# PAMSI – testowanie algorytmów wyszukiwania

Piotr Wilkosz

13/04/2014

## 1 Wstęp

Celem ćwiczenia było przetestowanie implementacji takich struktur danych jak:

- Tablica asocjacyjna - tworzenie z góry posortowanej tablicy i wyszukiwanie elementu
- Binarne drzewo przeszukiwań
- Tablica haszująca z adresowaniem liniowym

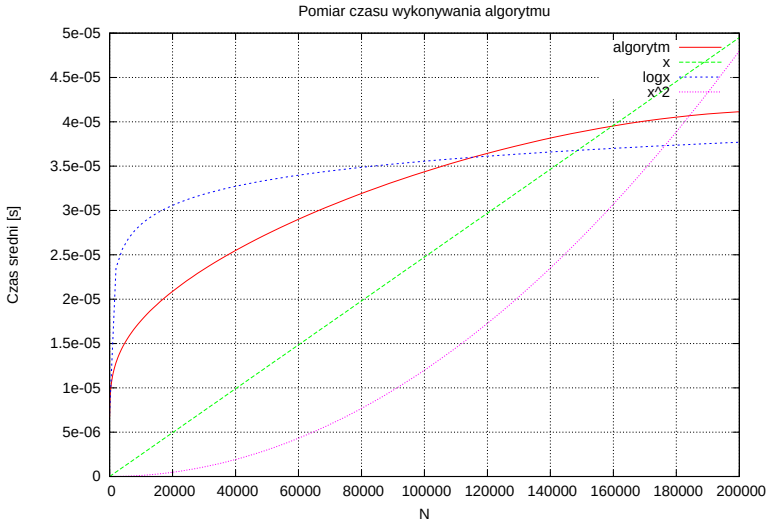
Zadaniem było zmierzenie czasu wykonywania operacji wyszukania losowego elementu wśród powyższych struktur.

## 2 Wyniki pomiarów

1. TAblica asocjacyjna

Tablica 1: Test nr 1

| N | czas | odchylenie |
|---|---|---|
| 10 | 6.7048e-06 | 3.49235e-06 |
| 100 | 1.20477e-05 | 2.6002e-06 |
| 1000 | 1.74256e-05 | 2.58526e-06 |
| 10000 | 2.34596e-05 | 3.69745e-06 |
| 100000 | 3.92021e-05 | 4.56431e-06 |
| 200000 | 4.11365e-05 | 6.67398e-06 |

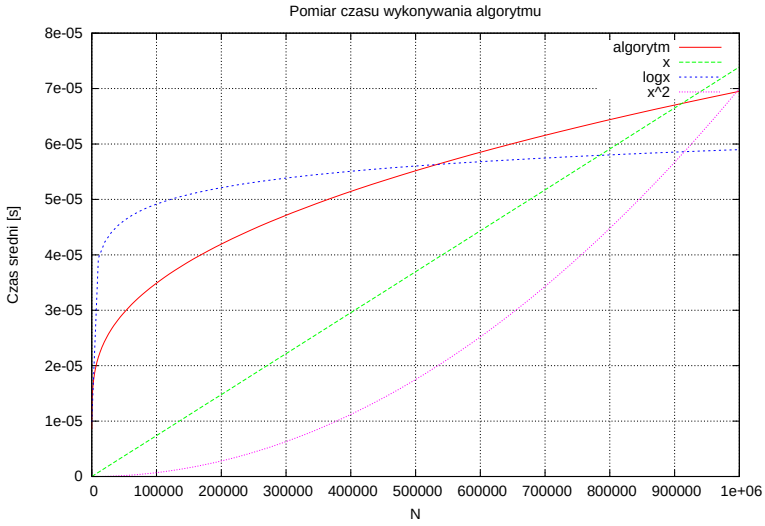



Rysunek 1: Test nr 1

2. Binarne drzewo przeszukiwań

Tablica 2: Test nr 2

| N | czas | odchylenie |
|---|---|---|
| 10 | 8.5488e-06 | 3.08769e-06 |
| 100 | 1.63919e-05 | 3.06447e-06 |
| 1000 | 2.26005e-05 | 3.77314e-06 |
| 10000 | 3.23784e-05 | 6.64965e-06 |
| 100000 | 4.77856e-05 | 8.44921e-06 |
| 1000000 | 6.9513e-05 | 8.87252e-06 |

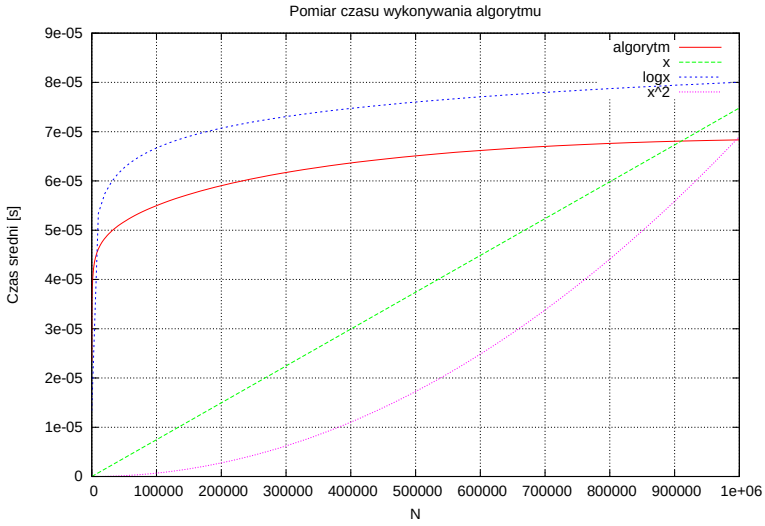



Rysunek 2: Test nr 2

3. Tablica haszująca

Tablica 3: Test nr 3

| N | czas | odchylenie |
|---|---|---|
| 10 | 2.33365e-05 | 3.41544e-06 |
| 100 | 5.06862e-05 | 7.05086e-06 |
| 1000 | 4.44422e-05 | 5.12816e-06 |
| 10000 | 5.34565e-05 | 8.91332e-06 |
| 100000 | 6.63957e-05 | 1.82487e-05 |
| 1000000 | 6.83327e-05 | 1.39888e-05 |




Rysunek 3: Test nr 3

# 3 Wnioski

- Wszystkie przetestowane algorytmy cechują się dużą szybkością. Dla danych rzędu miliona czas wyszukiwania w każdej z powyższych struktur jest zadowalający, tzn. praktycznie niezauważalny dla pojedynczego wyszukania

- Tablica asocjacyjna, bazująca na dwóch tablicach, została zaimplementowana w sposób umożliwiający przeszukiwanie binarne, czyli podczas wstawiania danych do struktury są one jednocześnie sortowane. Jednakże tworzenie owej struktury jest czasochłonne, co jest sporą przeszkodą podczas prowadzenia testów algorytmu, aczkolwiek jest dość wydajną strukturą, gdy dane nie są nazbyt często uzupełniane (problem długiego czasu tworzenia struktury wynika z operacji porównania dwóch obiektów typu string, co znacząco ów czas wydłuża oraz z konieczności odpowiedniego gospodarowania pamięci). Złożoność obliczeniowa wyszukiwania: $O(log_2 n)$

- Binarne drzewo przeszukiwań posiada podobne cechy, jak w przypadku poprzedniej struktury. Niewątpliwą zaletą jest prostsza implementacja, jednak w przeciwieństwie do powyższego algorytmu istnieje ryzyko, iż złożoność obliczeniowa wyszukiwnia wyniesie $O(n)$ (przypadek pesymistyczny). Oczekiwana złożoność wyszukiwania: $O(log_2 n)$

- Tablica haszująca posiada oczekiwaną złożoność obliczeniową $O(1)$. Podczas testów dla danych rzędu miliona, czas wyszukiwania stabili-

zował się dążąc do osiągnięcia wartości stałej, co jest oczywiste biorąc pod uwagę fakt, iż prawdopodobieństwo kolizji zmniejsza się wraz ze wzrostem rozmiaru tablicy. Pesymistyczny przypadek złożoności obliczeniowej wyszukiwania wynosi $O(n)$ jednak przy zastosowaniu odpowiedniej funkcji haszującej oraz dla dużych danych, algorytm taki jest o wiele szybszy od dwóch powyższych, co jest tym bardziej widoczne, im większy jest rozmiar problemu. Czas tworzenia tablicy haszującej jest również o wiele krótszy w porównaniu z powyższymi algorytmami.