Politechnika Wrocławska
Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Binarne drzewa poszukiwań

Autor: Rafał Borysionek 226262
Prowadzący: Mgr inż. Andrzej Wytyczak Partyka

24 maja 2017

1 Cel ćwiczenia

Celem ćwiczenia było zbadanie złożoności obliczeniowej wyszukiwania w równoważonym drzewie poszukiwań binarnych. Ponadto należało porównać uzyskaną złożoność względem listy jedkokierunkowej oraz na tej podstawie wyciągnąć wnioski.

2 Sposób wykonania

Do wykonania badań zdecydowano się na implementację drzewa AVL. Tą strukrurę danych oparto na interfejsie *itree*. Aby zbadać czas wykonywania wyszukiwania należało dopasować implementowane drzewo do interfejsu *irunnable*.

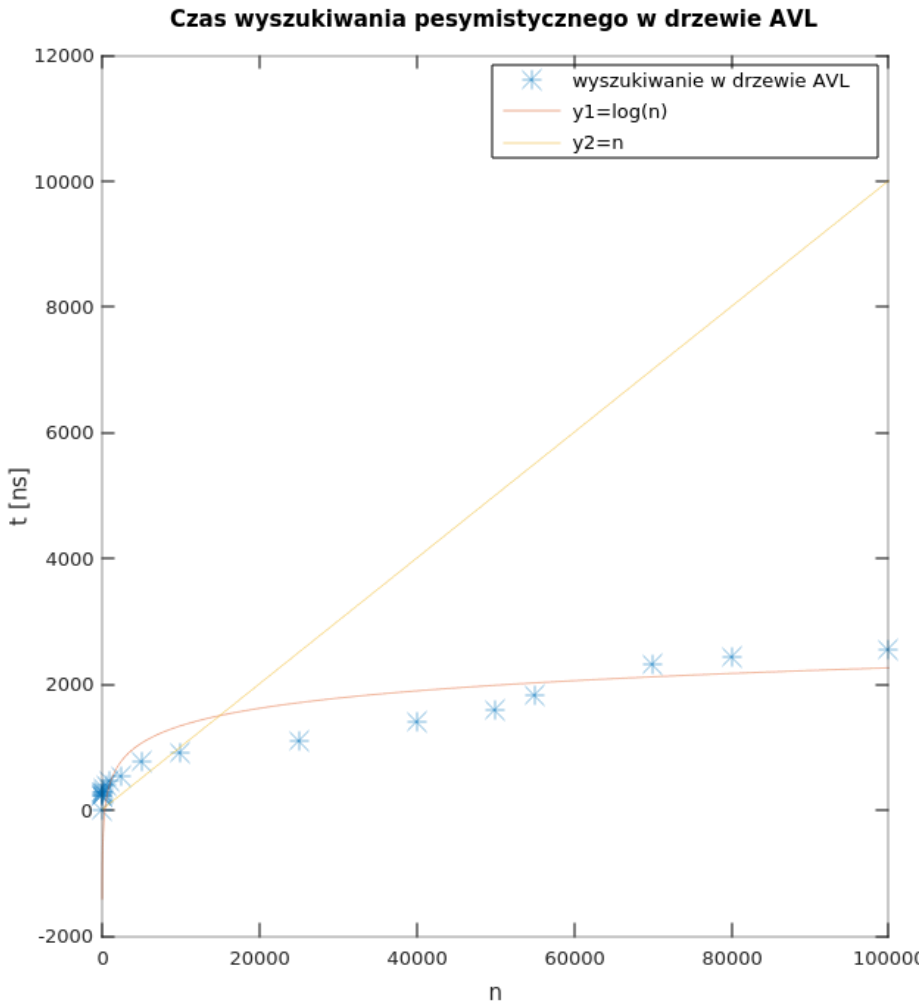
Jako że czas wyszukiwania w drzewie mającym nawet sto tysięcy elementów był bardzo niewielki, to zdecydowano się na większą precyzję zegara. Wobec tego badany czas jest podany w nanosekundach.

Badano pesymistyczny przypadek wyszukiwania. Szukany obiekt znajdował się zawsze na samym dole hierarchii drzewa.

3 Wyniki pomiarów

Liczba elementów	Czas [ns]
10	243
20	244
25	244
50	295
75	295
100	296
150	295
200	344
500	411
1000	461
2500	551
5000	776
10000	917
25000	1109
40000	1397
50000	1597
55000	1841
70000	2331
80000	2442
100000	2556

4 Wykres



5 Wnioski

Porównując charakterystyki przedstawionych funkcji na wykresie można dojść do wnioski, że pesymistyczne wyszukiwanie elementu z drzewa AVL ma złożoność obliczeniową O(log(n)). Jest to lepszy wynik niż w przypadku listy, gdzie złożoność obliczeniowa była równa O(n). Spowodowane jest to tym, że lista to tak na prawdę najgorszy przypadek nierównomiernie rozłożonego drzewa (drzewo, w którym każdy kolejny syn ma tylko jednego potomka).

W przypadku równomiernego drzewa wyszukiwań binarnych (jakim jest wyżej zaimplementowane drzewo AVL) przy każdym porównaniu kluczy odrzu-

cana jest około połowa aktualnie branych pod uwagę elementów drzewa, podczas gdy w liście nie są odrzucane żadne elementy i sprawdzany jest każdy z nich (przy pesymistycznym przypadku).

Drzewo AVL ma jednak także swoje wady. W przypadku, gdy użytkownik chce przechowywać mało danych, które często będą modyfikowane - lepiej sprawdzi się w tym lista. Spowodowane jest to faktem, że przy częstej modyfikacji drzewa AVL, potrzebne jest jego balansowanie, a przez to wzrasta złożoność obliczeniowa. Badane drzewo sprawdzi się dobrze w sytuacjach, gdy przechowywanych jest dużo elementów, oraz nie będą one zbyt często modyfikowane, a dostęp do nich musi być stosunkowo szybki. Oprócz tego, im więcej elementów jest umieszczone w drzewie AVL, tym większe oszczędności w czasie wyszukiwania (odrzucane są kolejne połowy drzewa, które nie są już brane pod uwagę do porównania wartości).

Literatura

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Wprowadzenie Do Algorytmów wydanie VII.* PWN 2015.

[2] Bjarne Stroustrup. *C++ Kompendium Wiedzy wydanie IV.* Helion 2014.

[3] Robert C. Martin. *Zwinne Wytwarzanie Oprogramowania: najlepsze zasady, wzorce i praktyki.* Helion 2015.

[4] Wikipedia
`https://en.wikipedia.org/wiki/AVL_tree`