

Wstęp do sztucznej inteligencji

Wojciech Typer

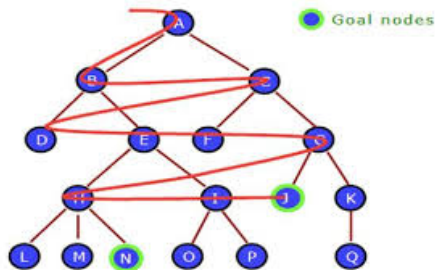
Przestrzeń stanów

Przestrzeń stanów to uporządkowana czwórka (V, E, S, F) , gdzie:

- $V \rightarrow$ zbiór wierzchołków reprezentujących stany powstałe w trakcie rozwiązywania problemów
- $E \rightarrow$ zbiór krawędzi reprezentujących możliwe przejścia między stanami
- $S \rightarrow$ niepusty podzbiór V , zawierający stany początkowe problemu
- $F \rightarrow$ niepusty podzbiór V , zawierający stany docelowe problemu (mogą być zdefiniowane wprost lub przez własności które chcemy osiągnąć)
- Rozwiązaniem będziemy nazywać ścieżkę w grafie od stanu początkowego do stanu docelowego

Podstawowe problemy przy przeszukiwaniu przestrzeni stanów

- Czy algorytm gwarantuje znalezienie rozwiązania?
- Czy algorytm zawsze się kończy?
- Czy algorytm znajduje optymalne rozwiązanie?
- Jaka jest złożoność czasowa i pamięciowa algorytmu?
- Jak można poprawić złożoność czasową i pamięciową?

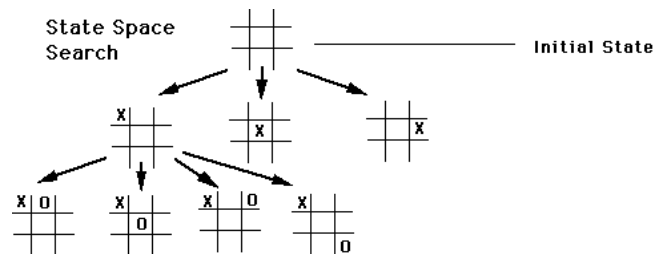


Sformułowanie zadania dla algorytmów przeszukiwania

- Precyzyjna definicja przestrzeni stanów
- określenie stanu początkowego
- określenie reguł przejścia między stanami (operatory akcji lub funkcja następnika)
- zbiór stanów docelowych lub funkcja weryfikacji osiągnięcia celu
- funkcja kosztu ścieżki

Kierunki przeszukiwania

- w przód (od stanu początkowego do celu)
- w tył (od stanu celu do początku)
- przeszukiwanie dwukierunkowe



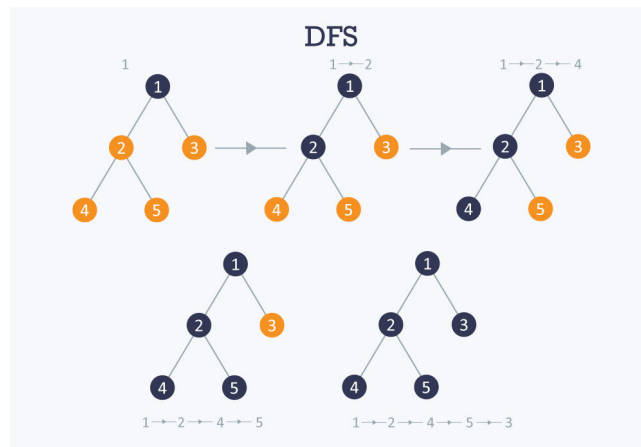
Strategie przeszukiwania

- Przeszukiwanie w głąb
 - zaczynamy w wierzchołu początkowym
 - dla aktualnego wierzchołka v :
 - * oznacz v jako zbadany
 - * jeśli v jest celem zakończ procedurę
 - * jeśli niezbadany jest wierzchołek do którego można przejść to przejdź do niego

- * jeśli nie ma już niezbadanych sąsiadów to wróć do wierzchołka z którego przyszedłeś

Zalety: łatwość implementacji, małe wymagania pamięciowe

Wady: znalezione rozwiązanie nie musi być optymalne



- Przeszukiwanie w głąb ze stosem
 - zaczynamy w wierzchołku początkowym
 - dla aktualnego wierzchołka v :
 - * oznacz v jako zbadany
 - * jeśli v jest celem zakończ procedurę
 - * dla każdego sąsiada v , który jest niezbadany, dodaj go na stos i oznacz jako odwiedzony
 - * jako następny, weź wierzchołek ze szczytu stosu

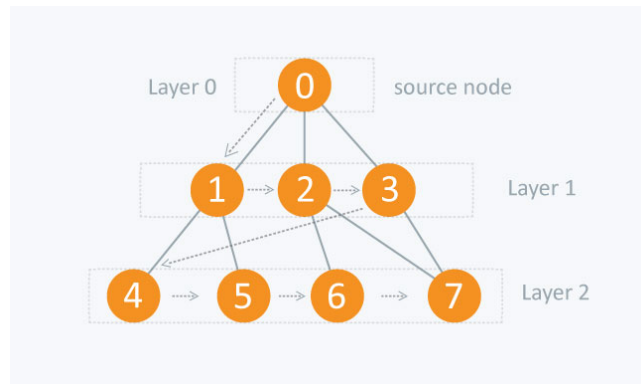
Zalety: łatwość implementacji, łatwe struktury pamięciowe

Wady: znalezione rozwiązanie nie musi być optymalne

- Przeszukiwanie wszerz
 - zaczynamy w wierzchołku początkowym
 - dla aktualnego wierzchołka v :
 - * oznacz v jako zbadany
 - * jeśli v jest celem zakończ procedurę
 - * dla każdego sąsiada v , który jest niezbadany, dodaj go do kolejki i oznacz jako odwiedzony
 - * jako następny, weź wierzchołek z początku kolejki

Zalety: znalezione rozwiązanie jest optymalne

Wady: duże wymagania pamięciowe



- Best-First-Search

- Rozszerzenie przeszukiwania wszerek po dodaniu funkcji kosztu, która eksploruje graf poprzez rozwinięcie najbardziej obiecującego węzła wybranego zgodnie z określoną regułą.

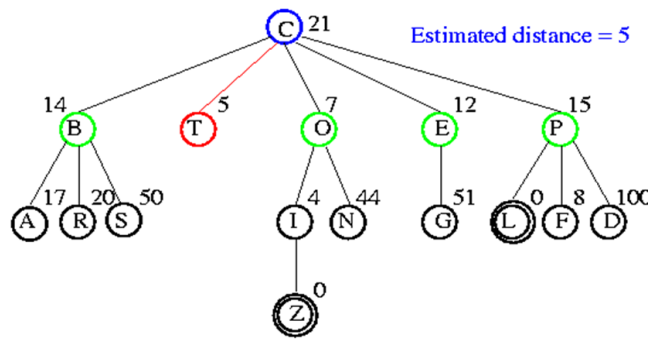
- * zaczynamy w wierzchołku początkowym, którego koszt ustwiamy na 0

- * dla aktualnego wierzchołka v :

- oznacz v jako zbadany
 - jeśli v jest celem zakończ procedurę
 - dla każdego sąsiada v który jest odwiedzony, jeśli koszt v plus koszt przejścia są mniejsze niż jego dotychczasowy koszt to zmodyfikuj go w kolejce priorytetowej nadając nowy mniejszy koszt
 - dla każdego sąsiada v , który jest niezbadany, dodaj go do kolejki priorytetowej z kosztem v plus koszt przejścia oraz oznacz jako odwiedzony
 - jako następny, weź wierzchołek z kolejki priorytetowej o najmniejszym koszcie

Zalety: znalezienie rozwiązania jest optymalne, ze względu na koszt ścieżki

Wady: kolejka priorytetowa jest trudniejsza w implementacji i ma większą złożoność czasową



- A* Search