Sprawozdanie z ćwiczenia laboratoryjnego

Tomasz Mikołajewski

2014

Spis treści

1 Wprowadzenie

Niniejszy dokument powstał w ramach przedmiotu Projektowanie Algorytmów i Metod Sztucznej Inteligencji. Jest on rezultatem przeprowadzonych ćwiczeń w laboratorium oraz pracy wykonanej w domu.

Ćwiczenie polegało na znalezieniu najkrótszej drogi na planszy. Jako utrudnienie zastosowano przeszkody na planszy, które musiały być omijane. Do wykonania postawionego zadania należało wykorzystać algorytm A*. Celem wykonanego zadania było również określenie skuteczności wymienionego algorytmu.

2 Kod programu

Program został oparty o wcześniej stworzone pliki zawierające kod potrzebny do wykonania *benchmark-u* podprogramu, czyli określenia czasu jaki potrzebny jest na jego wykonanie. Główny program zawiera funkcję otwierającą pliki z danymi oraz zapisującą rezultat operacji do pliku. Dodatkowo do programu wykonującego sprawdzenie algorytmów dodano metodę automatycznie tworzącą plik z danymi potrzebnymi do sprawdzenia

algorytmu. Wszystkie ustawienia można zmieniać w pliku *konfiguracja.hh*. Zdecydowanie ułatwia to prace przy analizie zadanej struktury.

3 Pomiary

Pomiary zostały podzielone na dwie części. Pierwszą z nich było wykonanie symulacji na małych planszach i sprawdzenie, jak algorytm poradzi sobie w określonych sytuacjach. W tej części otrzymane wyniki były porównywane z wizualizacją planszy wyświetlającą się na ekranie, która została wygenerowana przez program. Druga część pomiarów polegała na próbie określenia złożoności algorytmu. W tej części testowano zależność pomiędzy długością ścieżki, ilością odwiedzonych elementów, a czasem trwania algorytmu. Pomiary te zostały wykonane dla dużych plansz.

4 Wyniki pomiarów i obserwacje

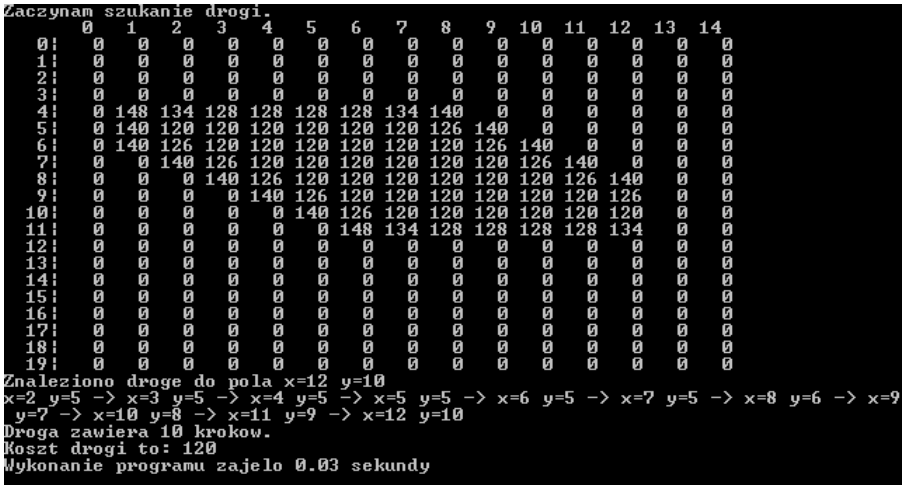
Po przeprowadzeniu serii pomiarów otrzymano wyniki przedstawione w tabelach. Na podstawie wyników utworzono wykresy zamieszczone poniżej. Na końcu sprawozdania znajdują się wnioski wyciągnięte z załączonych wyników pomiarów.

4.1 Przykłady działania algorytmu

4.1.1 Wyszukiwanie drogi do celu bez przeszkód

W celu sprawdzenia algorytmu wykonano wizualizację działania programu. W niniejszym punkcie przedstawiono wynik prostego przykładu polegającego na przemieszczeniu się pomiędzy punktem startowym i docelowym. Jest on zaprezentowany na Rysunku 1. Ten rozdział służy również zapoznaniu Czytelnika z przyjmowanymi oznaczeniami.

Po kilku próbach przyjęto standard wyświetlania wyniku jak na wcześniej wspomnianym Rysunku **??**. Prezentuje on w postaci tabeli planszę, na której odbywa się wyszukiwanie najkrótszej drogi. Na górze możemy znaleźć numeracje kolumn,a po lewej stronie numerację wierszy. Załóżmy zgodnie z intuicją, że kolumny to współrzędna x, a wiersze to współrzędna y. Pozostałe pola to konkretne węzły-punkty na planszy. W każdym z punktów wyświetlany jest **prognozowany koszt drogi** z punktu początkowego do pola docelowego, wiodącej przez omawiane pole. Prognoza ta jest sumą kosztu dotarcia do omawianego pola z pola startowego i wartości obliczonej za pomocą heurystyki przewidującej koszt drogi z omawianego pola do pola docelowego. W polach dla, których algorytm nie przeprowadzał obliczeń znajdują się 0 (wartość niemożliwa do otrzymania jeżeli pole celu nie jest polem początkowym).



Rysunek 1: Wyznaczenie prostej drogi do celu

Celem postawionym w prezentowanym przykładzie było znalezienie najkrótszej drogi z punktu x=2, y=5 do punktu o współrzędnych x=12, y=10. W punkcie startowym wpisany jest przewidywany koszt dotarcia do celu jeżeli na drodze nie będzie przeszkód. Następnie algorytm wykonał przeszukanie najbliższych pól, aby znaleźć pola o najmniejszej wartości przewidywanego kosztu i przemieścił się do pola x=3, y=5 oraz x=3, y=6. Tam algorytm sprawdzający przewidywane koszty dla pól sąsiednich został powtórzony. Widać, że istnieje kilka dróg prowadzących do celu o tym samym koszcie (przez pola o wartości 120). Poniżej tabeli został zaprezentowany wynik działania algorytmu, czyli jedna z najkrótszych dróg oraz ilość jej kroków oraz koszt. Jak widać heurystyka działa poprawnie, ponieważ końcowy koszt jest równy kosztowi przewidywanemu na początku.

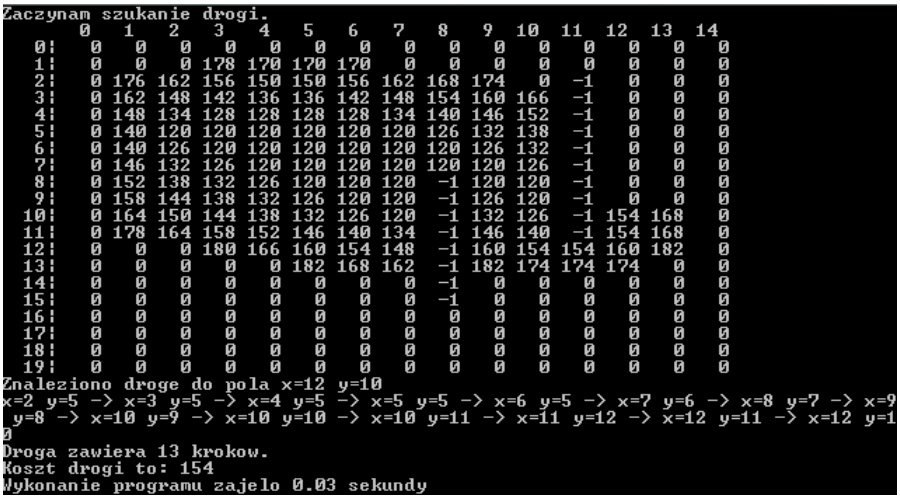
Warto zauważyć, że w przypadku przedstawionym na Rysunku 1 algorytm sprawdził całkiem sporą ilość pól. Jest to spowodowane ilością możliwych najkrótszych dróg. Sytuacja zawarta w tym punkcie sprawozdania jest wersją pesymistyczną dla wyszukiwania drogi pomiędzy dwoma punktami jeżeli na najprostszej drodze nie występują przeszkody. Dzieje się tak, ponieważ algorytm porusza się *na skos* przez planszę.

4.1.2 Szukanie drogi przy występowaniu przeszkód

Jeżeli na początkowo przewidywanej drodze pojawią się przeszkody, algorytm powinien je ominąć. W tej części sprawozdania zbadano działanie programu przy występujących przeszkodach.

W ramach sprawdzenia algorytmu zasymulowano sytuację widoczną na Rysunku 2. Do wcześniej omawianej sytuacji dodano dwie *ściany* widoczne w postaci wartości -1.

Zadanie wyznaczone dla algorytmu było analogiczne jak poprzednio to znaczy wyznaczenie najkorzystniejszej drogi z punktu x=2 y=5 do punktu x=12 y=10. Jak łatwo zauważyć program przewidywał koszt drogi taki sam jak otrzymany w poprzednim pod-

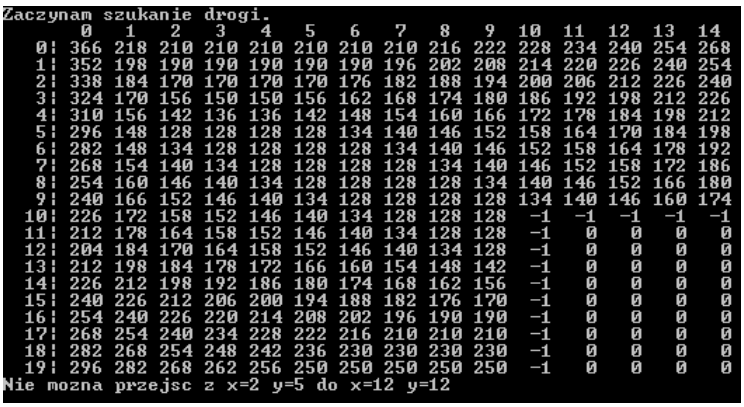


Rysunek 2: Omijanie przeszkód

punkcie aż nie dotarł do przeszkody. Pierwszą z przeszkód można ominąć polami x=8 y=6 oraz x=8 y=7. Jednakże po osiągnięciu kolumny 10 okazało się, że nie istnieje droga prowadząca do celu o koszcie wynoszącym 120. Algorytm zaczął więc sprawdzać inne możliwe drogi kierując się priorytetem najmniejszego kosztu. Jak widać najkrótsza z dróg wiedzie między innymi przez pole x=11 y=12, a jej koszt wynosi 154.

4.1.3 Sytuacje przy których nie istnieje droga spełniająca warunki zadania

W tym punkcie zostanie przedstawiona sytuacja w której nie można wyznaczyć drogi pomiędzy punktami. Jako przykład wykorzystano sytuację przedstawiono na Rysunku 3. Widać, że część planszy została odgrodzona od reszty przeszkodami.



Rysunek 3: Sytuacja bez wyjścia

Algorytm miał za zadanie znaleźć ścieżkę pomiędzy x=2, y=5 i x=12, y=12. Jak widać zadana droga nie istnieje. Jest to wersja pesymistyczna działania algorytmu, ponieważ nie licząc pól oddzielonych od reszty planszy musi on wykonać przeszukanie zupełne. Istnieje

Rysunek 4: Złożoność obliczeniowa

Kroków	Kroków po skosie	Kroków po prostej	Odwiedzono	Czas [s]
252	93	159	15041	0.06
272	132	140	18754	0.1
313	157	156	24807	0.17
349	235	114	27141	0.2
417	124	293	36751	0.25
393	242	151	36937	0.31
502	131	371	49105	0.38
554	125	429	54181	0.46
585	400	185	74587	0.92
628	406	222	90762	1.24
642	315	327	103649	1.41
686	306	380	116968	1.72
779	273	506	138919	3.01
981	653	328	215167	4.61

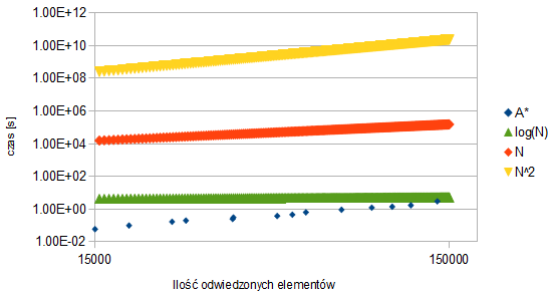
możliwość poprawy działania algorytmu w sytuacjach, gdy plansza ma skończony wymiar i określony kształt. Należałoby zmodyfikować algorytm, aby szukał rozwiązania tylko do momentu, aż *nie wydzieli* obszaru z planszy wraz z polem docelowym. W tym wypadku algorytm zakończyłby zadanie po sprawdzeniu pól o prognozowanym koszcie nie większym od 250. Jednakże opisane zagadnienie wykracza poza zasady działania algortymu A* i nie było przedmiotem badań w omawianym zadaniu.

4.2 Złożoność algorytmu przy wyszukiwaniu drogi bez przeszkód.

W celu zbadania złożoności algorytmu wykonano testy na planszy o rozmiarach 1000 na 1000. Przeprowadzano eksperyment dla różnych ścieżek. Otrzymane wyniki zebrano w tabeli Rysunek 4.

Na podstawie danych wykreślono wykres zależności czasu wykonywania algorytmu od ilości odwiedzonych pól **??** (Rysunek 6)

Warto zauważyć, że ilość odwiedzanych pól jest w przybliżeniu równa iloczynowi ilości kroków po skosie i kroków po prostej. Oznacza to, iż powstająca figura jest równoległobokiem o polu równym ilości odwiedzanych pól. Konsekwencją tego jest fakt, że przy określonej długości drogi najmniej korzystnym jest wyszukiwanie drogi w której należy wykonać taką samą ilość kroków po skosie jak po prostej (przypadek przedstawiony w rozdziale 4.1.1)



Rysunek 5: Złożoność algorytmu

5 Wnioski

Na podstawie danych i wykresów można wyciągnąć następujące wnioski:

- algorytm A* ma złożoność liniową
- omawiany algorytm może poradzić sobie z omijaniem przeszkód znajdujących się na zadanej planszy
- jeżeli nie istnieje droga prowadząca pomiędzy dwoma punktami to algorytm przeszuka zdecydowaną większość planszy (patrz punkt 4.1.3)
- najbardziej niekorzystnym wariantem wzajemnego położenia pola początkowego i pola końcowego dla badanego algorytmu jest położenie powodujące znalezienie drogi, w której liczba kroków po skosie jest równa liczbie kroków po prostej (lub innymi słowy jeśli po podzieleniu różnicy pomiędzy współrzędnymi x początku i końca oraz różnicy pomiędzy współrzędnymi y początku i końca otrzymamy wynik 2 lub 0.5)
- algorytm A* znajduje najlepszą drogę z możliwych
- A* można wykorzystać w grach komputerowych, w których należy wyznaczyć drogę pomiędzy dwoma punktami na planszy. Jeżeli z danego punktu będzie wyznaczana droga do wielu celów i podczas gry nie będzie wyliczana powtórnie to warto się zastanowić, czy nie lepiej zaimplementować algorytm Dijkstry. W innych przypadkach algorytm A* jest najlepszym z możliwych.