

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

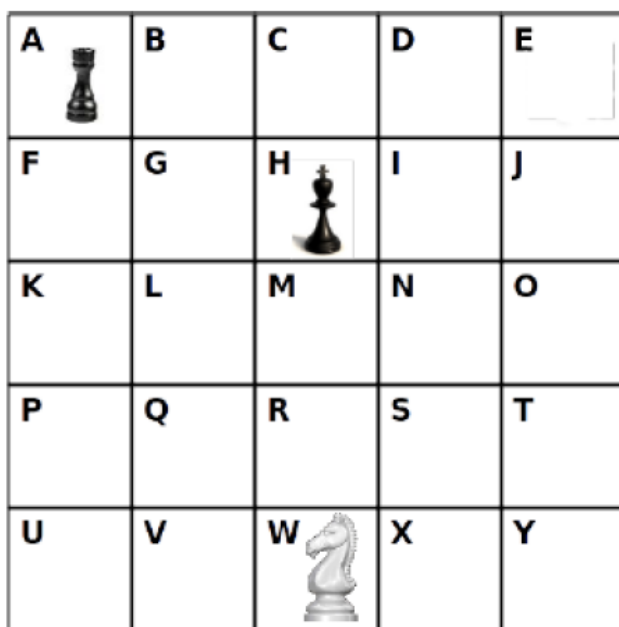
Raport z wykonania projektu - Projekt 4 "Koszmar Króla"

Patryk Szelewski 241490

08.06.2019r.

1 Działanie programu

Program ma za zadanie poszukiwać rozwiązania problemu zbitia czarnego króla umieszczonego na polu H zmodyfikowanej planszy do szachów, za pomocą białego skoczka umieszczonego na polu W. W problemie uwzględnić należy także wieżę umieszczoną na polu A planszy. Skoczek nie może poruszyć się na pole zagrożone biciem wieży.

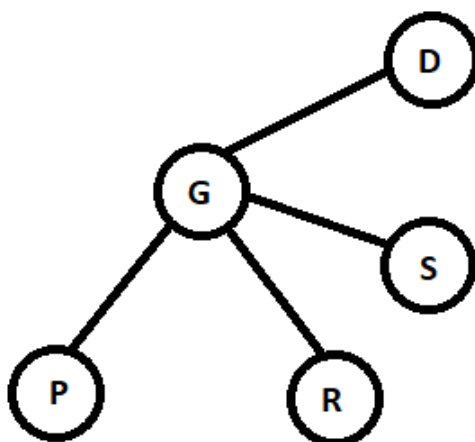


Rysunek 1: Rozważana plansza

2 Rozwiązanie problemu

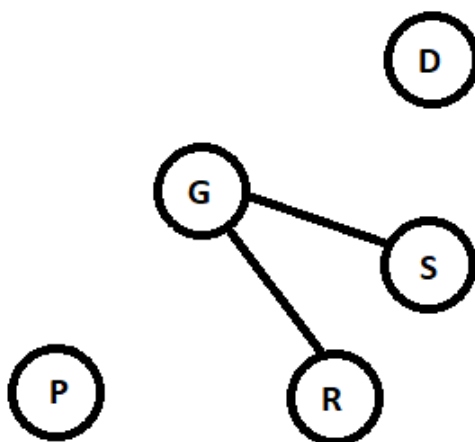
2.1 Generacja grafu

Pierwszym krokiem w celu znalezienia rozwiązania problemu, jest wygenerowanie grafu obrazującego powiązania między poszczególnymi polami na planszy. Pola ze sobą powiązane pozwalają na przejście z jednego do drugiego ruchem skoczka. Nie uwzględniając położenia wieży na planszy, tak wygląda część utworzonego grafu, pokazująca powiązania pola G:



Rysunek 2: Część graf ukazująca powiązania pola G bez uwzględniania wieży na polu A

Po uwzględnieniu położenia wieży w polu A, można znacząco zredukować ilość krawędzi w grafie:



Rysunek 3: Część graf ukazująca powiązania pola G z uwzględnieniem wieży na polu A

Jak widać, pola P oraz D nie są już powiązane z polem G, ponieważ leżą one w polu bicia wieży.




2.2 Algorytm przeszukiwania w głąb DFS

Pierwszym sposobem na przeszukanie grafu w poszukiwaniu rozwiązania jakim jest znalezienie ścieżki od pola startowego skoczka (W) do pola na którym znajduje się król (H) jest wykorzystanie algorytmu przeszukiwanie grafu w głąb DFS (Depth First Search). Pozwala on na odwiedzenie do każdego wierzchołka grafu, o ile graf jest spójny. Śledząc kolejne odwiedzane wierzchołki grafu, śledzić możemy sekwencję wierzchołków jakie odwiedzał algorytm w celu dojścia do zamierzonego przez nas wierzchołka.

2.2.1 Wyniki

Za pomocą algorytmu przeszukiwania w głąb udało się określić ścieżkę prowadzącą do zbitia króla. Kolejno odwiedzone przez algorytm pola:

- W, L, I, R, G, N, Q, H

A	B	C	D	E
				
F	G	H	I	J
	5.	 8.	3.	
K	L	M	N	O
	2.		6.	
P	Q	R	S	T
	7.	4.		
U	V	W	X	Y
		 1.		

Rysunek 4: Sekwencja ruchów skoczka

Jak widać algorytm poprawnie znalazł i skazał ścieżkę prowadzącą do znalezienia króla. Problemem jest natomiast ilość ruchów potrzebnych do osiągnięcia tego celu. Jak wynika z powyższego obrazka, algorytm potrzebował aż siedmiu ruchów, by zbić króla.

2.3 Algorytm A*

Algorytm A* (A star), służy do znajdowania najkrótszej ścieżki w grafie ważonym, od punktu startowego do wybranego celu. Z założenia nadaje się on więc znakomicie do znalezienia optymalnego rozwiązania problemu przedstawionego w zadaniu. Dla znalezionego w algorytmie wierzchołka jesteśmy w stanie zdefiniować funkcję oceniającą $f(n)$, do której minimalizacji dążymy:

$$f(n) = g(n) + h(n) \quad (1)$$

, gdzie $f(n)$ jest minimalizowaną przez nas funkcją, $g(n)$ określa drogę między polem "n", a wierzchołkiem startowym, $h(n)$ to przewidywana przez heurystykę droga od wierzchołka n do wierzchołka celu. Do implementacji algorytmu potrzebne jest zdefiniowanie funkcji heurystycznej. W rozważanym przypadku będzie to odległość w metryce Manhattan definiowana jako:

$$h(n) = |u - p| + |v - q| \quad (2)$$

gdzie (u,v) są współrzędnymi rozważanego pola, a (p, q) współrzędnymi wierzchołka celu (w naszym przypadku pola H). W naszym przypadku należało nieznacznie zmodyfikować wzór, by uwzględnić implementację planszy w programie jako tabeli jedno wymiarowej. W programie zamiast oznaczeń literowych (A-Y), użyto

numeracji przy użyciu kolejnych liczb naturalnych (0-24). Zatem zmodyfikowany wzór funkcji heurystycznej $h(n)$ ma postać:

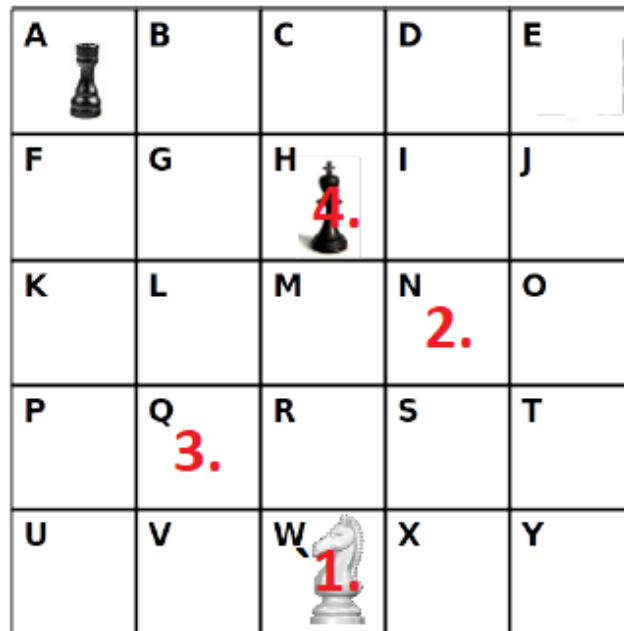
$$h(n) = |a/5 - b/5| + |a\%5 - b\%5|; \quad (3)$$

,gdzie "÷" jest operatorem dzielenia całkowitego, z "%" operatorem "modulo". Dodatkowo przyjęto, że koszt jednego ruchu konika szachowego, zliczany w funkcji $g(n)$ wynosił będzie 1.

2.3.1 Wyniki

Za pomocą algorytmu A* udało się znaleźć najkrótszą ścieżkę prowadzącą do zbitia czarnego króla

- W, N, Q, H



Rysunek 5: Sekwencja ruchów skoczka

Za pomocą A* udało się znaleźć optymalną drogę osiągnięcia zamierzonego celu, czyli zbitia czarnego króla. Z ułożenia pionków na planszy wynika, że minimalna ilość ruchów potrzebnych do zbitia króla wynosi trzy.




2.4 Algorytm Dijkstry

Dodatkowo do rozwiązania problemu bicia czarnego króla, użyto także algorytmu Dijkstry. Za jego pomocą, podobnie jak w przypadku algorytmu A*, jesteśmy w stanie wygenerować najkrótsze rozwiązanie problemu. Przy okazji znajdujemy także najkrótsze ścieżki prowadzące do każdego z pól na planszy (wykluczając pola wyłączone poprzez obecność wieży w polu A).

2.4.1 Wyniki

Najkrótsza ścieżka prowadząca do zbitia króla na polu H, znaleziona za pomocą algorytmu Dijkstry prezentuje się następująco:

- W, L, S, H

A	B	C	D	E
				
F	G	H	I	J
		 4.		
K	L	M	N	O
	2.			
P	Q	R	S	T
			3.	
U	V	W	X	Y
		 1.		

Rysunek 6: Sekwencja ruchów skoczka

Różni się ona zatem od ścieżki znalezionej przez algorytm A*. Zachowana jest jednak minimalna ilość ruchów potrzebnych do realizacji zadania wynosząca 3. Z wyników działania algorytmu dowiedzieć się możemy także, że największa wymagana ilość ruchów potrzebna do zbiccia króla, przy zachowaniu położenia czarnej wieży na polu A, potrzebna byłaby dla króla ustawionego na polu M oraz O i wynosiłaby cztery ruchy.

3 Wnioski

Jak widać każdy z zastosowanych w programie algorytmów:

- algorytm DFS
- algorytm A*
- algorytm Dijkstry

pozwolił na znalezienie ścieżki stanowiącej rozwiązanie problemu "Koszmaru króla". Optymalne rozwiązania zapewnia zarówno algorytm A*, jak i algorytm Dijkstry. Z racji jednak potrzeby znalezienia najkrótszego połączenia wyłącznie pomiędzy dwoma wierzchołkami grafu, preferowanym algorytmem służącym do tego celu jest na pewno rozwiązanie A*, którego głównym zastosowaniem jest rozwiązywanie problemu najkrótszej ścieżki pomiędzy dwoma elementami w grafie.

Literatura

- [1] https://en.wikipedia.org/wiki/Algorytm_Dijkstry
- [2] https://pl.wikipedia.org/wiki/Algorytm_A*
- [3] https://eduinf.waw.pl/inf/alg/001_search/0125.php