Złożone struktury danych

Sprawozdanie

Przygotowali:
Andrzej Wieczorek
Adrian Sworowski

Spis treści

1. Założenia i cele zadania

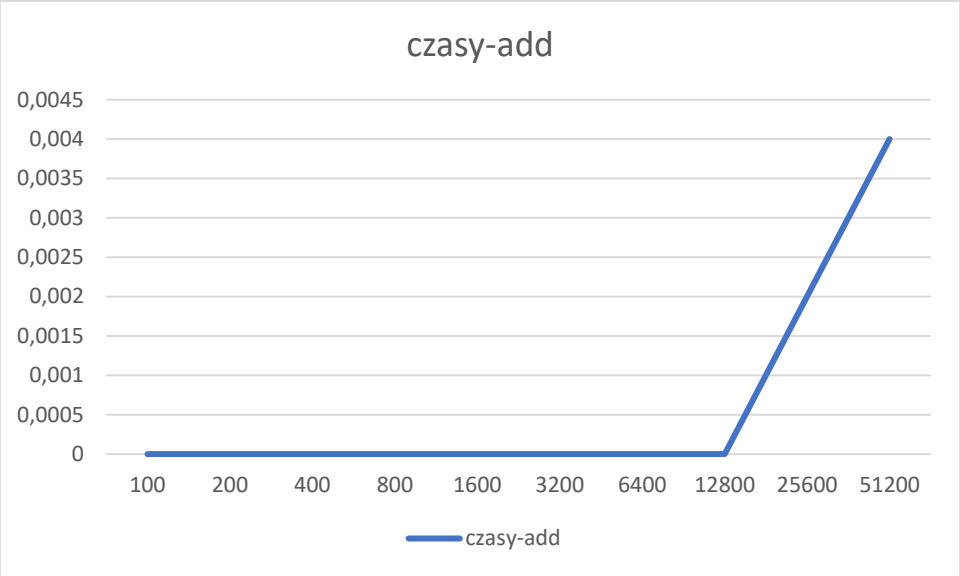
Celem zadania było przygotowanie Celem zadania było przygotowanie programu implementującego dwie struktury danych – listę kierunkową oraz drzewo BST. W kodzie dostarczonym za pośrednictwem Github Classroom należało utworzyć funkcje dodające elementy do struktury w kolejności rosnącej, losowej lub w przypadku BST – zgodnie z regułą połowienia binarnego. Struktury miały zostać później przeszukane, a następnie elementy miały zostać wyczyszczone. Oba warianty miały mieć zmierzone czasy działania wstawienia oraz wyszukania elementów, przedstawione za pomocą tabeli czasów i wykresu.

2. Metodyka testowa

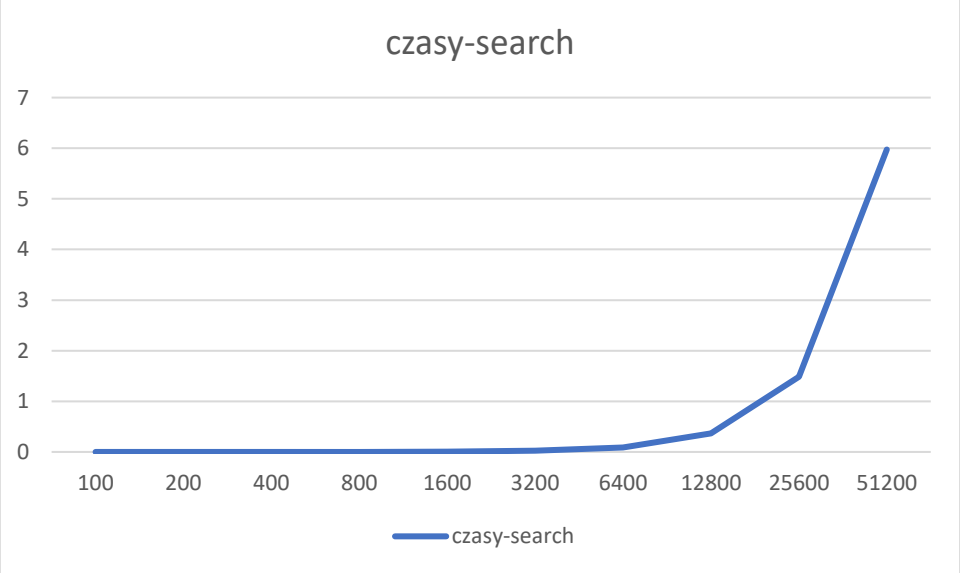
Czasy zostały zmierzone za pośrednictwem dostarczonego kodu. Wartości zostały wpisane do pliku .txt za pośrednictwem odpowiedniej modyfikacji w/w, a następnie wczytane do programu Excel celem dalszej obróbki oraz przygotowania wykresów.

Kod został przygotowany dla dziesięciu, w przypadku struktury listy, oraz ośmiu w przypadku BST, wielkości próbek – 100, 200, 400, 800, 1600, 3200, 6400 oraz 12800 i dodatkowo 25600 i 51200 dla list. Czas wykonania algorytmu dla każdej z tych próbek został przekazany do wykresu i odpowiednich tabel. Zawarte dane są uśrednione dla dziesięciu kolejnych uruchomień programu.

3. Lista kierunkowa – porządek rosnący



Wykres 1. Czas dodawania elementów do listy



Wykres 2. Czas wyszukiwania elementów listy

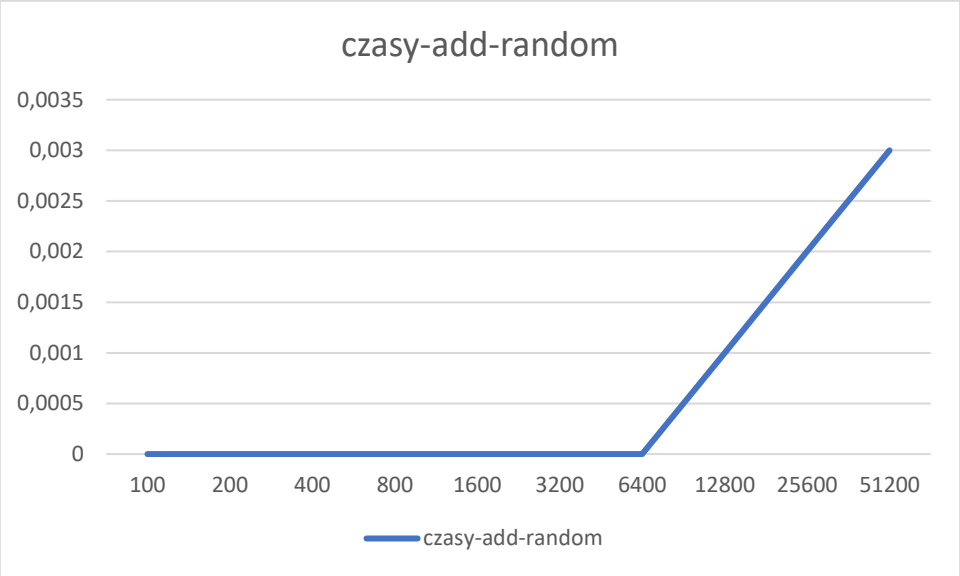
Liczba elementów	Czas dodawania	Czas wyszukiwania
100	0	0
200	0	0
400	0	0
800	0	0,001
1600	0	0,005
3200	0	0,022
6400	0	0,087
12800	0	0,364
25600	0,002	1,488
51200	0,004	5,974

Tabela 1. Dokładne czasy wykonania funkcji dla listy kierunkowej

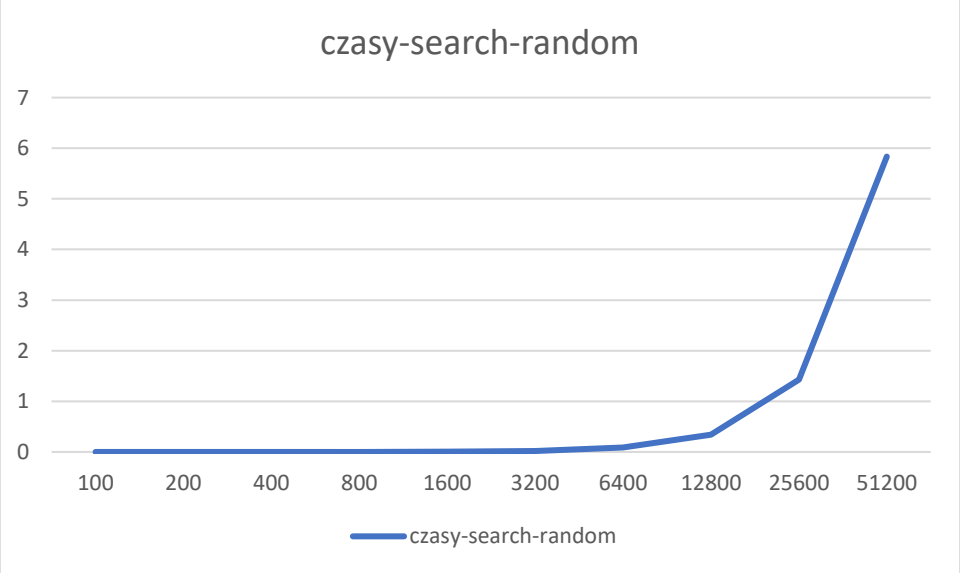
W przypadku mniejszych próbek widać że czas działania był znacząco mały zarówno w przypadku dodawania elementów, jak i ich wyszukiwania. Dopiero 800 i więcej elementów rozróżniło czasy działania między funkcjami, i to dość znacząco. Dla takiej ilości próbek można też zauważyć liniowość działania funkcji – dla podwojonej ich ilości w każdym wypadku funkcja wykonywała się około czterokrotnie dłużej. Potwierdziło się to również po dodaniu do przykładu kolejnych próbek, 102400 i 204800 – czasy wykonania dla tych próbek wynosiły odpowiednio około 24 i 98 sekund.

Po zwiększeniu liczby próbek zachowanie to wpłynęło też na funkcję dodawania. Tu jednak zachowanie było inne – między próbką 51200 i 102400 różnica była marginalna, niezależnie od tego, ile razy uruchomiona była funkcja (między pozostałymi próbkami funkcja wykonywała się ok. dwa razy wolniej, tu na poziomie kilku milisekund). Między 102400 i 204800 zachowanie było jednak podobne do tego, co w mniejszych przypadkach.

4. Lista porządkowa – kolejność losowa



Wykres 3. Czas dodawania elementów do listy – kolejność losowa



Wykres 4. Czas wyszukiwania elementów listy – kolejność losowa

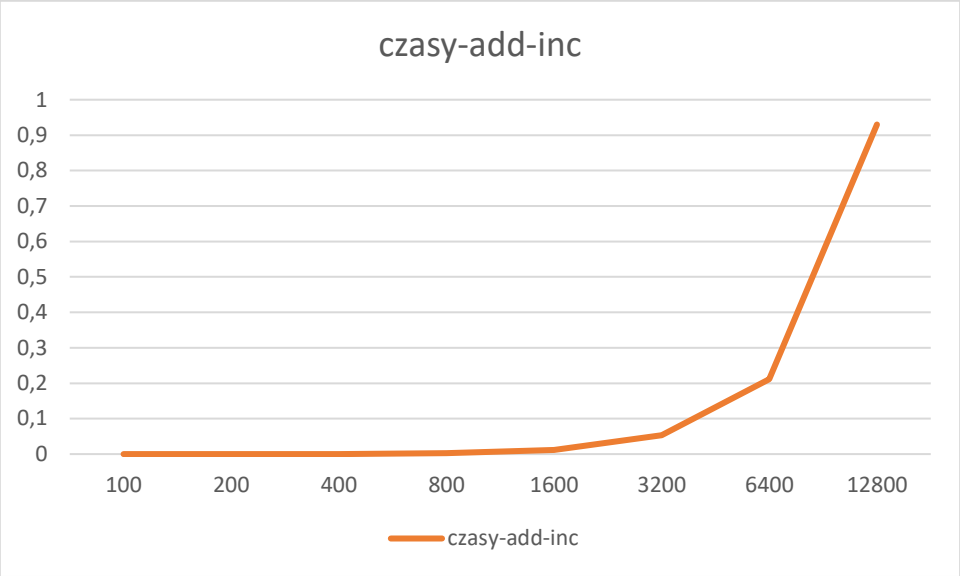
Liczba elementów	Czas dodawania	Czas wyszukiwania
100	0	0
200	0	0
400	0	0
800	0	0,001
1600	0	0,006
3200	0	0,02
6400	0	0,085
12800	0,001	0,342
25600	0,002	1,426
51200	0,003	5,834

Tabela 2. Dokładne czasy wykonania funkcji dla listy kierunkowej – kolejność losowa

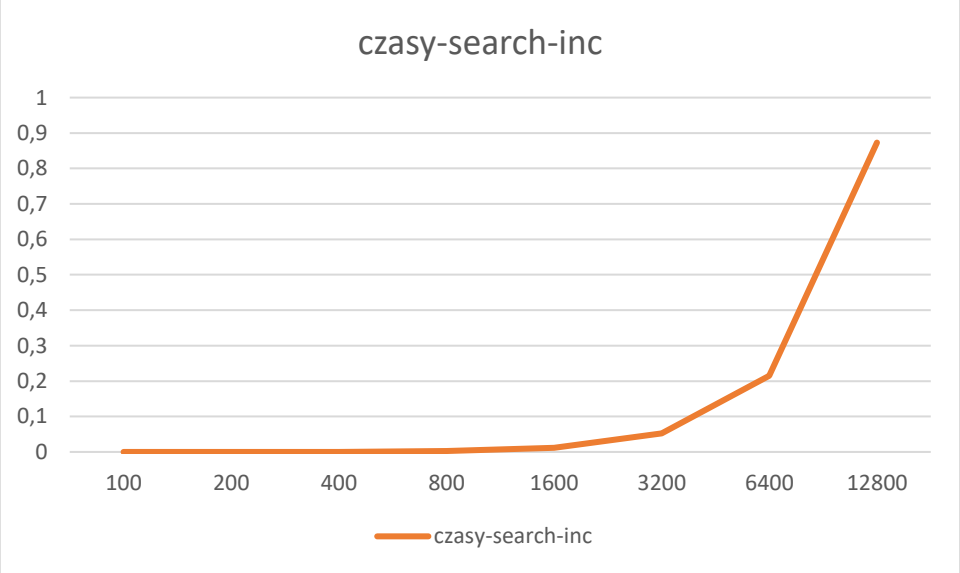
Zachowanie funkcji w przypadku listy z elementami w kolejności losowej jest bardzo podobne co w poprzednim punkcie. Dopiero dla większej ilości przypadków można zauważyć zwiększenie czasu wykonania funkcji wyszukiwania – czterokrotnie, dla podwojonej ilości próbek. Zachowanie to zauważyć można również w przypadku dodania kolejnych próbek.

Dodawanie elementów zachowywało się jednak inaczej niż poprzednio, stopniowo o kilka milisekund (około 3-4) dla przypadków powyżej 25600. Nie zaobserwowane zostało to zjawisko co poprzednio (dot. próbki 51200 i 102400).

5. Drzewo BST – porządek rosnący



Wykres 5. Czasy dodawania elementów do drzewa w porządku rosnącym

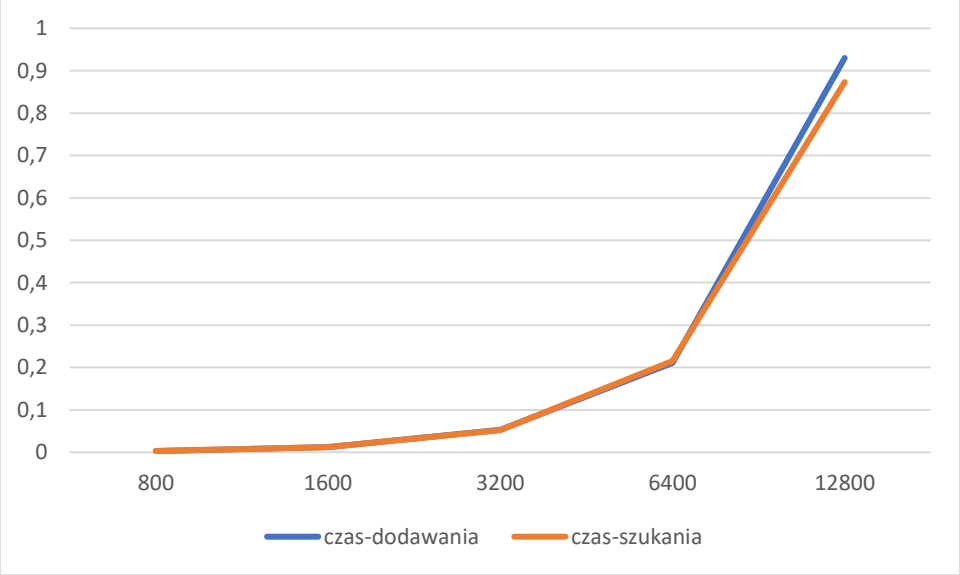


Wykres 6. Czasy wyszukiwania elementów w drzewie w porządku rosnącym

Liczba elementów	Czas dodawania	Czas szukania
100	0	0
200	0	0
400	0	0
800	0,003	0,003
1600	0,012	0,012
3200	0,053	0,052
6400	0,211	0,215
12800	0,93	0,873

Tabela 3. Dokładne czasy wykonania funkcji dla drzewa BST – porządek rosnący

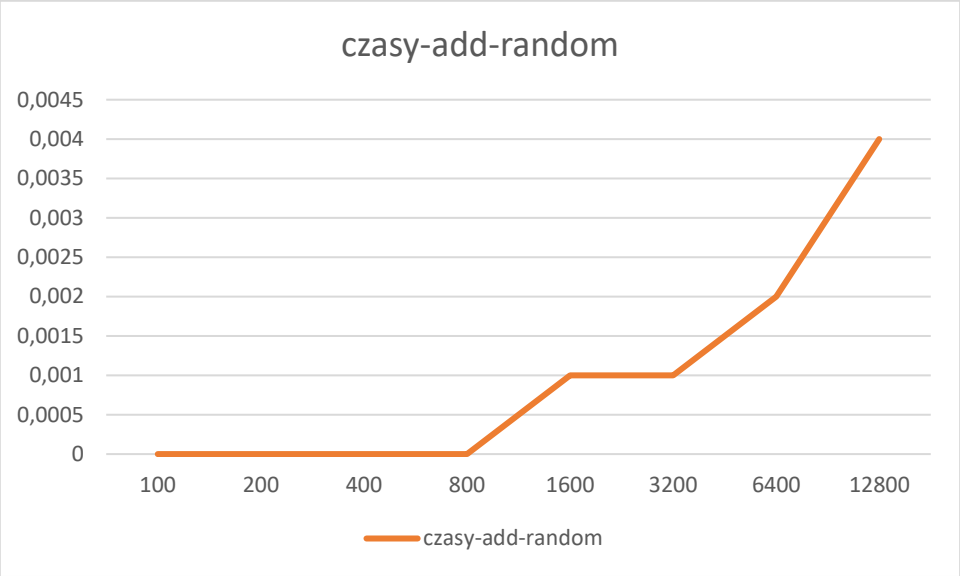
BST zachowało się inaczej niż lista, w obydwu przypadkach. Jest to bardziej złożona struktura danych niż poprzednio omawiana, co można zauważyć chociażby w zapisanych czasach wykonywania funkcji. Funkcja dodawania elementów oraz szukania zachowały się bardzo podobnie, niezależnie od liczby elementów lub uruchomień – zajmowało to podobną ilość czasu.



Wykres 7. Czasy dodawania i szukania dla większej ilości próbek

Czas wykonywania funkcji w przypadku drzewa w kolejności rosnącej był największy. Był to jednak najtrudniejszy przypadek, zważywszy na to, że zachowane musiały zostać odpowiednie zasady, aby w dalszym ciągu zachowana została struktura drzewa.

6. Drzewo BST – kolejność losowa



Wykres 8. Czasy dodawania elementów do drzewa w kolejności losowej

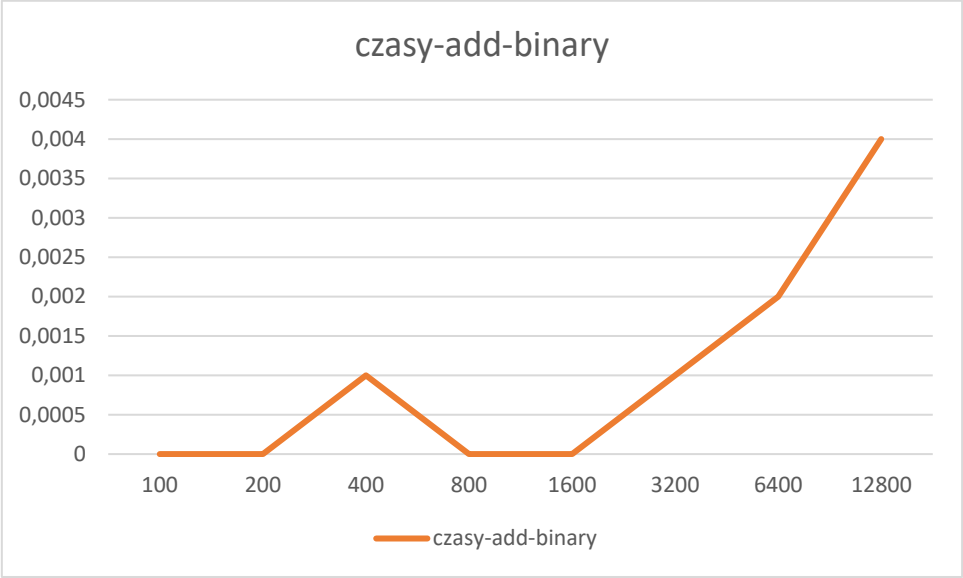
Wykres 9. Czasy wyszukiwania elementów w drzewie w kolejności losowej

Liczba elementów	Czas dodawania	Czas szukania
100	0	0
200	0	0
400	0	0
800	0	0,001
1600	0,001	0
3200	0,001	0
6400	0,002	0,002
12800	0,004	0,004

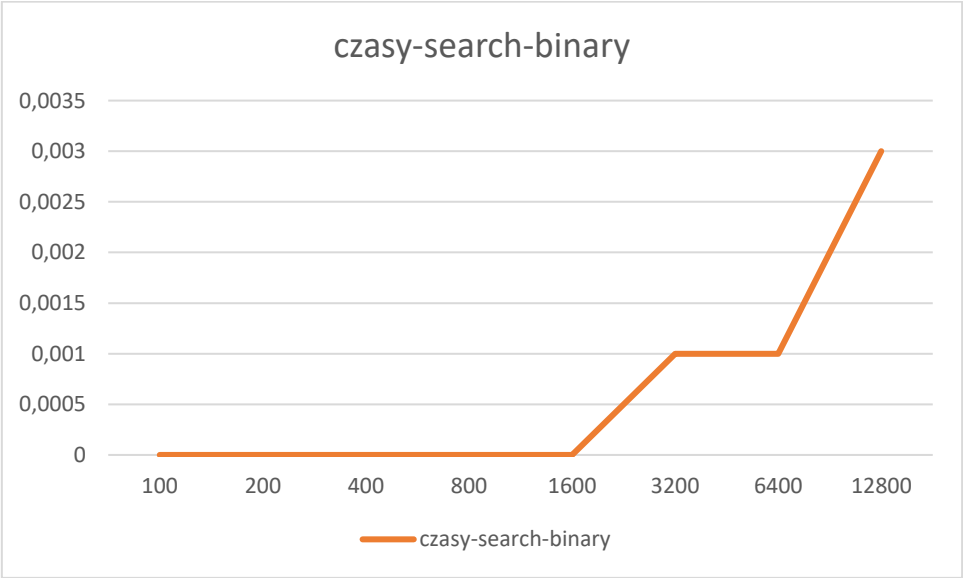
Tabela 4. Dokładne czasy wykonania funkcji dla drzewa BST – kolejność losowa

Funkcje dla drzewa w kolejności losowej wykonywały się znacznie szybciej niż w przypadku drzewa w porządku rosnącym. Zauważyć tu można jednak jedno specyficzne zachowanie – skok w czasie wykonywania funkcji wyszukiwania dla mniejszej próbki, zauważalny na wykresie dla próbki 800. Po kilkukrotnym sprawdzeniu próbki czas działania faktycznie był nieco większy dla pojedynczych próbek, niezależnie od liczby uruchomień – zazwyczaj w przypadkach 400, 800 i 1600. Spowodowane to mogło być większą złożonością dla takiej liczby próbek, podczas gdy kolejny przypadek mógł się okazać dla algorytmu łatwiejszy.

7. Drzewo BST – reguła połowienia binarnego



Wykres 10. Czas dodawania elementów do drzewa wg. reguły połowienia



Wykres 11. Czas wyszukiwania elementów w drzewie wg. reguły połowienia

Liczba elementów	Czas dodawania	Czas szukania
100	0	0
200	0	0
400	0,001	0
800	0	0
1600	0	0
3200	0,001	0,001
6400	0,002	0,001
12800	0,004	0,003

Tabela 5. Dokładny czas działania funkcji dla drzewa BST – reguła połowienia binarnego

Dla reguły połowienia binarnego, nazywanego również metodą bisekcji, można zauważyć podobny czas wykonania funkcji co w przypadku losowym, ale też – jak w funkcji wyszukiwania w kolejności losowej – podczas dodawania występuje powtarzalne zwiększenie czasu, malejące w kolejnym kroku. To również może być spowodowane nieco zwiększoną złożonością dla mniejszego przypadku. Zachowanie pozostałych przypadków jest jednak bardzo zbliżone dla poprzednich metod – czasy dodawania i wyszukiwania elementów w drzewie są podobne, jeśli nie identyczne.