

RAPORT

1. Użyte struktury danych

- **Reprezentacja typu vector:**

Stan opisany jest listą długości n, gdzie indeks odpowiada wierszowi, a wartość – kolumnie hetmana.

Przykład: [1, 3, 0, 2] oznacza hetmany na polach (0,1), (1,3), (2,0), (3,2).

Sprawdzenie konfliktów jest szybkie dzięki prostemu porównywaniu kolumn i przekątnych.

- **Reprezentacja typu tuple:**

Stan to lista krotek (wiersz, kolumna).

Przykład: [(0,1), (1,3), (2,0), (3,2)].

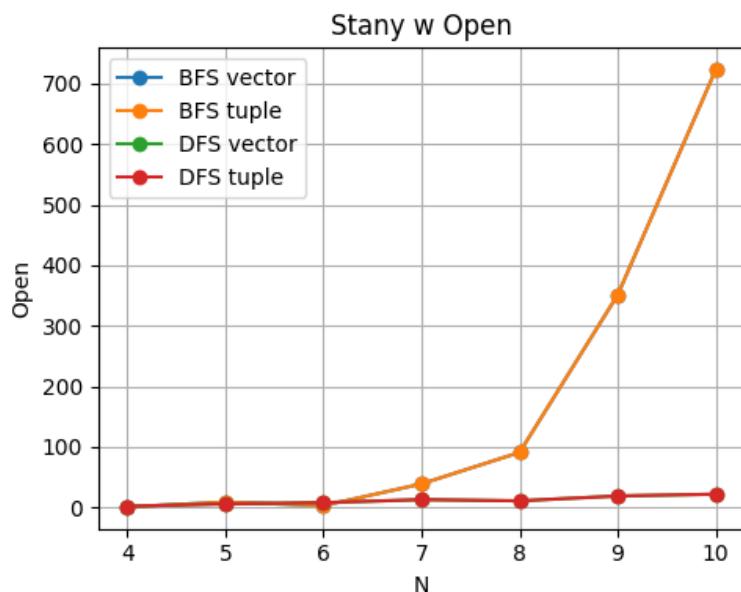
Umożliwia bardziej ogólne rozszerzenie, lecz wymaga podwójnej pętli przy sprawdzaniu konfliktów, co zwiększa koszt obliczeń.

Obie struktury były wykorzystywane przez algorytmy BFS i DFS, tworząc cztery kombinacje:

- BFS + vector
- BFS + tuple
- DFS + vector
- DFS + tuple

2. Wykresy i analiza wyników

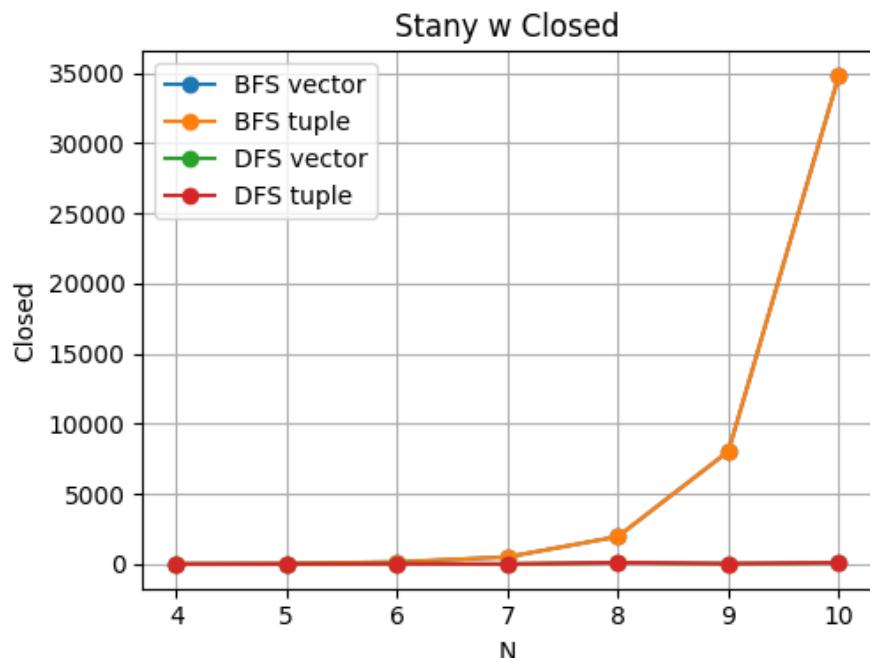
Poniższe wykresy przedstawiają liczbę产生的 stanów oraz czas wykonania w zależności od rozmiaru problemu ($N = 4–10$).



Liczba stanów w OPEN

Podobna tendencja jak wyżej BFS utrzymuje w pamięci duże kolejki, co zwiększa zapotrzebowanie na pamięć.

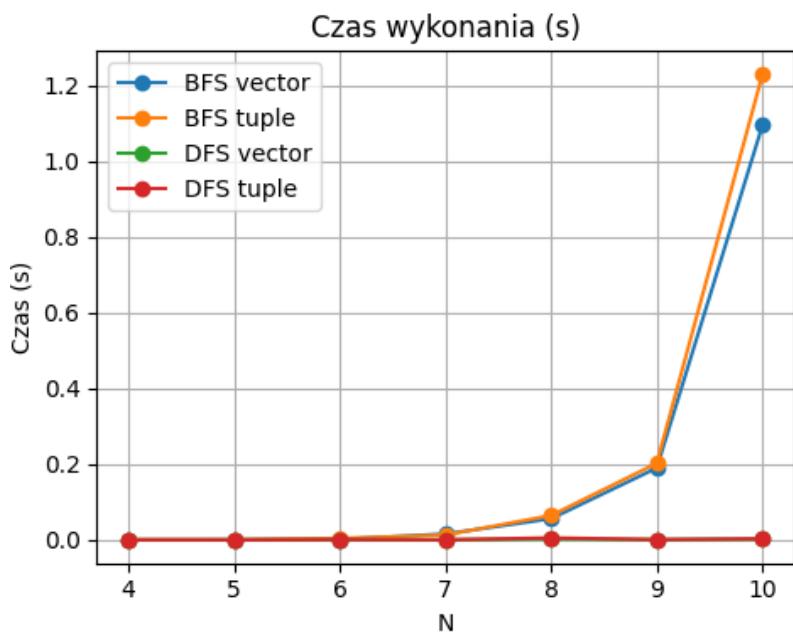
DFS korzysta ze stosu, dzięki czemu pamięć jest używana bardziej oszczędnie.



Liczba stanów w zbiorze CLOSED

Algorytm BFS eksploruje znacznie większą liczbę stanów rośnie ona wykładniczo z N.
Dla N=10 liczba zamkniętych stanów przekracza 34 000.

DFS działa znacznie szybciej, przetwarzając jedynie dziesiątki stanów.



Czas wykonania

BFS jest wolniejszy, a czas rośnie gwałtownie wraz z N (np. 1,1 s dla vector, 1,28 s dla tuple przy N=10).

DFS znajduje rozwiązania znacznie szybciej (rzędu milisekund).

Reprezentacja tuple jest konsekwentnie wolniejsza od vector, średnio o 10–30 %

3. Interpretacja wyników

1. Wydajność algorytmów:

- DFS jest wyraźnie szybszy i mniej pamięciożerny w tym problemie, ponieważ zwykle trafia na rozwiązanie bez pełnego przeszukania całej przestrzeni.
- BFS gwarantuje znalezienie rozwiązania najkrótszego (o minimalnej głębokości), lecz jego koszt obliczeniowy szybko rośnie wraz z N.

2. Wpływ reprezentacji danych:

- Reprezentacja vector jest bardziej efektywna – prostsze porównania, mniej złożona struktura, lepsze wykorzystanie pamięci podręcznej.
- Reprezentacja tuple jest bardziej ogólna i czytelna, ale wolniejsza z uwagi na dodatkowe operacje na listach krotek.

3. Poprawność

- Dla każdego N=4–10 algorytmy zwracały poprawne rozwiązania (brak konfliktów).
- Zwiększanie N wpływało jedynie na liczbę产生的 stanów i czas, nie na poprawność wyników.

4. Wnioski

- Najbardziej efektywne podejście: DFS z reprezentacją vector.
- BFS jest przydatny do analizy pełnej przestrzeni stanów, lecz niepraktyczny dla większych N z powodu rosnącej złożoności czasowej i pamięciowej.
- Reprezentacja danych ma zauważalny wpływ na wydajność, jednak dominujący czynnik stanowi sam charakter algorytmu przeszukiwania.