Algorytmy i struktury danych

Projekt 1

Projektodawca: Kamil Reczek

Patryk Tomaszewski
2020-04-02

Spis treści

1. Wprowadzenie do projektu

Projekt zakłada przetestowanie dwóch sposobów przeszukiwania zbiorów liczbowych i wskazanie, który z nich jest wydajniejszy, w jakich przypadkach oraz dlaczego. Metody przeszukiwania zbiorów to:

- Wyszukiwanie proste liniowe – polega na wybieraniu ze zbioru kolejnych liczb oraz przyrównywaniu ich do szukanej. (Algorytm nie wymaga posortowania elementów)
- Wyszukiwanie binarne – polega na wyznaczeniu środka z szeregu szukanych elementów i sprawdzenie czy środkowy element jest szukany. Jeżeli nie to algorytm tworzy nowy podzbiór na prawo lub na lewo od ostatniego porównywanego elementu (w zależności czy szukana była większa lub mniejsza od porównywanej liczby). Następnie algorytm powtarza czynność do znalezienia szukanego elementu. (Wyszukiwanie binarne wymaga posortowania zbioru elementów)

a. Narzędzia użyte w projekcie

Do przeprowadzenia badania dwóch algorytmów wyszukiwania zostały użyte następujące narzędzia:

- Program Microsoft Visual Studio 16.5.2
- Konsola CMD
- Pakiet Biurowy Microsoft Office (wykresy)

b. Środowisko testowe

Testy zostały przeprowadzone na systemie operacyjnym Windows 10 Pro w wersji 1909. Za to za platformę testową posłużył komputer stacjonarny o następującej specyfikacji:

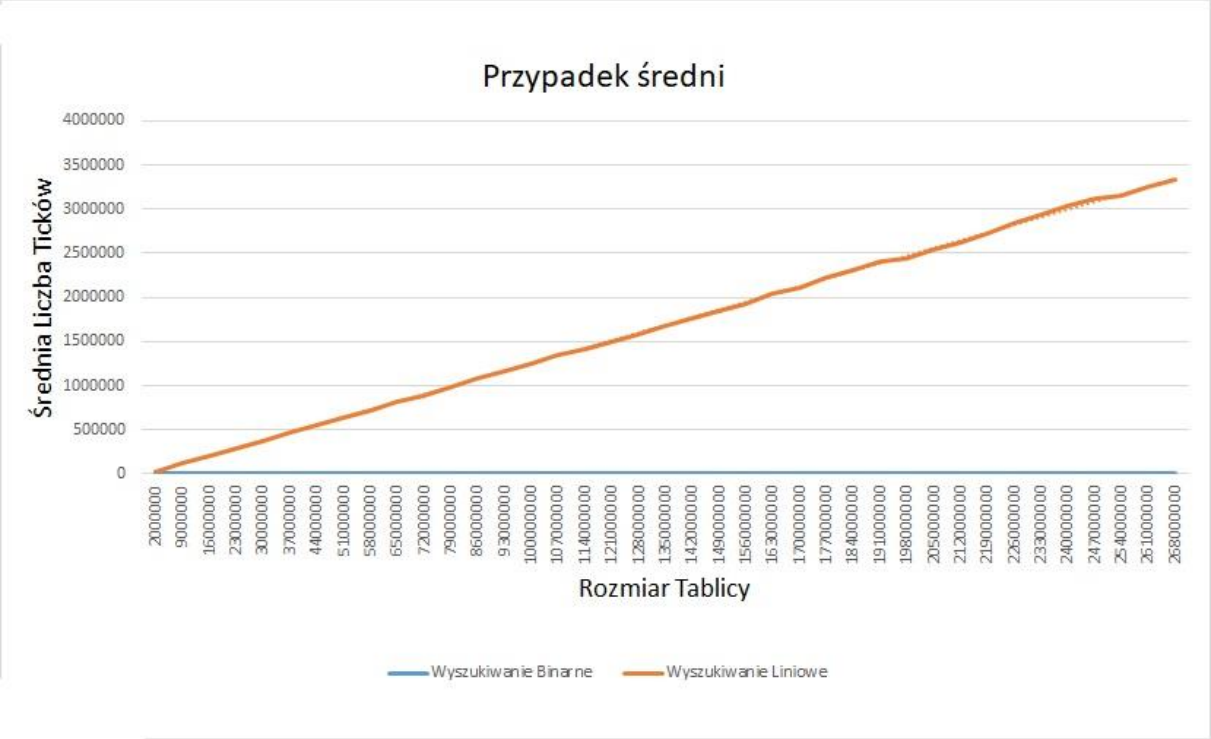
- CPU – AMD Ryzen 3700X
- RAM – 2x8 GB 3200MHz
- MB – MSI B350
- Dysk – M.2 PCIe 3.0 256GB

c. Kod źródłowy

Kod źródłowy można znaleźć na repozytorium GitHub pod tym: https://github.com/piciu0908/WyszukiwanieP1

2. Dane oraz wyniki

a) Przypadek średni – pomiar czasu



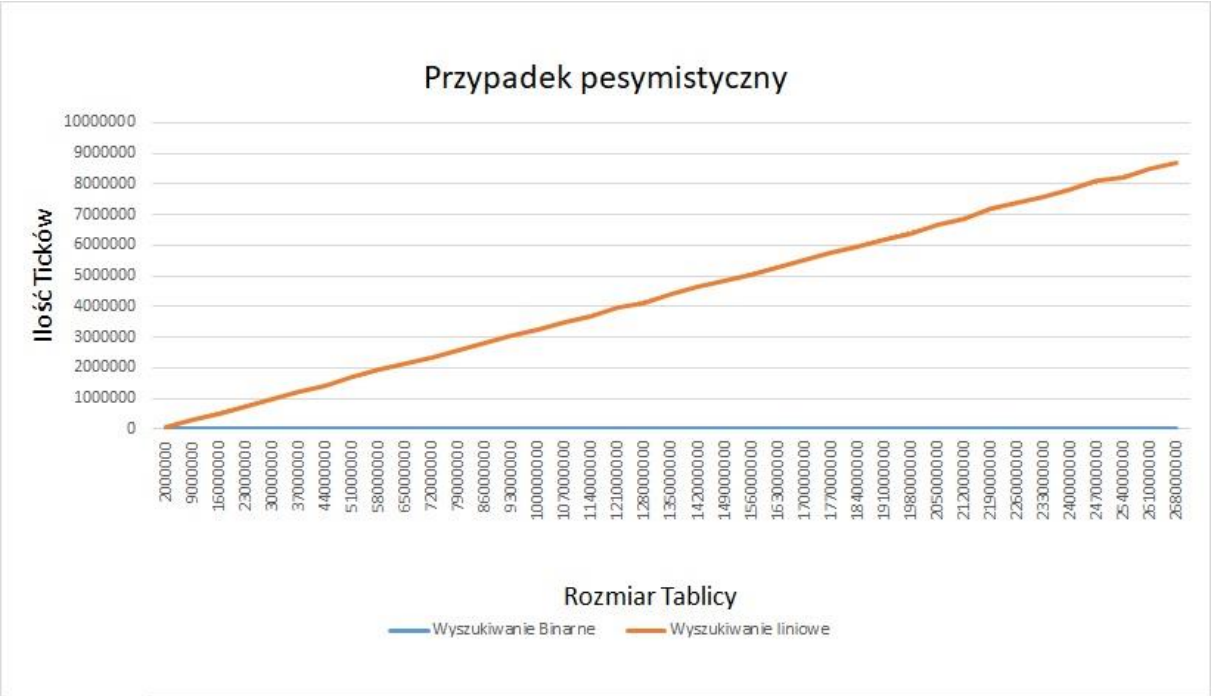
Wyniki zostały uzyskane w następujący sposób:

a) Dla wyszukiwania binarnego było to znalezienie 1900000 elementów oraz zmierzenie kosztu ich znalezienia. Następnie suma kosztu została podzielona przez liczbę elementów by uzyskać średnią dla danej tablicy.
b) W wyszukiwaniu liniowym kluczowym elementem jest znalezienie środkowego elementu dla danego rozmiaru tablicy i obliczenie kosztu jego odnalezienia.

Wniosek:

- W przypadku pomiaru czasu obu algorytmów zdecydowaną przewagę ma wyszukiwanie binarne, gdyż dla bardzo dużej liczby elementów jest o wiele szybsze.

b) Przypadek pesymistyczny – pomiar czasu



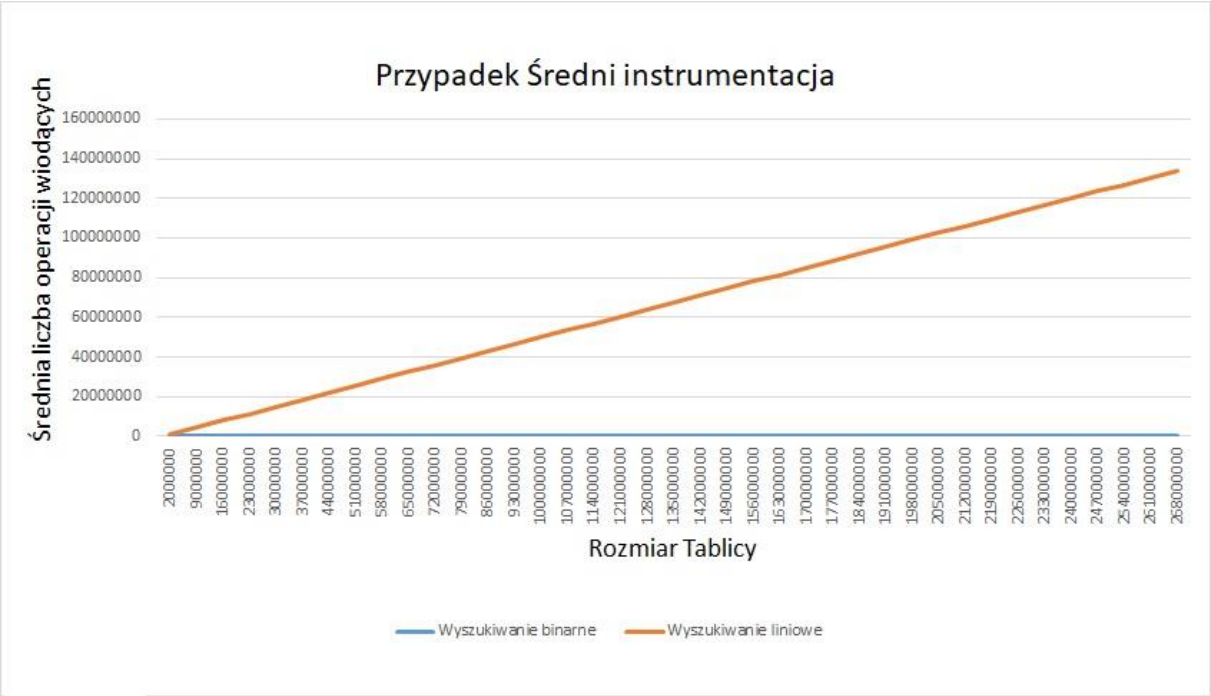
Wyniki zostały uzyskane w następujący sposób:

a) Dla obu algorytmów było to szukanie elementu, który nie znajduje się w zbiorze szukanym. Kluczem było zliczenie czasu wszystkich operacji przejścia całej tablicy.

Wniosek:

- W przypadku gdy algorytmy musza przeszukać całe tablice po raz kolejny im większa jest tablica tym bardziej wyszukiwanie binarne pogłębia swoją przewagę nad wyszukiwaniem liniowym.

c) Przypadek średni instrumentacja



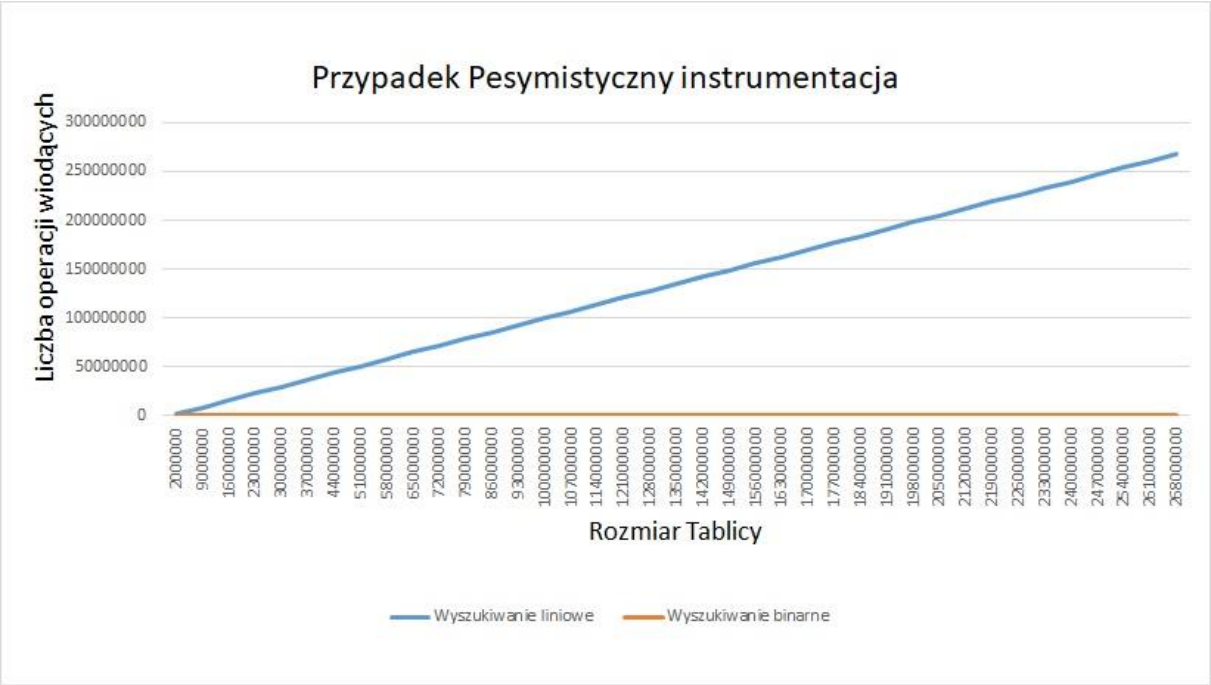
Wyniki zostały uzyskane w następujący sposób:

a) Dla wyszukiwania binarnego było to znalezienie 1900000 elementów oraz zliczenie ilości operacji krytycznej/wiodącej. Następnie suma kosztu została podzielona przez liczbę elementów by uzyskać średnią dla danej tablicy.
b) W wyszukiwaniu liniowym został zdefiniowany środek oraz zliczona liczba operacji potrzebnych do jego znalezienia.

Wniosek:

- Nawet w przypadku średniej liczby operacji algorytm wyszukiwania liniowego przegrywa z binarnym. Wyszukiwanie binarne ma wielką przewagę ze względu na rozmiar przeszukiwanych tablic.

d) Przypadek pesymistyczny instrumentacja



Wyniki zostały uzyskane w następujący sposób:

a) Dla wyszukiwania binarnego zliczenie wszystkich operacji krytycznych/wiodących dla przeszukania całej tablicy.
b) W wyszukiwaniu liniowym został zdefiniowany element spoza tablicy i zostały zliczone operacje krytyczne/wiodące dla przejścia całej tablicy.

Wniosek:

- Liczba wykonanych operacji w wyszukiwaniu binarnym jest dużo mniejsza by ukończyć działanie algorytmu.

3. Podsumowanie

Podsumowując. Algorytm wyszukiwania binarnego jest zdecydowanie szybszy oraz bardziej optymalny dla dużych liczb. Jeśli chcemy by nasz program działał szybciej, a operujemy na dużych wartościach to warto go stosować. Sprawa może wyglądać nieco inaczej dla małych liczb gdzie wyszukiwanie liniowe może być bardziej optymalnym wyjściem.