Informatyka, studia dzienne, inż. I st. semestr VI

Sztuczna Inteligencja i Systemy Ekspertowe 2014/2015

Prowadzący: mgr inż. Michał Pryczek piątek, 8:15

Data oddania: _____ Ocena: _____

Marcin Wawrzonowski 180729

Zadanie 1: Piętnastka

1. Wprowadzenie

Celem zadania było zapoznanie się, implementacja i ocena różnych strategii przeszukiwania przestrzeni stanów. W ramach zadania należało napisać program rozwiązujący układankę „Piętnastka". Program realizował zadanie przeszukując przestrzeń stanów z pomocą trzech strategii : przeszukiwania wgłąb (Depth-First Search – DFS), przeszukiwania wszerz (Breadth-First Search) oraz przeszukiwania algorytmem A* (A-Star). Jako, że A* jest algorytmem heurystycznym, została zaimplementowana heurystyka typu Manhattan.

Jeśli dany problem możemy przedstawić w postaci zbioru stanów oraz możemy określić przejścia pomiędzy tymi stanami, przeszukiwanie przestrzeni stanów jest idealną metodą jego rozwiązania. W szczególności, możliwe staje się szukanie rozwiązań optymalnych pod kątem odległości od stanu początkowego (liczonej w ilości przejść między stanami).

Przestrzeń stanów możemy przedstawić w postaci grafu. Poszczególnymi stanami są wierzchołki grafu, a przejściami między tymi stanami – krawędzie. W przypadku omawianego problemu, wierzchołki przechowują kolejne, unikalne stany układanki. Rozwiązaniem, którego poszukujemy jest stan, w którym kolejne pola „Piętnastki" są ustawione rzędami w kolejności rosnącej, a wolne pole znajduje się w prawym dolnym rogu. Dla tego rodzaju problemu, każdy wierzchołek grafu, jaki będziemy generować, będzie miał od dwóch do czterech stanów sąsiadujących. Zależy to od tego, gdzie znajduje się wolne pole układanki. Zastosowanymi strategiami są:

— *Przeszukiwanie wgłąb* (DFS) – w tej strategii zagłębiamy się w kolejne nieodwiedzone wierzchołki z wierzchołka, w którym aktualnie się znajdujemy.

— *Przeszukiwanie wszerz* (BFS) – odwiedzane są w pierwszej kolejności stany o odległości najmniejszej od stanu początkowego, tj. najpierw są odwiedzane wszystkie wierzchołki o odległości zerowej, potem 1, 2 i tak dalej, dopóki nie znajdziemy rozwiązania. W związku z tym, BFS gwarantuje zawsze znalezienie najkrótszego rozwiązania.
— *Przeszukiwanie algorytmem A-Star* – algorytm ten wprowadza funkcję kosztu i odwiedza najpierw wierzchołki o najniższym koszcie. Koszt jest obliczany przy użyciu heurystyki typu Manhattan. W naszym przypadku polega to na obliczeniu odległości sumy odległości w osi poziomej i pionowej, położenia każdego pola układanki od położenia tego pola w stanie będącym rozwiazaniem.

2. Działanie i implementacja

Program realizuje rozwiązywanie układanek o rozmiarze 4x4 pól, trzema różnymi, wymienionymi wyżej technikami. Użytkownik może wybrać maksymalny poziom trudności rozwiązywanej układanki (poziom trudności rozumiemy poprzez ilość kroków generatora, tworzącego pseudolosową układankę, od stanu początkowego – czyli poszukiwanego przez program; w ten sposób zapewniamy, iż układankę na pewno da się rozwiązać), ilość testów dla każdej techniki przypadającą na dany poziom trudności (przydatne, gdy chcemy uzyskać bardziej miarodajne pomiary) oraz maksymalną ilość kroku dla algorytmu. Ostatnia opcja jest niezbędna, by rozróżnić, po ilu krokach uznajemy, że algorytm nie znalazł poszukiwanego rozwiązania, aby uniknąć czekania w nieskończoność. Następnie program wykonuje testy i zapisuje wyniki do pliku. Wynikami są: Czas rozwiązywania układanki w milisekundach, (obliczony przy pomocy funkcji QueryPerformanceTimer biblioteki Windows.h), ilość kroków, jakie wykonał algorytm (czyli ilość wierzchołków, jakie odwiedził – miara użycia pamięci), długość ścieżki zwróconego rozwiązania oraz informacja o tym czy algorytm w ogóle znalazł rozwiązanie (czy przekroczył maksymalną liczbę kroków).

Wierzchołek grafu został zaimplementowany w postaci struktury zawierającej: tablicę[4][4] zmiennych typu unsigned char, obrazującą aktualny stan pól układanki, wskaźniki do swoich sąsiadów, wskaźnik do rodzica, wykorzystywany w celu odtworzenia ścieżki, po której poruszał się algorytm oraz inne zmienne pomocnicze. Całkowity rozmiar obiektu tej struktury to 48 bajtów.

Technika DFS została zaimplementowana przy użyciu stosu, zgodnie z [1]. W każdym kroku pętli zdejmujemy ze stosu znajdujący się na nim wierzchołek, sprawdzamy czy jest on naszym rozwiązaniem, jeśli nie – odkładamy na stos niesprawdzonych sąsiadów wierzchołka i przechodzimy do następnego kroku.

Technika BFS została zaimplementowana w podobny sposób, co DFS, z tą różnicą, iż zamiast stosu wykorzystujemy kolejkę FIFO [2].

Technika A-Star została zaimplementowana przy użyciu kolejki priorytetowej, na którą podobnie jak w BFS odkładamy wierzchołki. Kolejka zwraca nam zawsze węzeł o najniższym koszcie. Heurystyka kosztu jest generowana zgodnie z algorytmem Manhattan [4], przez wzięcie sumy odległości pozycji

pól od ich miejsc w rozwiązaniu. W A-Star musimy zadbać także o relaksację, czyli weryfikację i ewentualną zmianę kosztu dla sąsiadujących wierzchołków. Obliczana jest ona na nowo dla dzieci węzła, niezależnie od tego, czy je już odwiedziliśmy. W razie, gdy koszt jest niższy, wierzchołek jest usuwany z kolejki priorytetowej i dodawany na nowo, tak aby zachować sortowanie.

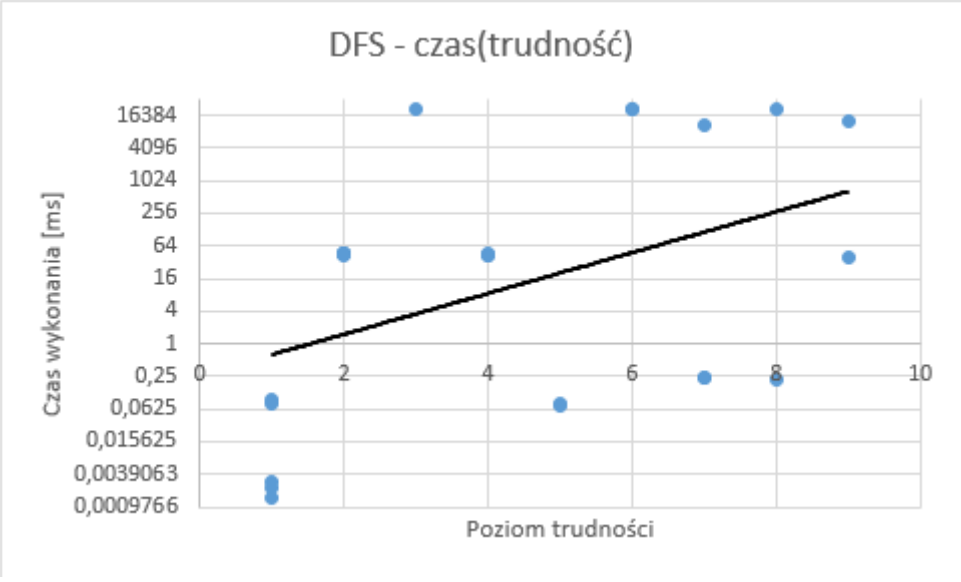
3. Wyniki

Wszystkie testy zostały wykonane na komputerze z procesorem Core i7 920 @ 3.80 GHz, 8 GB RAM i systemem operacyjnym MS Windows 7.

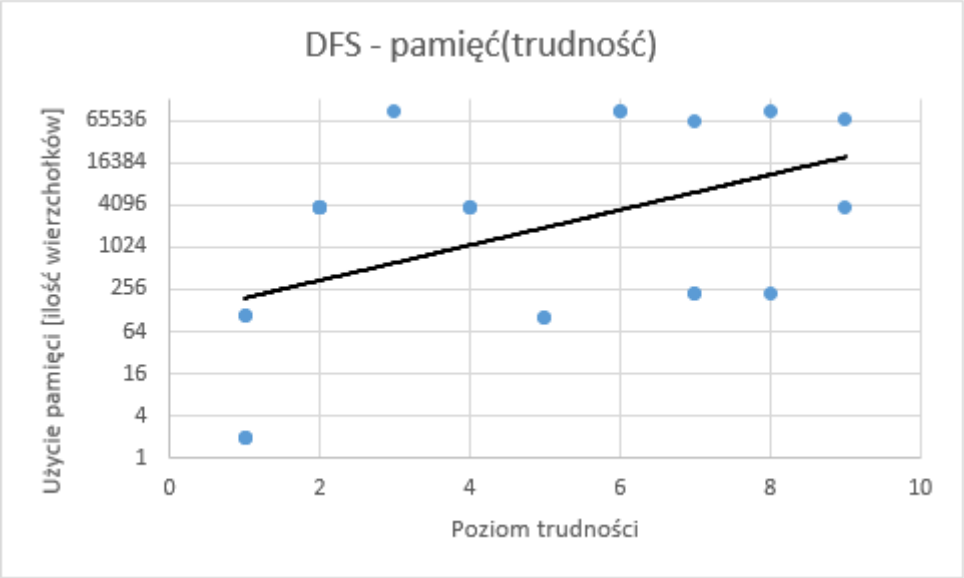
Przyjęto maksymalną liczbę kroków 100 000, tj. 4 800 000 bajtów pamięci zajętej przez wierzchołki grafu. Jeśli algorytm przekroczył tę liczbę, uznajemy, że się nie powiódł.

Dla każdego poziomu trudności i algorytmu wykonano 5 prób, za każdym razem generując losową układankę danego poziomu. Wartości podane na wykresach są wartościami średnimi z maksymalnie 5 prób. Przykładowo, jeśli algorytm zawiódł w 3 próbach na 5, podany został średni wynik tylko dla 2 prób zakończonych powodzeniem.

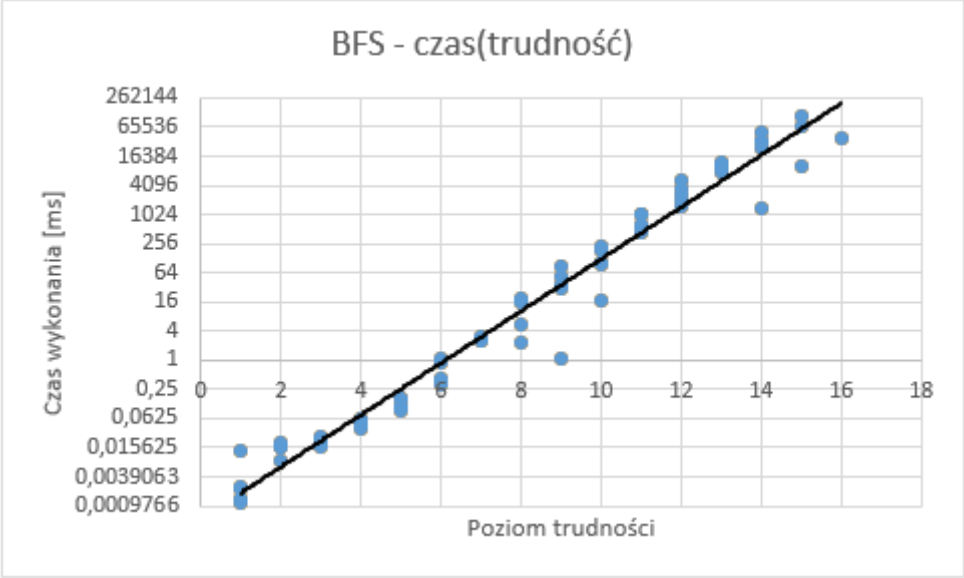
Jako współczynnik a rozumiemy współczynnik regresji liniowej dla danego wykresu.



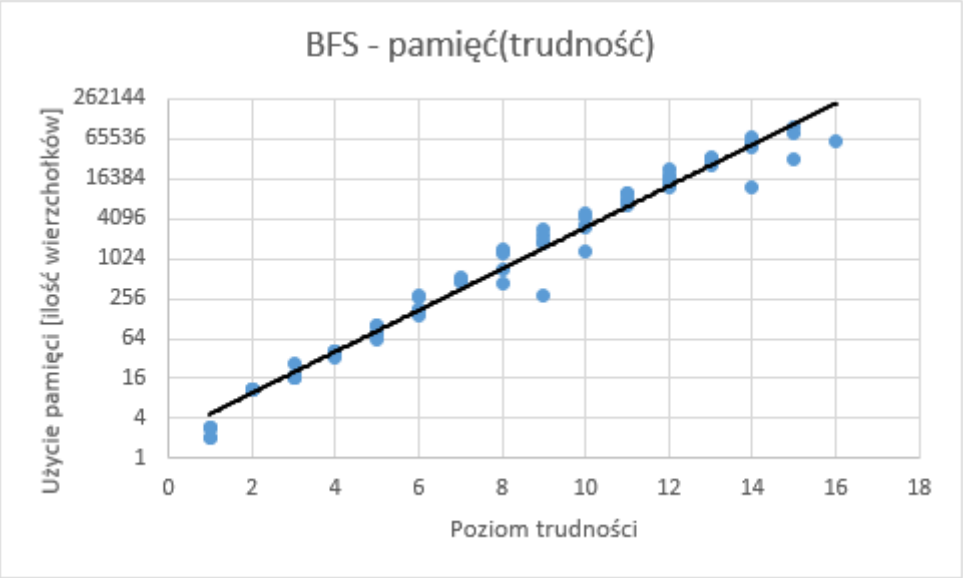
Rysunek 1. Współczynnik $a = 1163.06$



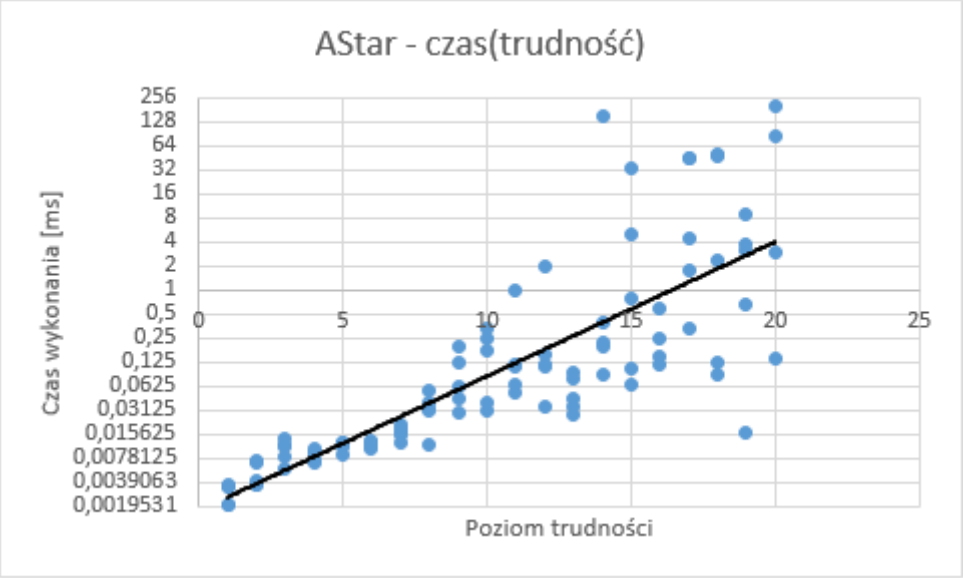
Rysunek 2. Współczynnik $a = 5316.96$



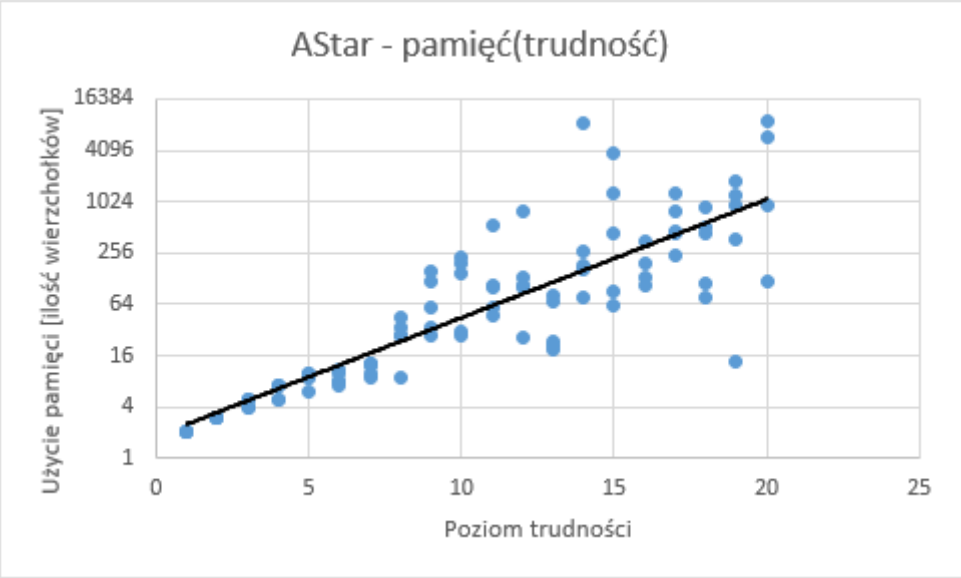
Rysunek 3. Współczynnik $a = 3.44$



Rysunek 4. Współczynnik $a = 2.05$



Rysunek 5. Współczynnik $a = 1.47$



Rysunek 6. Współczynnik $a = 1.38$

4. Wnioski

— Algorytm DFS sprawdzał się tylko przy bardzo niskich poziomach układanki. Na wyższych poziomach trudności, przeszukiwanie kończyło się niepowodzeniem lub długość podawanych rozwiązań była tysiące razy dłuższa niż optymalna długość. Ze względu na charakterystykę swojego działania, nie nadaje się on do rozwiązywania tego rodzaju problemów. Oczywiście zdarzały się wyjątki od tej reguły, lecz związane są one bardziej z losowością układanki niż z wydajnością algorytmu.

— Algorytm BFS jako jedyny zawsze wyszukiwał rozwiązania o najkrótszej, optymalnej długości. Całkiem dobrze nadawał się do podawania rozwiązań w układankach o trudności do 10-12 poziomu. Z każdym kolejnym poziomem czas przeszukiwania oraz użycie pamięci rosły wykładniczo, co można zaobserwować na wykresie. Wskazuje to na dalszą nieprzydatność algorytmu.

— Algorytm A-Star z heurystyką Manhattan sprawdził się nieporównywalnie lepiej niż jego poprzednicy. Ze względu na zastosowanie heurystyki, zarówno czas przeszukiwania jak i użycie pamięci na każdym poziomie trudności jest wielokrotnie niższe niż dla BFS lub DFS. Podczas testów zdarzały się jednak duże rozbieżności czasowe i pamięciowe na tych samych poziomach trudności, dla jednej próby wyniki potrafiły być dużo wyższe niż dla pozostałych, co wpłynęło na wartości średnie przedstawione na wykresach.

Literatura

[1] `http://en.wikipedia.org/wiki/Depth-first_search`

[2] http://en.wikipedia.org/wiki/Breadth-first_search
[3] http://heyes-jones.com/astar.php,
http://web.mit.edu/eranki/www/tutorials/search/
[4] http://stackoverflow.com/questions/8214063/
a-algorithm-with-manhattan-heureustic