PAMSI – testowanie implementacji struktur danych

Piotr Wilkosz

16/03/2014

1 Wstęp

Celem ćwiczenia było przetestowanie implementacji takich struktur danych jak:

- Stos — zawierający listę lub tablicę dynamiczną
- Kolejka — zawierająca listę lub tablicę dynamiczną

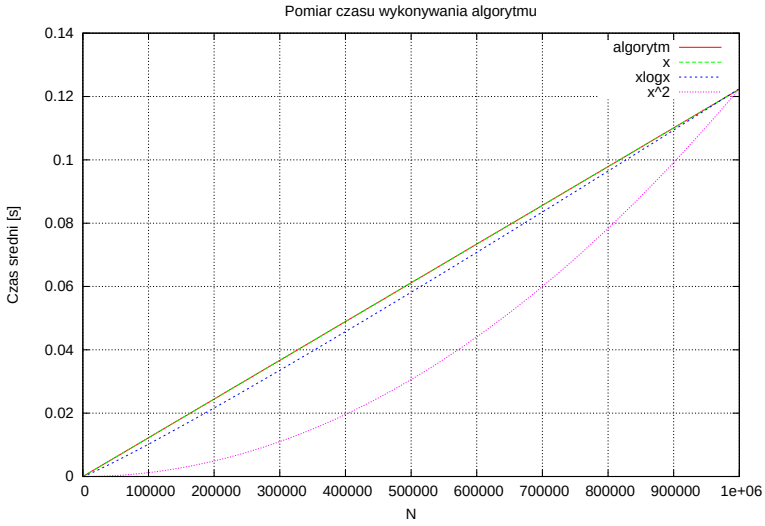
Zadaniem było zmierzenie czasu wykonywania operacji wypełnienia powyższych struktor danych.

2 Wyniki pomiarów

1. Stos bazujący na liście:

Tablica 1: Pomiar czasu zapełnienia stosu bazującego na liście

N	czas	odchylenie
10	3.1777e-06	9.66352e-07
100	1.41498e-05	1.37796e-06
1000	0.000129143	1.37132e-05
10000	0.00125259	0.000136966
50000	0.00610872	0.000647302
100000	0.0148653	0.00293036



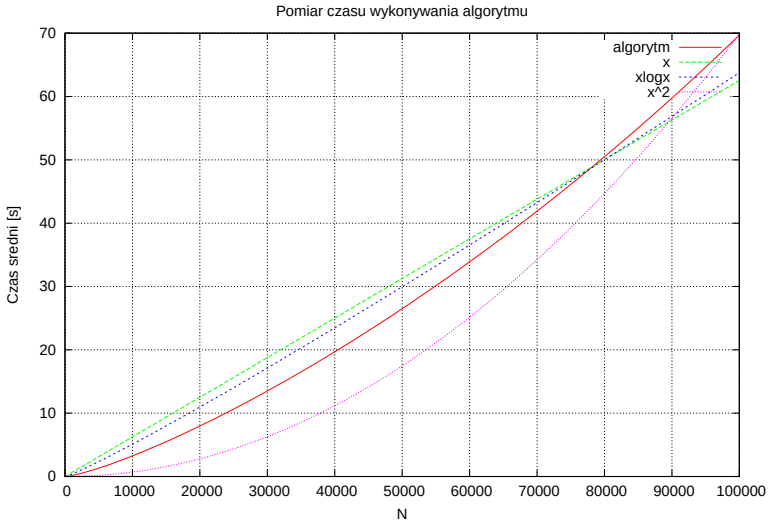
Rysunek 1: Wykres do tabeli 1

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia stosu bazującego na liście szacuje się na $O(n)$.

2. Stos bazujący na tablicy dynamicznej, każdorazowo powiększającej swój rozmiar:

Tablica 2: Pomiar czasu zapełnienia stosu bazującego na tablicy dynamicznej; każdorazowe powiększanie pamięci

N	czas	odchylenie
10	5.2315e-06	1.84447e-06
100	9.61087e-05	1.17477e-05
1000	0.00626336	0.000569029
10000	0.602328	0.0547606
50000	16.1983	1.47384
100000	63.4257	6.11571



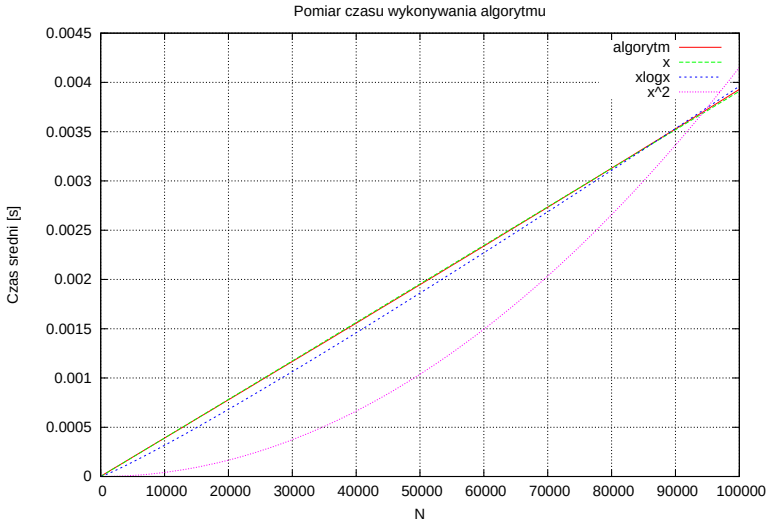
Rysunek 2: Wykres do tabeli 2

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia stosu bazującego na tablicy każdorazowo powiększającej swój rozmiar szacuje się na $O(n^2)$.

3. Stos bazujący na tablicy podwajającej swój rozmiar po zapełnieniu stosu:

Tablica 3: Pomiar czasu zapełnenia stosu bazującego na tablicy; podwajanie pamięci

N	czas	odchylenie
10	2.9054e-06	4.54692e-07
100	7.941e-06	7.84532e-07
1000	3.41665e-05	3.28946e-06
10000	0.000393291	4.00849e-05
50000	0.0017202	0.000172796
100000	0.00372878	0.00034095



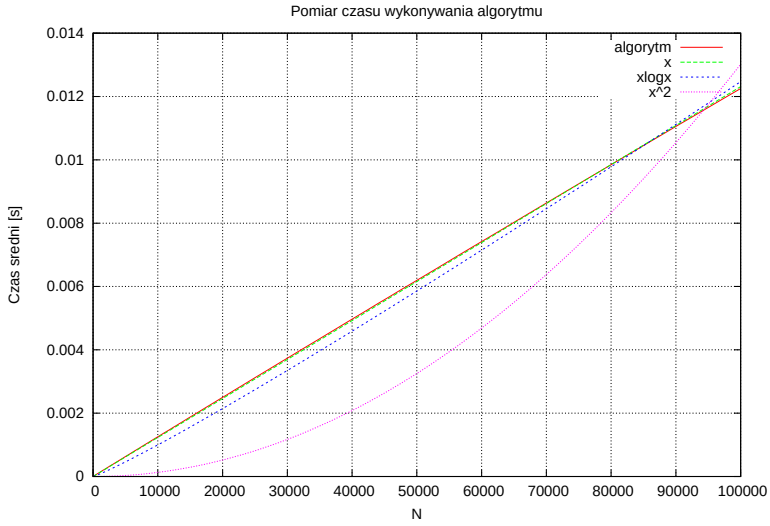
Rysunek 3: Wykres do tabeli 3

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia stosu bazującego na tablicy podwajającej swój rozmiar szacuje się na $O(n)$.

4. Kolejka bazująca na liście:

Tablica 4: Pomiar zapełnienia koljeki bazującej na liście

N	czas	odchylenie
10	2.9402e-06	4.02804e-07
100	1.43175e-05	1.45537e-06
1000	0.000123787	1.32083e-05
10000	0.00121058	0.000122136
50000	0.0060172	0.000595832
100000	0.012138	0.00127857



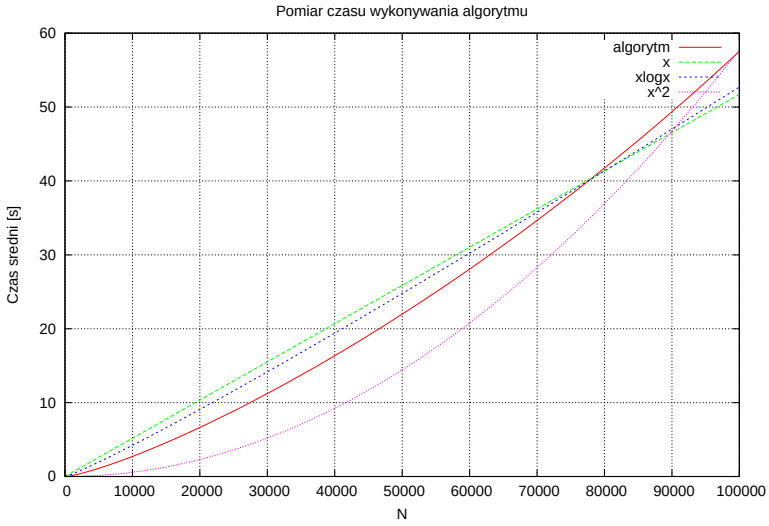
Rysunek 4: Wykres do tabeli 4

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia kolejki bazującej na liście szacuje się na $O(n)$.

5. Kolejka bazująca na tablicy każdorazowo powiększającej swój rozmiar:

Tablica 5: Pomiar czasu zapełnienia kolejki bazującej na tablicy; każdorazowe powiększanie pamięci

N	czas	Odchylenie
10	4.4419e-06	1.43333e-06
100	9.5843e-05	8.89311e-06
1000	0.00578163	0.000530114
10000	0.56162	0.0531706
50000	15.1532	1.37704
100000	57.5228	5.79702



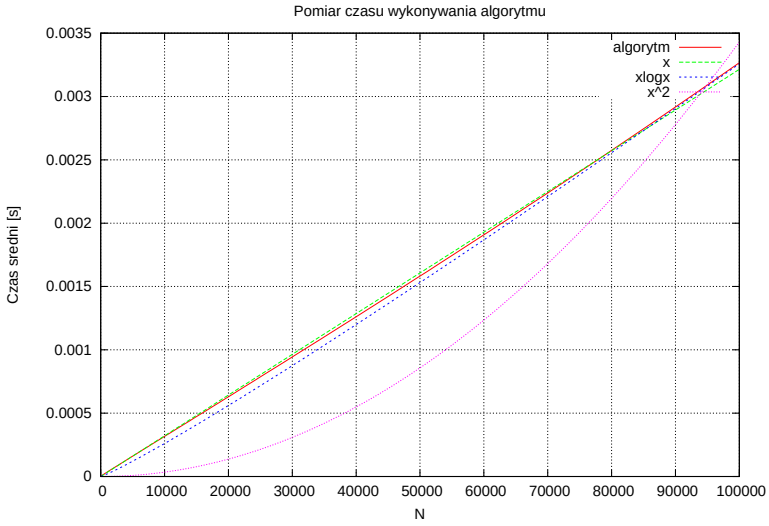
Rysunek 5: Wykres do tabeli 5

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia kolejki bazującej na tablicy każdorazowo powiększającej swój rozmiar szacuje się na $O(n^2)$.

6. Kolejka bazująca na tablicy podwajającej swój rozmiar po zapełnieniu kolejki:

Tablica 6: Pomiar czasu zapełnienia kolejki bazującej na tablicy; podwajanie pamięci

N	czas	Odchylenie
10	2.7934e-06	5.70677e-07
100	6.977e-06	7.54621e-07
1000	3.06953e-05	4.08372e-06
10000	0.000342118	3.42047e-05
50000	0.00149814	0.000148217
100000	0.00326706	0.000321491



Rysunek 6: Wykres do tabeli 6

Na podstawie wykresu złożoność obliczeniową algorytmu zapełnienia kolejki bazującej na tablicy podwajającej swój rozmiar szacuje się na $O(n)$.

3 Wnioski

- Najbardziej wydajne pod względem szybkości wykonania okazały się struktury wykorzystujące listę lub tablicę podwajającą swój rozmiar pod wypełnieniu struktury Złożoność obliczeniową takiej implementacji szacuje się na $O(n)$
- Struktury, które każdorazowo zwiększały rozmiar tablicy działają dużo wolniej, aczkolwiek są oszczędniejsze pod względem zagospodarowania pamięci. Ich złozoność obliczeniową szacuje się na $O(n^2)$