PAMSI - Sprawozdanie 3

Filip Guzy 218672

23 kwietnia 2016

Implementacja struktur danych

W celu realizacji struktur danych takich jak lista, stos i kolejka zaimplementowano trzy interfejsy: IList, IStack oraz IQueue. Zawierają one wirtualne metody opisujące każdą ze struktur. Na bazie interfejsu listy jest możliwe zaimplementowanie dwóch pozostałych typów danych, jednak aby zwiększyć przejrzystość projektowanych struktur lepiej zastosować dla nich oddzielne interfejsy. Jednym z istotnych ograniczeń dla języka C++ jest możliwość dziedziczenia metod tylko po jednym interfejsie, w przeciwieństwie do Javy lub C#, gdzie programista ma możliwość tworzenia obiektu na bazie kilku interfejsów. Opisy metod oraz sposoby ich działania zostały zawarte w plikach nagłówkowych odpowiednich implementacji ADT. Warto również uwzględnić, że są to implementacje węzłowe, wykorzystujące dynamicznie alokowane węzły jako kolejne elementy listy, kolejki lub stosu, co będzie miało znaczenie przy wykonywanych poniżej pomiarach.

Przeszukiwanie listy

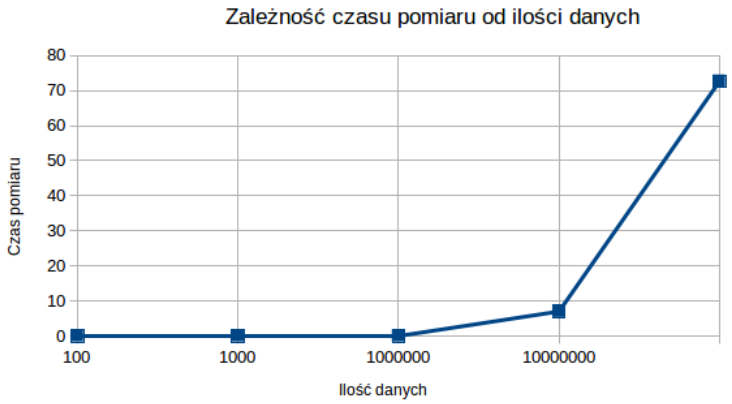
W implementacji klasy listy zawarto metodę przeznaczoną do sprawdzania czasu przeszukiwania listy dla dowolnej ilości danych. W celu zbadania złożoności obliczeniowej przeszukiwania list o różnych rozmiarach zmierzono czasy (w milisekundach) wykonania tej czynności dla następujących danych wejściowych: 10, 100, 1000, 1000000, 10000000. Nie wykonano pomiarów dla 1000000000 elementów, ponieważ w implementacji listy wykorzystano węzły alokowane dynamicznie jako kolejne elementy listy, przez co dla danych wejściowych wymagana pamięć przekroczyła dostępną pamięć RAM maszyny wirtualnej. Pomiary dla każdej ilości danych wejściowych powtórzono dziesięciokrotnie, a wyniki przedstawiono w poniższej tabeli:

Ilość danych	10	100	1000	1000000	10000000
Czas [ms]	0,00199997	0,00100005	0,00999999	6,898	70,193
	0,000999928	0,000999928	0,0140001	6,832	72,634
	0,00199997	0,000999928	0,015	6,998	87,602
	0,00100005	0,000999928	0,00999999	7,009	71,251
	0,00100005	0,000999928	0,0140001	7,077	69,218
	0,000999928	0,000999928	0,0129999	6,842	71,151
	0,00100005	0,00100005	0,0139999	6,978	71,436
	0,00100005	0,000999928	0,0120001	6,961	68,91
	0,000999928	0,000999928	0,013	7,224	71,44
	0,000999928	0,000999928	0,00800002	6,913	72,68

Dla każdej ilości danych wejściowych wyznaczono średnią arytmetyczną. Wyniki przedstawiono w poniższej tabeli:

Ilość danych	10	100	1000	1000000	10000000
Czas [ms]	0,0011999852	0,0009999524	0,01230001	6,9732	72,6515

Otrzymane dane przedstawiono również na wykresie:



Aby sprawdzić, czy pomiary są zgodne z założeniami przeanalizowano fragment kodu odpowiedzialny za przeszukanie tablicy:

```
while(tmp− >next) { // przechodzimy wszystkie wezly
if(tmp− >elem>0) {
// jeśli znajdziemy szukany element, robimy jakąś operację, tutaj żadną
}
else // jeżeli nie to
tmp=tmp− >next; // zmieniamy węzeł na kolejny
}
```

Wyrażenia warunkowe if i else mają złożoność obliczeniową równą O(1), tak jak operacja zmiany węzła, która następuje po każdym wywołaniu instrukcji else, więc wnętrze pętli while ma złożoność obliczeniową równą O(1). Biorąc pod uwagę pętlę while, która została użyta do przejścia przez wszystkie węzły listy, których jest n, złożoność obliczeniowa całego algorytmu przeszukania powinna wynosić O(n). Tak też jest w rzeczywistości: każde dziesięciokrotne zwiększenie ilości danych wejściowych powoduje dziesięciokrotne zwiększenie czasu wykonania algorytmu, zatem rzeczywista złożoność operacji jest liniowa, czyli zawiera się w O(n). Dysponując możliwością zmierzenia czasu dla większych ilości danych ta zależność byłaby lepiej widoczna na wykresie.

Wnioski

Otrzymane pomiary wskazują na to, że lista została zaimplementowana poprawnie i wszystkie operacje są wykonywane w odpowiednim czasie. Złożoność obliczeniowa algorytmu przeszukiwania listy należy do O(n), zatem jest zgodna z teorią.

Implementacja listy wykorzystująca węzły wykorzystuje więcej pamięci niż implementacja oparta np. na tablicy statycznej, jednak ograniczenie miejsca w przypadku tej drugiej uniemożliwia elastyczne operowanie miejscem w strukturze przy wykonywaniu praktycznych projektów.