

# Sprawozdanie Struktury danych i złożoność obliczeniowa

Maksim Birszel nr indeksu 241353

27 maja 2019

## 1 Złożoności obliczeniowe operacji na strukturach według literatury

- Tablica

- Dostęp do określonego elementu: natychmiastowy, ze względu na indeksowanie każdej komórki tablicy -  $O(1)$
- Wyszukanie elementu: o określonej wartości, w najgorszym wypadku będziemy musieli odczytać wszystkie  $n$  komórek tablicy -  $O(n)$
- Wstawienie nowego elementu na określoną pozycję: wstawienie nowego elementu na koniec tablicy wymaga po prostu powiększenia rozmiaru tablicy o 1 i wpisania wartości do ostatniej komórki (zatem jest to stała liczba operacji, niezależna od liczby elementów, czyli  $O(1)$ ), jednak wstawienie nowego elementu w środek tablicy wymaga powiększenia rozmiaru tablicy o 1 oraz przesunięcia wszystkich elementów leżących na prawo od wstawianego o 1 pozycję w prawo, czyli w najgorszym wypadku (wstawienie elementu na pierwszą pozycję): stała liczba operacji zwiększania rozmiaru tablicy +  $n$  przesunięć + 1 zapisanie =  $O(1) + O(n) + O(1) = O(n)$ .
- Usunięcie elementu: analogicznie do wstawiania, w najgorszym wypadku (usunięcie pierwszego elementu) trzeba przesunąć  $n-1$  elementów o 1 pozycję w lewo ( $n-1$  operacji) i zmniejszyć rozmiar tablicy o 1 (stała liczba operacji) -  $O(n)$ .

- Lista

- Dostęp do elementu: jeżeli element jest umieszczony na końcu listy, to aby do niego dotrzeć, trzeba przejrzeć wszystkie pozostałe  $n-1$  elementów -  $O(n)$ .
- Wyszukanie elementu: jak wyżej -  $O(n)$ .
- Wstawienie elementu: należy utworzyć nową komórkę  $O(1)$ , wypełnić wartość pola danych  $O(1)$ , pole wskaźnika wypełnić wartością głowy  $O(1)$ , a wartość głowy wypełnić wartością wskazującą na nowy element  $O(1)$  - całkowity czas omawianej operacji wynosi zatem  $O(1)$ .

- Usunięcie elementu: wartość wskaźnika elementu poprzedzającego należy wypełnić wartością wskazującą na element następny  $O(n)$  (trzeba ten wskaźnik znaleźć!) i usunąć z pamięci dany element  $O(1)$  -  $O(n)$

- **Kopiec**

- Dostęp do elementu: w kopcu mamy dostęp tylko do korzenia -  $O(1)$ .
- Usunięcie (pobranie) elementu: procedura pobrania (pierwszego) elementu wymaga, po odczytaniu go z korzenia, przywrócenia właściwości kopca. Polega to na tym, że ostatni element ze struktury wstawiamy do korzenia (w miejsce pobranego elementu) i sukcesywnie zamieniamy go z większym z potomków dopóki ten potomek jest większy od zamienianego elementu. Zatem w najgorszym wypadku wykonamy  $O(\log n)$  takich zamian, co stanowi złożoność obliczeniową całej procedury.
- Wstawienie elementu: nowy element wstawiamy na koniec struktury i zamieniamy go z rodzicem dopóki rodzic jest mniejszy od wstawionego elementu -  $O(\log n)$ .
- Biorąc pod uwagę powyższe, utworzenie poprawnego kopca ze zbioru  $n$  losowych wartości sprowadza się do  $n$ -krotnego wykonania operacji wstawienia elementu -  $O(n \log n)$ .

**Podsumowanie:**

Struktura	Dostęp	Wyszukiwanie	Wstawianie	Usunięcie
Tablica	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Lista	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Kopiec	$O(1)$	$O(n \log n)$	$O(\log n)$	$O(\log n)$

## 2 Plan eksperymentu

- Wielkość struktur - eksperyment został przeprowadzony dla kilku rozmiarów każdej ze struktur. Kolejno dla 10 elementów, 100 elementów, 1000 elementów oraz 10000 elementów. Dla każdej z nich pomiar czasu został powtórzony wielokrotnie a wynik tych pomiarów uśredniony.
- Sposób generowania elementów struktur - elementy są generowane losowo i zapisane w pliku tekstowym, z którego każdorazowo są pobierane.
- Sposób pomiaru czasu - Do pomiaru czasu została użyta zaproponowana funkcja QueryPerformanceCounter.
- Wyniki w tabelach zostaną przedstawione w milisekundach
- Liczbą używaną podczas obliczeń jest 4 bitowy int

Tablica 1: Tabela wartości zmierzonych dla operacji na tabeli

Elem.	Dostęp	Wysz.	Wst(0)	Wst(ost)	Wst(dow)	Usu(0)	Usu(dow)	Usu(ost)
10	0.00035	0.031	0.0014	0.0012	0.0014	0.00071	0.00076	0.0035
100	0.024	0.042	0.027	0.0044	0.0017	0.0098	0.0011	0.023
1000	3.8	0.095	0.089	0.012	0.0025	0.024	0.24	0.045
10000	230	0.23	0.13	0.14	0.0012	0.064	0.079	0.075

Tablica 2: Tabela wartości zmierzonych dla operacji na liście w milisekundach

Elem.	Dostęp	Wysz.	Wst(0)	Wst(ost)	Wst(dow)	Usu(0)	Usu(dow)	Usu(ost)
10	0.21	0.019	0.0017	0.00012	0.00045	0.00036	0.0071	0.036
100	0.39	0.028	0.0019	0.00018	0.00051	0.00035	0.014	0.044
1000	3.8	0.048	0.0018	0.00013	0.00064	0.00036	0.032	0.078
10000	22	0.39	0.013	0.00014	0.00061	0.00036	0.089	0.082

Tablica 3: Tabela wartości zmierzonych dla operacji na kopcu

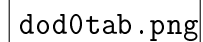
Ilość elementów	Dostęp	Wyszukiwanie	Wstawianie	Usuwanie
10	0.00024	0.00035	0.027	0.022
100	0.0043	0.0066	0.032	0.054
1000	0.073	0.082	0.068	0.078
10000	0.21	0.22	0.15	0.19

- Wszystkie struktury danych alokowane są dynamicznie, oraz realokowane po dodaniu/usunięciu elementu.

### 3 Zestawienie wyników eksperymentu

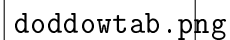
Tylko dla tablicy i listy:

- Wst(0) - wstawienie na pozycji 0
- Wst(ost) - wstawienie na pozycji ostatniej
- Wst(dow) - wstawienie na dowolnej pozycji
- Usu(0) - usunięcie elementu z pozycji 0
- Usu(ost) - usunięcie ostatniego elementu
- Usu(dow) - usunięcie elementu z dowolnej pozycji
- Wysz. - wyszukiwanie elementu w strukturze



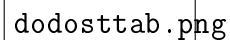
dod0tab.png

Rysunek 1: Dodanie elementu na pozycji 0 w tablicy



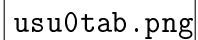
doddowtab.png

Rysunek 2: Dodawanie elementu na pozycji dowolnej w tablicy



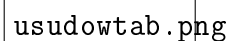
dodosttab.png

Rysunek 3: Dodawanie elementu na pozycji ostatniej w tablicy



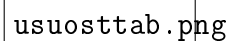
usu0tab.png

Rysunek 4: Usuwanie elementu na pozycji 0 w tablicy



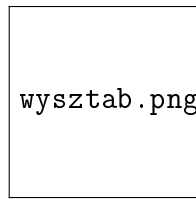
usudowtab.png

Rysunek 5: Usuwanie elementu na pozycji dowolnej z tablicy

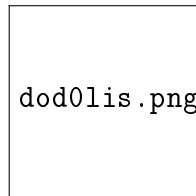


usuosttab.png

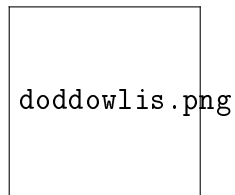
Rysunek 6: Usuwanie elementu z pozycji ostatniej z tablicy



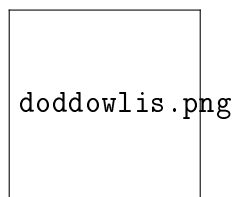
Rysunek 7: Wyszukiwanie elementu w tablicy



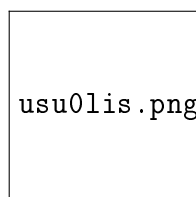
Rysunek 8: Dodanie elementu na pozycji 0 w liscie



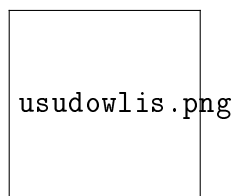
Rysunek 9: Dodawanie elementu na pozycji dowolnej w liscie



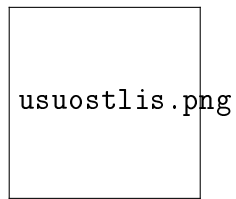
Rysunek 10: Dodawanie elementu na pozycji ostatniej w liscie



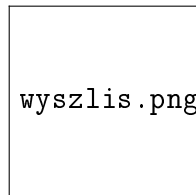
Rysunek 11: Usuwanie elementu na pozycji 0 w liscie



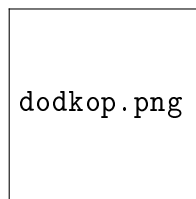
Rysunek 12: Usuwanie elementu na pozycji dowolnej z listy



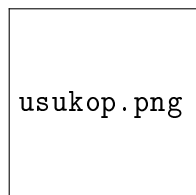
Rysunek 13: Usuwanie elementu z pozycji ostatniej z listy



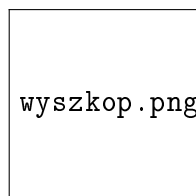
Rysunek 14: Wyszukiwanie elementu w liscie



Rysunek 15: Dodawanie elementu do kopca



Rysunek 16: Usuwanie elementu z kopca



Rysunek 17: Wyszukiwanie elementu w kopcu

## 4 Wnioski

Większość operacji pokrywa się z założeniami podawanymi przez literature, jednak niektóre z nich przyjmują np czas liniowy, podczas, gdy powinny mieć dostęp stały. Jest to prawdopodobnie spowodowane realokacją pamięci, której czasu nie uwzględniamy w obliczeniach. Jeśli chodzi o dostęp do poszczególnych elementów to najszybsza okazuje się tablica oraz kopiec (ponieważ jego elementy są zaimplementowane jako elementy tablicy). Mają one dostęp stały do poszczególnych elementów, a lista liniowy. Możemy również zauważyć, że wartości elementów nie mają wpływu na długość operacji w tablicy oraz liście, czego nie możemy powiedzieć o kopcu.

Dzięki zachowanej własności kopca jest on najszybszy pod względem wyszukiwania elementów. Złożoność tej operacji wynosi  $O(n \log n)$  podczas, gdy w tablicy i liście jest to liniowy dostęp  $O(n)$ . Podobny rezultat pokazują nam wyniki pomierzonych czasów dostępu do danej operacji. Im więcej elementów, tym dłuższy czas oczekiwania na rezultat ze względu na to, że przeszukujemy większy zbiór.

Wstawianie elementów najszybciej odbywa się w liście pod warunkiem, że wstawiamy na jej koniec lub początek. W przeciwnym wypadku musimy przejść po pozostałych elementach, aby wyszukać wskaźnik na wymaganą przez nas pozycję, co znacznie wydłuża czas operacji. Podczas wstawiania elementów do kopca dodajemy je jako ostatni element struktury a następnie wywołujemy operację przywrócenia własności kopca, co daje nam w rezultacie złożoność  $O(\log n)$ . W tablicy jak widzimy po wynikach obliczeń, czas dodawania nowych liczb zależy od aktualnej ilości elementów, jakie przechowuje. Podczas wstawiania elementu musimy przesunąć pozostałe (z wyjątkiem wstawienia na koniec tablicy, jednak nawet w tym wypadku musimy przekopiować tablice do drugiej o nowej ilości elementów, a starą usunąć, aby uniknąć wycieków pamięci.

Usuwanie elementów w liście oraz tablicy ma złożoność liniową, tzn im więcej elementów, tym dłużej dana operacja się wykonuje, co potwierdzają wyniki eksperymentu. Z kolei w kopcu sytuacja wygląda podobnie jak podczas operacji wstawiania, tzn. złożoność wynosi  $(\log n)$ , ponieważ po usunięciu danego elementu musimy na nowo przywrócić własność kopca.

Rozbieżności w wynikach pomiarów nie są idealnie precyzyjne. Jest to spowodowane wieloma czynnikami: liczby wybierane są losowo z określonego zakresu, algorytmy są mojego autorstwa i mogą zawierać nieoptymalne rozwiązania niektórych operacji, obliczenia mogły zostać wykonane zbyt małą liczbą razy, aby dawały wiarygodne uśrednione wyniki, komputer na którym wykonywano obliczenia mógł w danym momencie wykonywać inne operacje przez co wykonana w tej chwili operacja mogła zostać opóźniona oraz wiele innych czynników.