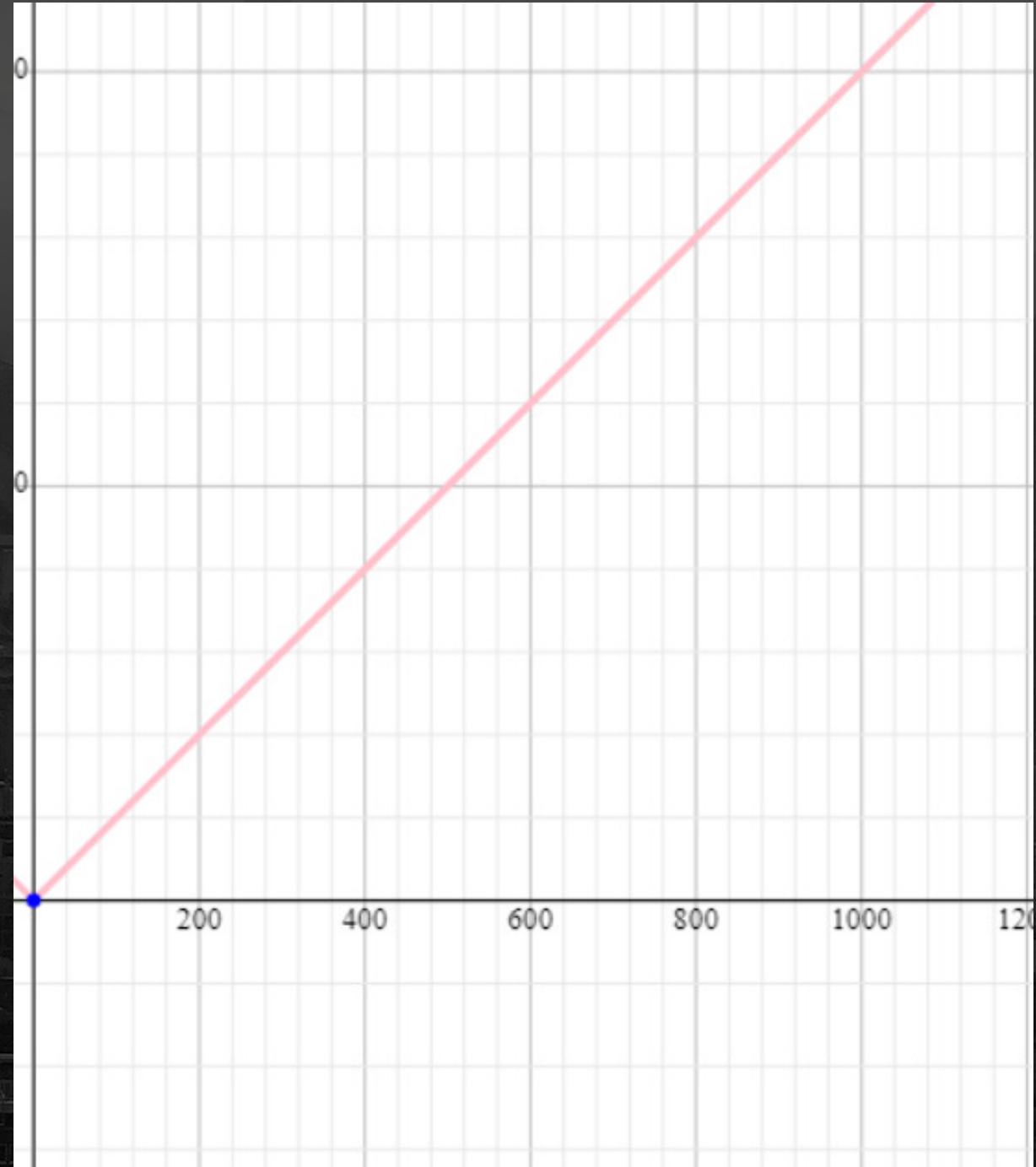
A* jest przykładem algorytmu heurystycznego.

Algorytm A* od wierzchołka początkowego tworzy ścieżkę, za każdym razem wybierając wierzchołek x z dostępnych w danym kroku niezbadanych wierzchołków tak, by minimalizować pewną funkcję $f(x)$.

Złożoność:

TIME: $O(|E|)$

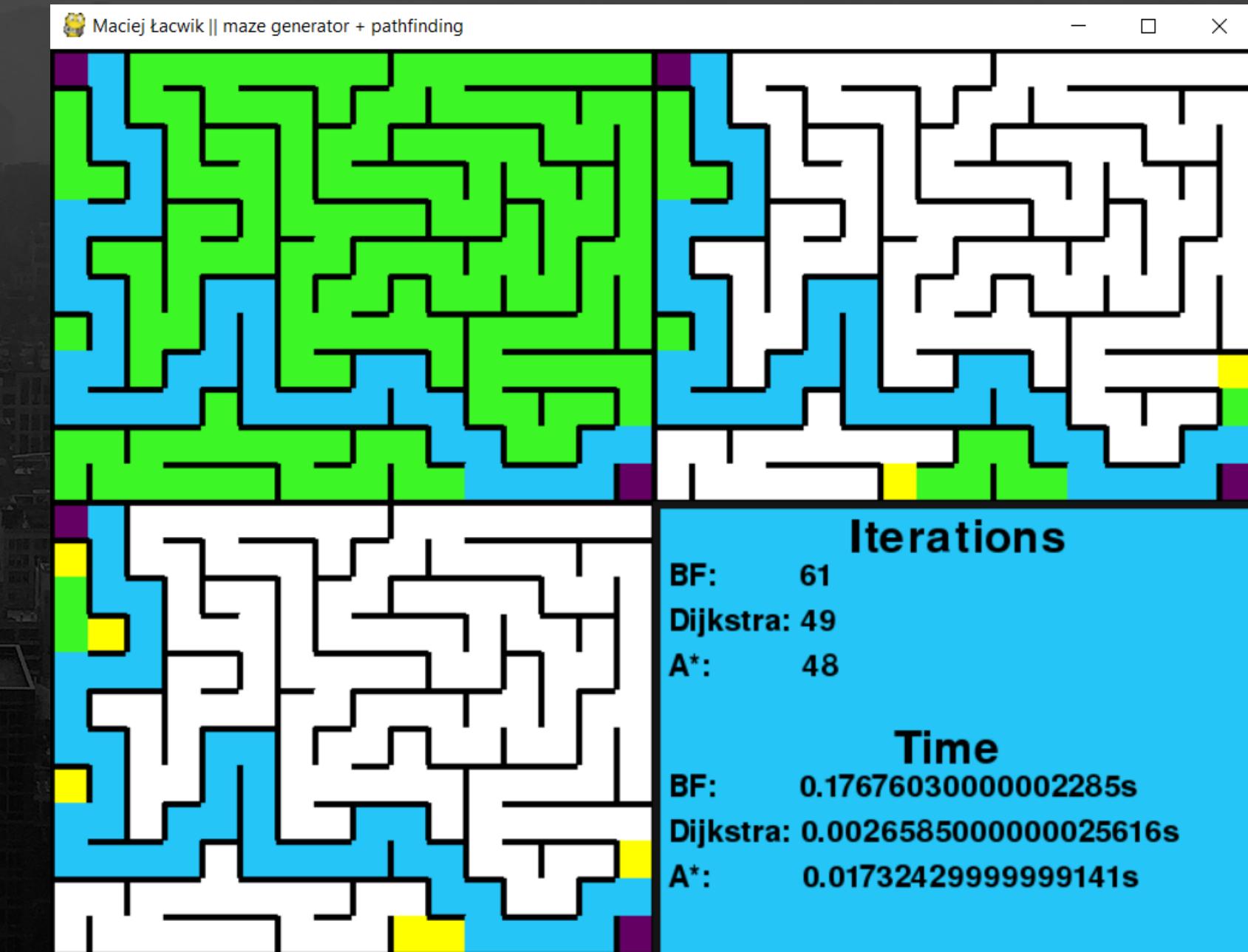
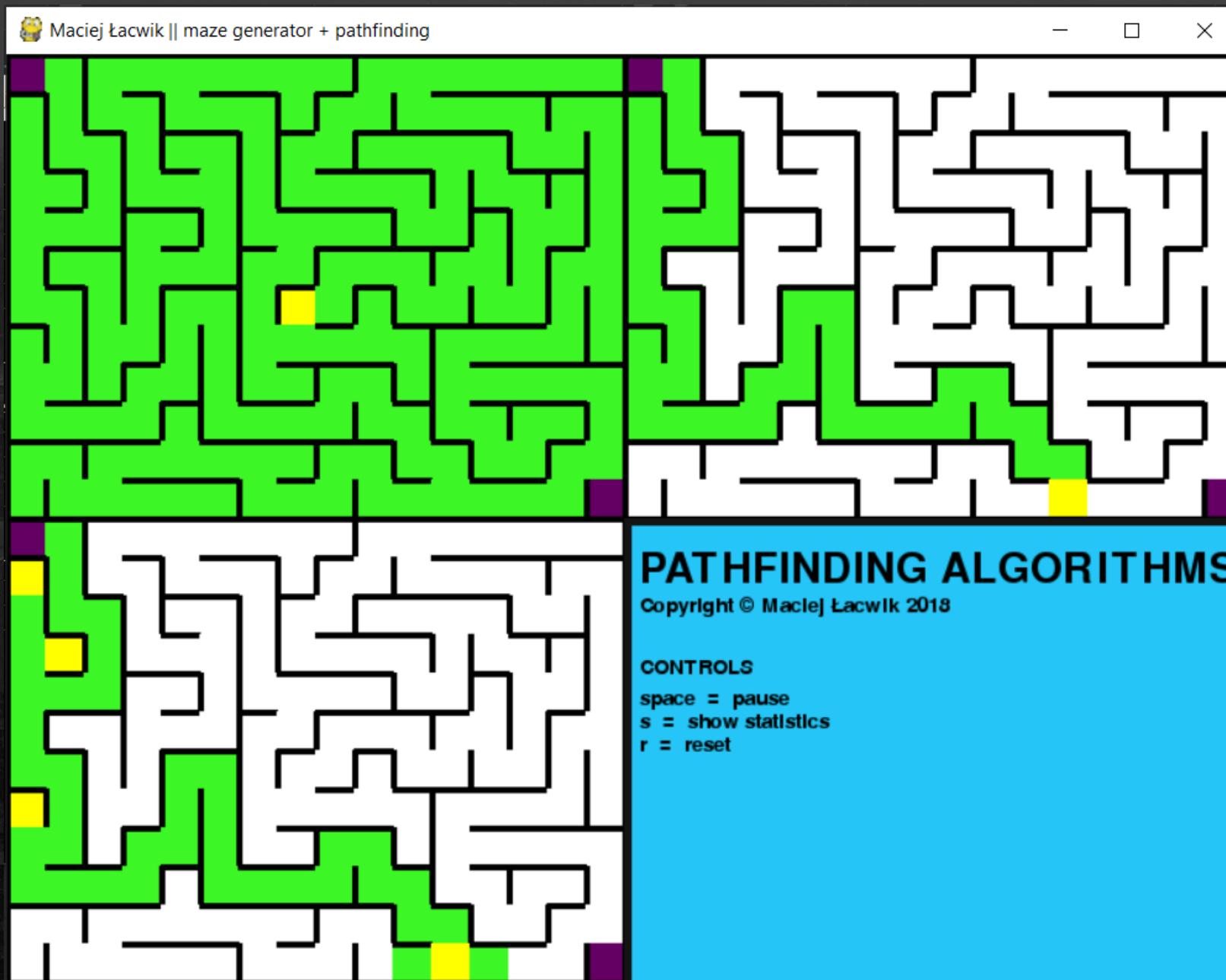
MEMORY: $O(|V|)$



OBSŁUGA PROGRAMU

Aplikacja została napisana w języku Python 3.7 z wykorzystaniem PyGame pod Visual Studio 2017. Po komplikacji programu (brak pliku wykonywalnego) zostanie wygenerowany labirynt i osadzony w trzech oknach. W lewym-górnym zostanie uruchomiony algorytm Bellmana-Forda, w prawym-górnym Djikstry natomiast w lewym dolnym A*. W prawym-dolnym rogu ekranu przedstawione są klawisze obsługiwane przez aplikację

OBSŁUGA PROGRAMU



OBSŁUGA PROGRAMU

wierzchołek fioletowy -> punkt startu/końca

wierzchołek zółty -> wierzchołek przeszukiwany w danej iteracji algorytmu

wierzchołek zielony -> element już uwzględniony w algorytmie

ZESTAWIENIE WYNIKÓW

ILOŚĆ WĘZŁÓW A CZAS DZIAŁANIA

Bellman-Ford

192	0.2332845000000212
320	0.5553692000000297
480	1.291350500000009
672	2.3565699000000015
896	4.078089799999997
1152	6.808401899999995

Dijkstra

192	0.0019824000000046027
320	0.008138299999977505
480	0.009013999999990752
672	0.0240483000001601
896	0.03323949999998756
1152	0.039002899999943

A*

192	0.01345929999999175
320	0.1479494000000301
480	0.1399613999999713
672	1.366387299999944
896	2.237468899999966
1152	3.1262103999999686

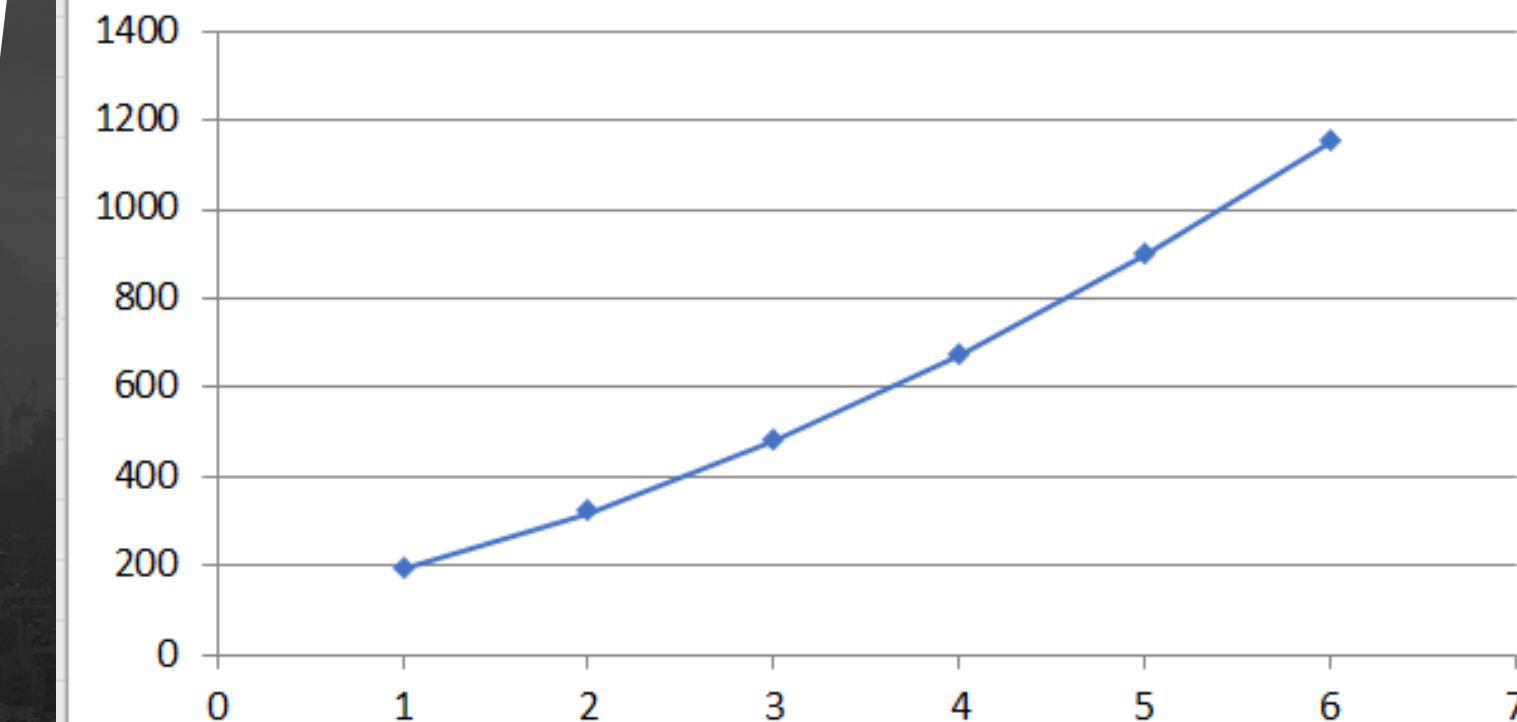
ZESTAWIENIE WYNIKÓW

ILOŚĆ WĘZŁÓW A CZAS DZIAŁANIA

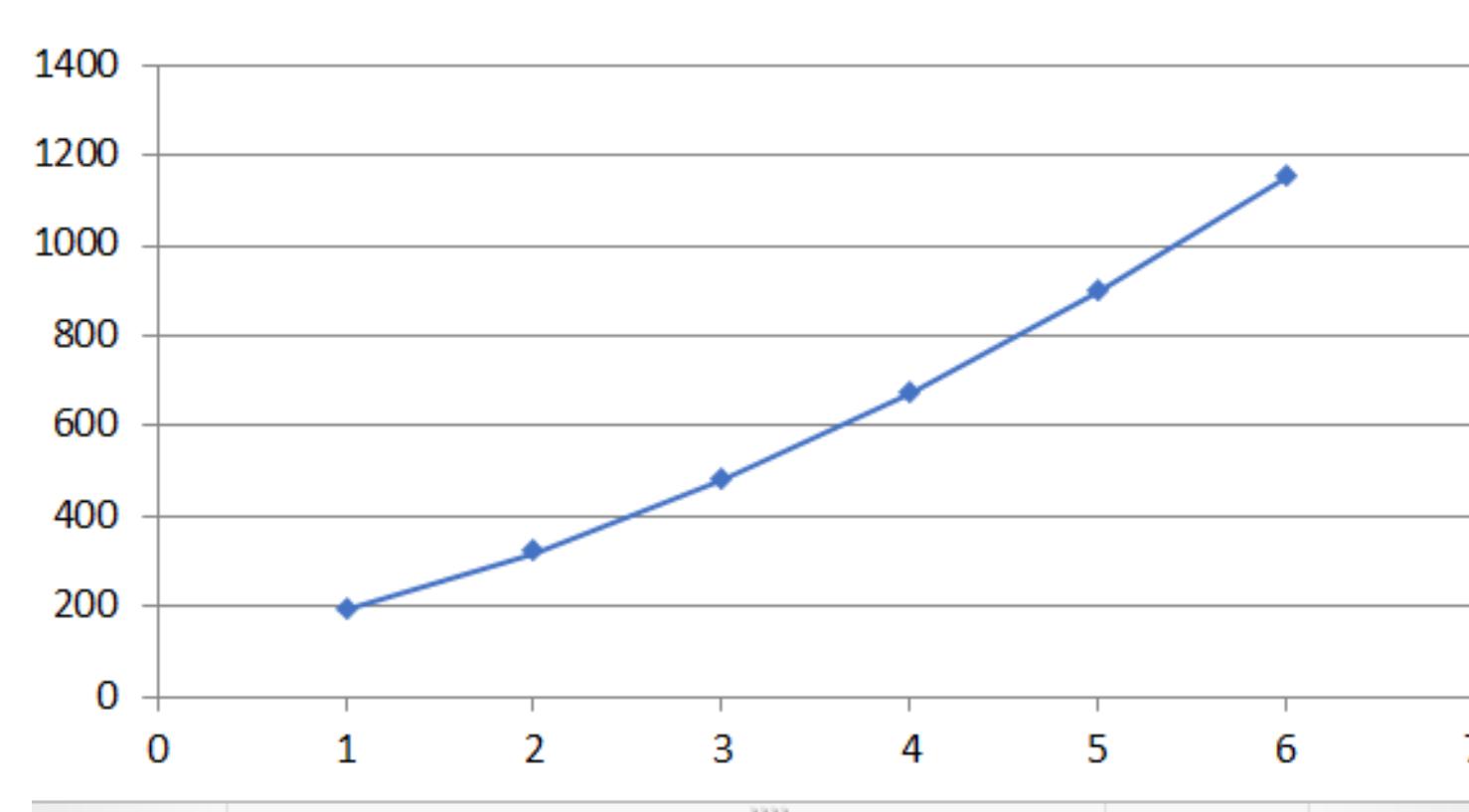
Bellman-Ford

Djikstra

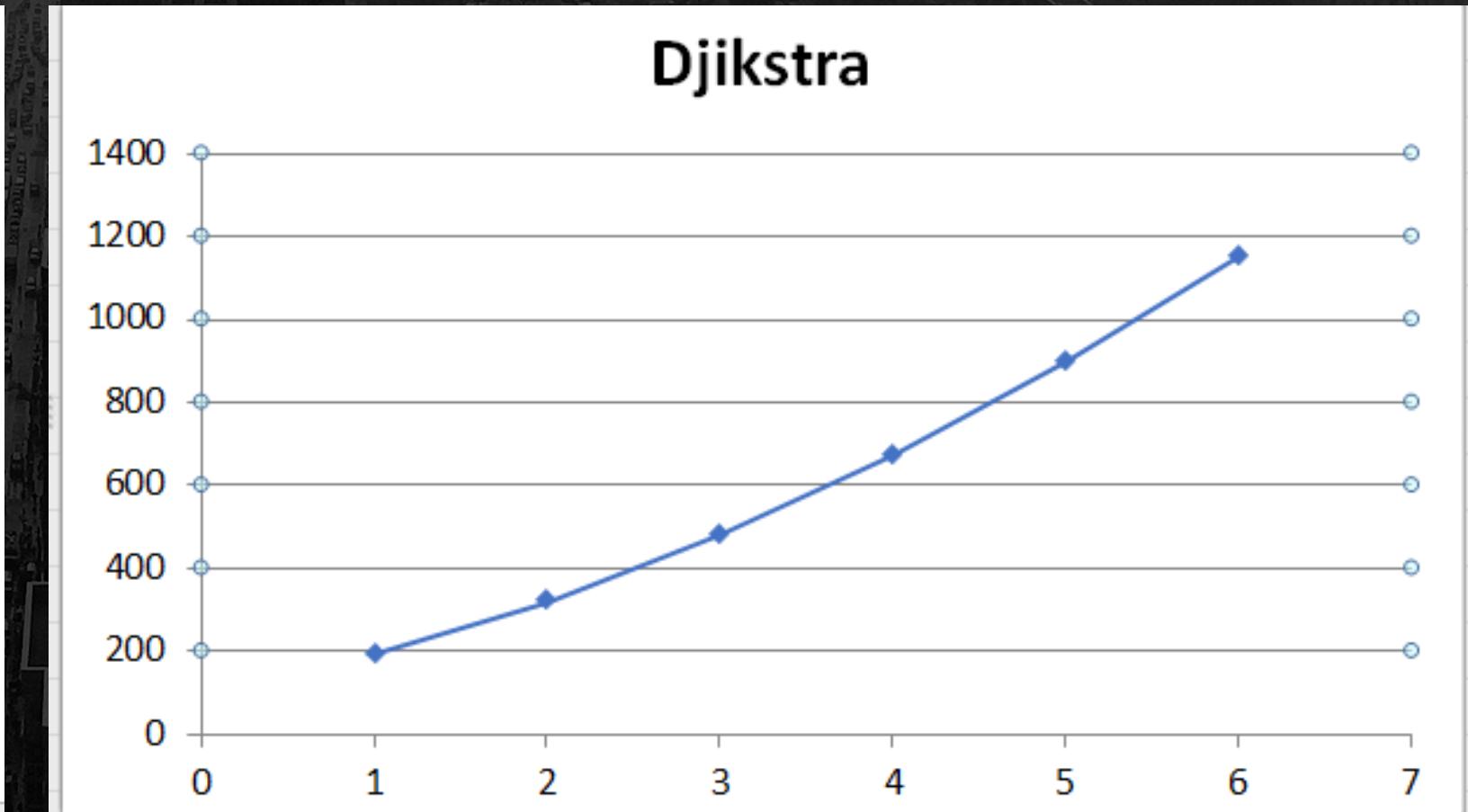
A*



Bellman - Ford



Djikstra



A* PROBLEMY

A* jako algorytm heurystyczny najlepiej sprawdza się w zróżnicowanym grafie ważonym. W danej implementacji koszt przejścia z jednego wierzchołka do każdego innego sąsiadującego wynosi zawsze 1, co tworzy problemy w zrozumieniu która ścieżka dojścia do celu jest tą optymalną. Kolejnym ważym czynnikiem wprowadzającym błąd obliczeniowy jest dobranie odpowiedniej heurystyki dla algorytmu (w tym przypadku euklidesowa) oraz co najważniejsze w tej implementacji - ograniczenie możliwych sąsiadów danego wierzchołka z 8 do 4.

A* PROBLEMY

ograniczenie sąsiadów wynika z dobranego problemu. Przedstawione algorytmy są zaprezentowane na losowo generowanym labiryncie, gdzie automat może się poruszać tylko w czterech kierunkach świata. Algorytm A* do optymalnego działania wymaga również przeszukania wierzchołków po skosie od danej lokalizacji. Z całości problemów wynika jeden fakt - zbyt duża liczba przeszukanych wierzchołków, co wprowadza błąd obliczeniowy

PODSUMOWANIE



IMPLEMENTACJA ALGORYTMÓW
ZOSTAŁA PRZEPROWADZONA
POPRAWNIE. POMIMO SPECYFICZNYCH
WARUNKÓW NARZUCONYCH PRZEZ
GENEROWANIE MAPY (GRAFU),
PORÓWNANIE DZIAŁANIA JEST DOŚĆ
DOBRZE WIDOCZNE.
UZYSKANE CZASY DZIAŁANIA PO
UWZGLĘDNIENIU MARGINESU
WYNIKAJĄCEGO Z OGRANICZEŃ
SPRZĘTOWYCH ORAZ JĘZYKOWYCH -
ZBLIŻAJĄ SIĘ DO CZASU
TEORETYCZNEGO