Sprawozdanie
Projektowanie aglorytmów i metod sztucznej inteligencji
Laboratorium 7

Michał Wieczorek
226284

24 maja 2017

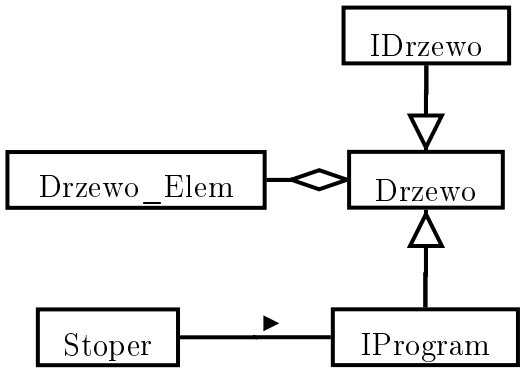
1 Cel i przebieg laboratorium

Omówienie działania drzewa binarnego. Sposób rozmieszczenia elementów w drzewie. Złożoność obliczeniowa struktury drzewa. Najlepszy i najgorszy przypadek rozłożenia elementów w strukturze.
Implementacja interfejsu drzewa binarnego. Implementacja prototypów funkcji struktury.

2 Struktura i działanie algorytmu

2.1 Struktura kodu

Dodano nową klasę *Drzewo*, dziedziczy ona bezpośrednio z interfejsu *IDrzewo* oraz *IProgram*. Dodatkowo dodano nową klasę *Drzewo-Elem*, z której złożone są węzły drzewa. Zdecydowałem się na implementację drzewa czerwono-czarnego, dlatego klasa ta posiada dodatkowe pole kolor. Dodatkowo w klasie *Drzewo* dodano pole *wartownik*, na którego wskazuje korzeń i liście drzewa.



Rysunek 1: Diagram klas

2.2 Drzewo czerwono-czarne

Interfejs drzewa składa się z następujących metod:

- Dodawanie nowego węzła o zadanej wartości liczbowej

- Usunięcie węzła o podanej wartości
- Wyszukanie elementu o podanej wartości i zwrócenie wskaźnika na ten element
- Wyświetlenie zawartości drzewa od lewej do prawej (razem z kolorami węzłów) *in-order*
- Wyświeltenie zawartości drzewa wyświetlając najpierw korzen, a później resztę elementów *pre-order*

Drzewo czerwono-czarne posiada następujące cechy:

- Korzeń drzewa jak i jego liście są zawsze czarne oraz wskazują na wartownika
- Każdy węzeł drzewa jest albo czerwony, albo czarny
- Jeśli węzeł jest czerwony to jego synowie są czarni
- Węzeł czerwony nie może łączyć się z innym czerwonym węzłem
- Każda ścieżka z wybranego węzła do liścia ma tyle samo czarnych węzłów

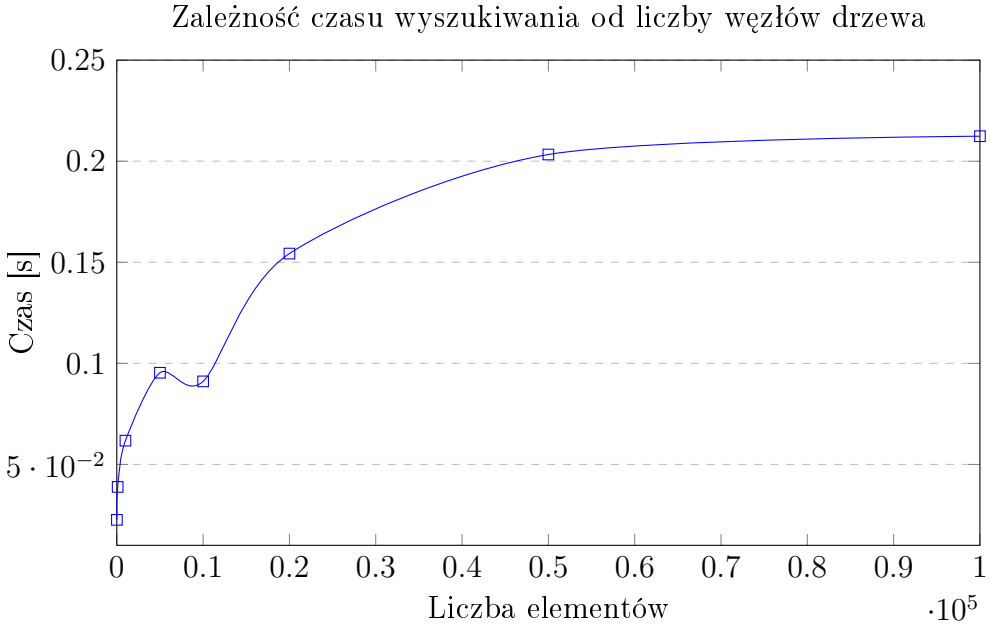
Dodawanie nowych węzłów do drzewa powoduje zmianę kolorów węzłów oraz rotacje drzewa. Dodany węzeł ma zawsze kolor czerwony, dlatego jeśli jego rodzic jest również czerwony należy dokonać operacji *naprawy drzewa*. Uruchomiona zostaje funkcja która koryguje kolorystykę i równoważy drzewo.

3 Testy programu

Mierzony jest czas wykonania 1000000 wyszukiwań, aby pomiary były wiarygodne. W tym miejscu należy zaznaczyć, że elementy do drzewa były dodawane w kolejności rosnącej - $1, 2, 3, ..., n$. Skutkiem czego jest wydłużenie się prawej częsci drzewa względem lewej. Rożnica ta, zależy od tego w którym momencie wykonano rotacje i wyrównano drzewo.

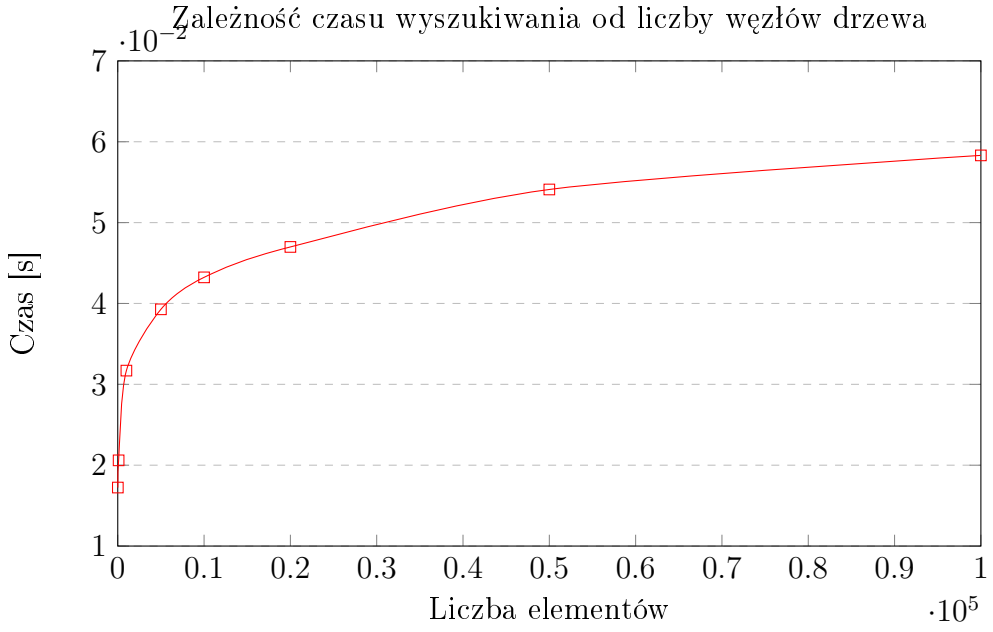
3.1 Najgorszy przypadek

Wykres przedstawia najgorszy przypadek, czyli wyszukiwanie ostatniego elementu najdłużej gałęzi. Możemy zauważyć, że są pewne nierówności. Wynika to z faktu, że w niektórych momentach drzewo było bardziej zrównoważone, i prawa część była tylko nieznacznie dłuższa niż lewa. Charakterystyka ma kształt funkcji logarytmicznej.



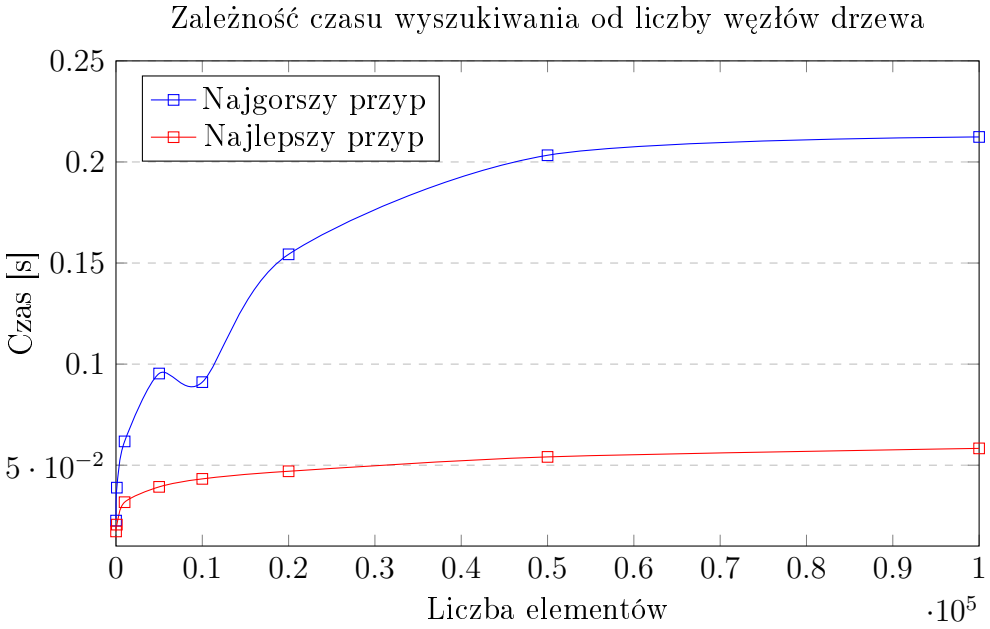
3.2 Najlepszy przypadek

W tym przypadku wyszukiwano element w lewej gałęzi drzewa. Jak widać charakterystyka jest bardziej regularna, nie ma tu żadnych załamań.

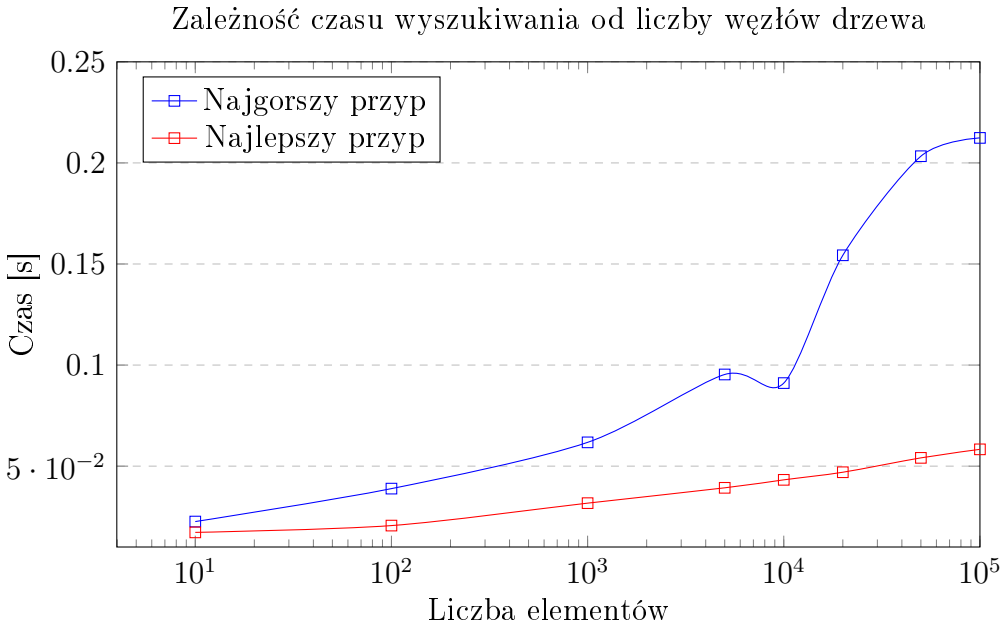


3.3 Porównanie

Czasy wyszukiwania w najdłuższej gałęzi są znacznie wyższe od tych w krótszej.



Wykresy w skali logarytmicznej



4 Wnioski

Podsumowując. złożoność obliczeniowa algorytmu wyszukiwania w drzewie czerwono-czarnym wynosi $O(log(n))$ gdzie n - liczba węzłów. Porównując z listą, której złożoność wyszukiwania wynosi $O(n)$, dostajemy całkiem szybki dostęp do wprowadzonych danch, szczególnie dla większych drzew. Dodatkowo oznaczanie węzłów kolorami daje nam większą kontrolę nad algorytmem, łatwiej zauważyć co dzieje się w środku drzewa.