

Sprawozdanie

## 1. Wstęp

Sprawdzano złożoność obliczeniową algorytmu zapęnlającego daną strukturę. Mierzono czas dla wypełnienia struktur o rozmiarach:

1, 10, 100, 200, 300, 400, 500, ..., 1000, 2000, 3000, ..., 10000, 20000, 30000, ..., 100000, 200000, ..., 1000000  
(dla listy do 700000)

Każdy test wykonano 10 razy, czas uśredniono.

Do testów wykorzystano następujące implementacje struktur danych:

- **Ta blicę dynamiczną**, której rozmiar zwiększano o 1 za każdym dodaniem nowego elementu. (stos, funkcja push())

```
6  template<class T>
7  void Stack<T>::push(T data){
8      if (addCount < sizeStc)
9          stack[addCount] = data;
10     else
11         stack.push_back(data);
12     addCount++;
13 }
```

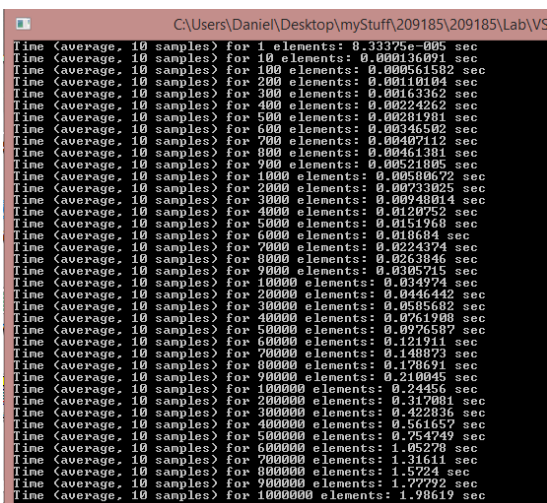
- **Ta blicę dynamiczną**, której rozmiar zwiększano dwukrotnie po każdorazowym wypełnieniu. (stos, funkcja push\_prc() )

```
15  template<class T>
16  void Stack<T>::push_prc(T data){
17      //duplicate memory
18      if (addCount == sizeStc)
19      {
20          stack.resize(sizeStc * 2);
21          sizeStc *= 2;
22      }
23      stack[addCount] = data;
24      addCount++;
25 }
```

- **Listę**, opartą na wskaźnikach. (dodawanie na koniec listy)  
(lista, funkcja push() )

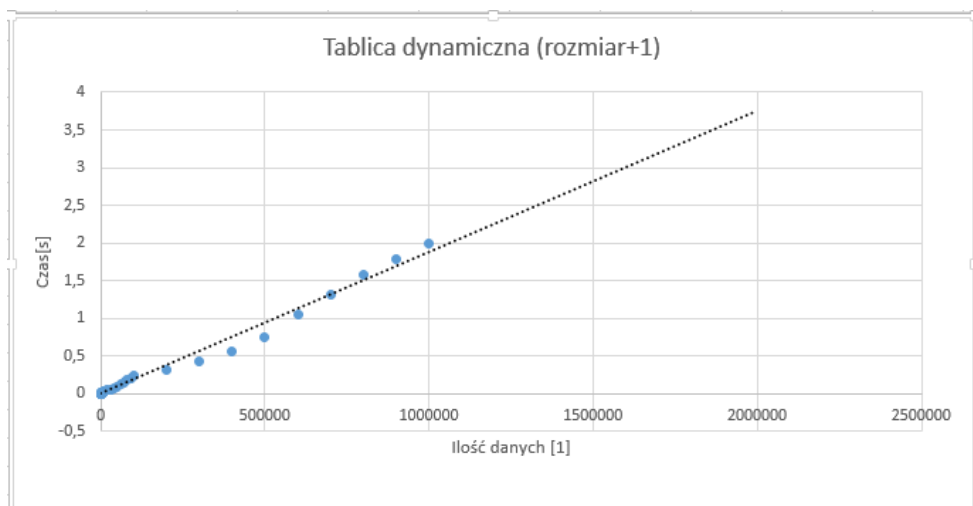
```
7  void List<T>::push(T data){
8      // first element to the list
9      if (sizeLst == 0){
10         tailPtr = new Node<T>(data);
11         headPtr = tailPtr;
12         tempPtr = tailPtr;
13         sizeLst++;
14     }
15     else{
16         tempPtr = tailPtr;
17         tailPtr = new Node<T>(data);
18         tempPtr->nextNode = tailPtr;
19         tempPtr = tailPtr;
20         sizeLst++;
21     }
22 }
```

## 1.Tablica dynamiczna (rozmiar zwiększany o 1)



```
C:\Users\Daniel\Desktop\myStuff\209185\209185\Lab\VS
Time (average, 10 samples) for 1 elements: 0.33375e-005 sec
Time (average, 10 samples) for 10 elements: 0.000136091 sec
Time (average, 10 samples) for 100 elements: 0.000561582 sec
Time (average, 10 samples) for 200 elements: 0.00110104 sec
Time (average, 10 samples) for 300 elements: 0.00163362 sec
Time (average, 10 samples) for 400 elements: 0.00224262 sec
Time (average, 10 samples) for 500 elements: 0.00281981 sec
Time (average, 10 samples) for 600 elements: 0.00346502 sec
Time (average, 10 samples) for 700 elements: 0.00409712 sec
Time (average, 10 samples) for 800 elements: 0.00461381 sec
Time (average, 10 samples) for 900 elements: 0.00521805 sec
Time (average, 10 samples) for 1000 elements: 0.00580672 sec
Time (average, 10 samples) for 2000 elements: 0.00733825 sec
Time (average, 10 samples) for 3000 elements: 0.00948014 sec
Time (average, 10 samples) for 4000 elements: 0.0120752 sec
Time (average, 10 samples) for 5000 elements: 0.0151968 sec
Time (average, 10 samples) for 6000 elements: 0.018664 sec
Time (average, 10 samples) for 7000 elements: 0.0224374 sec
Time (average, 10 samples) for 8000 elements: 0.0263846 sec
Time (average, 10 samples) for 9000 elements: 0.0305715 sec
Time (average, 10 samples) for 10000 elements: 0.034974 sec
Time (average, 10 samples) for 20000 elements: 0.0446442 sec
Time (average, 10 samples) for 30000 elements: 0.0585682 sec
Time (average, 10 samples) for 40000 elements: 0.0761788 sec
Time (average, 10 samples) for 50000 elements: 0.0976587 sec
Time (average, 10 samples) for 60000 elements: 0.121911 sec
Time (average, 10 samples) for 70000 elements: 0.148873 sec
Time (average, 10 samples) for 80000 elements: 0.178691 sec
Time (average, 10 samples) for 90000 elements: 0.210045 sec
Time (average, 10 samples) for 100000 elements: 0.24456 sec
Time (average, 10 samples) for 200000 elements: 0.317081 sec
Time (average, 10 samples) for 300000 elements: 0.422336 sec
Time (average, 10 samples) for 400000 elements: 0.561657 sec
Time (average, 10 samples) for 500000 elements: 0.754749 sec
Time (average, 10 samples) for 600000 elements: 1.05278 sec
Time (average, 10 samples) for 700000 elements: 1.31611 sec
Time (average, 10 samples) for 800000 elements: 1.5724 sec
Time (average, 10 samples) for 900000 elements: 1.77792 sec
Time (average, 10 samples) for 1000000 elements: 1.98619 sec
```

Rysunek 1 Wywołanie testu



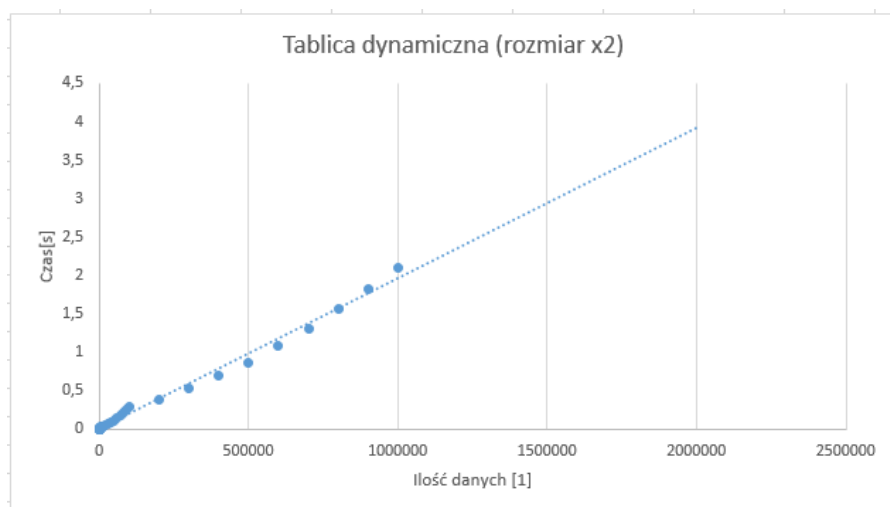
Rysunek 2 Wynik testów wraz z idealnym przebiegiem funkcji liniowej

W przypadku dodawania elementów do tablicy, wraz z zwiększaniem kolejnych rozmiarów o 1, złożoność teoretyczną zakładam jako  $O(n)$ , ponieważ wraz z kolejnym rozszerzeniem rozmiar zwiększa się tylko o 1 element.

## 2.Tablica dynamiczna (rozmiar zwiększany x2)

```
C:\Users\Daniel\Desktop\myStuff\209185\209185\Lab\
Time (average, 10 samples) for 1 elements: 9.48325e-005 sec
Time (average, 10 samples) for 10 elements: 0.000380588 sec
Time (average, 10 samples) for 100 elements: 0.000758066 sec
Time (average, 10 samples) for 200 elements: 0.00134106 sec
Time (average, 10 samples) for 300 elements: 0.00200988 sec
Time (average, 10 samples) for 400 elements: 0.0027914 sec
Time (average, 10 samples) for 500 elements: 0.0036618 sec
Time (average, 10 samples) for 600 elements: 0.00464223 sec
Time (average, 10 samples) for 700 elements: 0.00494068 sec
Time (average, 10 samples) for 800 elements: 0.0056419 sec
Time (average, 10 samples) for 900 elements: 0.00635456 sec
Time (average, 10 samples) for 1000 elements: 0.00705160 sec
Time (average, 10 samples) for 2000 elements: 0.00868983 sec
Time (average, 10 samples) for 3000 elements: 0.0109158 sec
Time (average, 10 samples) for 4000 elements: 0.0137812 sec
Time (average, 10 samples) for 5000 elements: 0.0173324 sec
Time (average, 10 samples) for 6000 elements: 0.021183 sec
Time (average, 10 samples) for 7000 elements: 0.0255523 sec
Time (average, 10 samples) for 8000 elements: 0.0304317 sec
Time (average, 10 samples) for 9000 elements: 0.0354004 sec
Time (average, 10 samples) for 10000 elements: 0.0405647 sec
Time (average, 10 samples) for 20000 elements: 0.051669 sec
Time (average, 10 samples) for 30000 elements: 0.0602277 sec
Time (average, 10 samples) for 40000 elements: 0.0901057 sec
Time (average, 10 samples) for 50000 elements: 0.114382 sec
Time (average, 10 samples) for 60000 elements: 0.142804 sec
Time (average, 10 samples) for 70000 elements: 0.170819 sec
Time (average, 10 samples) for 80000 elements: 0.215938 sec
Time (average, 10 samples) for 90000 elements: 0.255362 sec
Time (average, 10 samples) for 100000 elements: 0.297463 sec
Time (average, 10 samples) for 200000 elements: 0.300055 sec
Time (average, 10 samples) for 300000 elements: 0.520903 sec
Time (average, 10 samples) for 400000 elements: 0.694234 sec
Time (average, 10 samples) for 500000 elements: 0.86895 sec
Time (average, 10 samples) for 600000 elements: 1.09061 sec
Time (average, 10 samples) for 700000 elements: 1.31314 sec
Time (average, 10 samples) for 800000 elements: 1.57392 sec
Time (average, 10 samples) for 900000 elements: 1.83228 sec
Time (average, 10 samples) for 1000000 elements: 2.10156 sec
```

Rysunek 3 Wywołanie testu



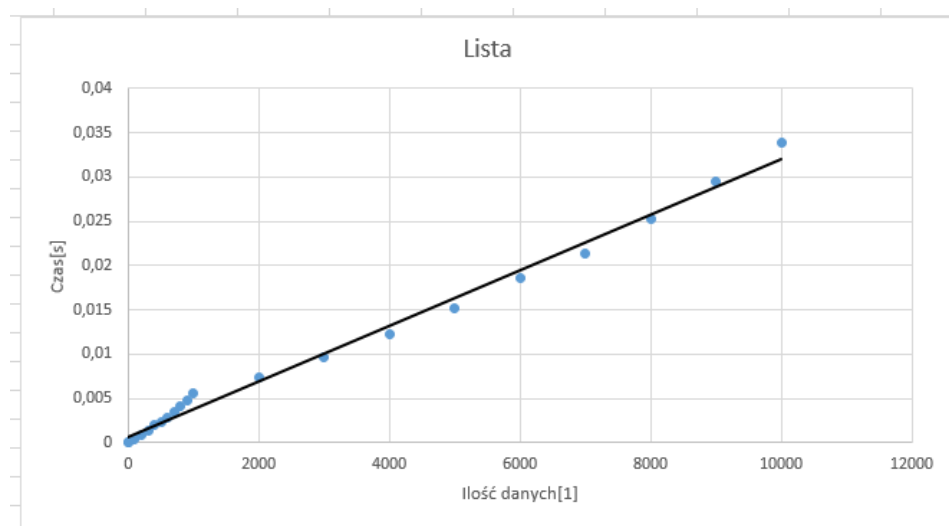
Rysunek 4 Wynik testów wraz z idealnym przebiegiem funkcji liniowej

W przypadku dodawania elementów do tablicy z powiększaniem rozmiarem dwukrotnie, teoretyczna złożoność wynosi  $O(2n)$ , ponieważ za każdym razem zwiększamy ilość elementów dwukrotnie.

### 3.Lista (wskaźniki)

```
C:\Users\Daniel\Desktop\myStuff\209185\209185\La
Time (average, 10 samples) for 1 elements: 0.00306064 sec
Time (average, 10 samples) for 10 elements: 0.00162943 sec
Time (average, 10 samples) for 100 elements: 0.00153349 sec
Time (average, 10 samples) for 200 elements: 0.00173693 sec
Time (average, 10 samples) for 300 elements: 0.00205721 sec
Time (average, 10 samples) for 400 elements: 0.00253579 sec
Time (average, 10 samples) for 500 elements: 0.00343144 sec
Time (average, 10 samples) for 600 elements: 0.00388057 sec
Time (average, 10 samples) for 700 elements: 0.0045289 sec
Time (average, 10 samples) for 800 elements: 0.00519859 sec
Time (average, 10 samples) for 900 elements: 0.00566842 sec
Time (average, 10 samples) for 1000 elements: 0.00622509 sec
Time (average, 10 samples) for 2000 elements: 0.00737398 sec
Time (average, 10 samples) for 3000 elements: 0.0101815 sec
Time (average, 10 samples) for 4000 elements: 0.012843 sec
Time (average, 10 samples) for 5000 elements: 0.0162473 sec
Time (average, 10 samples) for 6000 elements: 0.0197179 sec
Time (average, 10 samples) for 7000 elements: 0.023579 sec
Time (average, 10 samples) for 8000 elements: 0.0278149 sec
Time (average, 10 samples) for 9000 elements: 0.032407 sec
Time (average, 10 samples) for 10000 elements: 0.0372433 sec
Time (average, 10 samples) for 20000 elements: 0.0422113 sec
Time (average, 10 samples) for 30000 elements: 0.0620117 sec
Time (average, 10 samples) for 40000 elements: 0.0824516 sec
Time (average, 10 samples) for 50000 elements: 0.105989 sec
Time (average, 10 samples) for 60000 elements: 0.134226 sec
Time (average, 10 samples) for 70000 elements: 0.164056 sec
Time (average, 10 samples) for 80000 elements: 0.19242 sec
Time (average, 10 samples) for 90000 elements: 0.222514 sec
Time (average, 10 samples) for 100000 elements: 0.2556 sec
Time (average, 10 samples) for 200000 elements: 0.324144 sec
Time (average, 10 samples) for 300000 elements: 0.427055 sec
Time (average, 10 samples) for 400000 elements: 0.563889 sec
Time (average, 10 samples) for 500000 elements: 0.73774 sec
Time (average, 10 samples) for 600000 elements: 0.939541 sec
Time (average, 10 samples) for 700000 elements: 1.16279 sec
```

Rysunek 5 Wywołanie testu



Rysunek 6 Wynik testów wraz z idealnym przebiegiem funkcji liniowej

Dla listy operującej na wskaźnikach, teoretyczna złożoność jest liniowa,  $O(3n)$ , za każdym razem dodajemy jedynie nowy wskaźnik wraz z przemieszczeniem starych wskaźników na kolejne/poprzednie elementy.