Zadanie projektowe nr 1

Badanie efektywności operacji dodawania, usuwania oraz wyszukiwania elementów w różnych strukturach danych

Należy zaimplementować oraz dokonać pomiaru czasu działania operacji takich jak dodawanie, usuwanie i wyszukiwanie elementu w następujących strukturach danych:

a) Tablica,
b) Lista dwukierunkowa,
c) Kopiec binarny,
d) Drzewo poszukiwań binarnych (BST),
e) Drzewo czerwono-czarne.

Należy przyjąć następujące założenia:

- podstawowym elementem struktur jest 4 bajtowa liczba całkowita ze znakiem,
- wszystkie struktury danych powinny być alokowane dynamicznie (w przypadku tablic powinny zajmować jak najmniej miejsca – powinny być relokowane przy dodawaniu/usuwaniu elementów),
- dla tablicy i listy rozpatrzyć osobno operacje dodawania i usuwania elementu na następujących pozycjach:
    a. początek,
    b. koniec,
    c. losowe miejsce.
- dla kopca wystarczy usuwanie elementu ze szczytu, elementy dodawane powinny oczywiście być automatycznie umieszczane we właściwych miejscach tak, aby zachowany był warunek kopca,
- należy zmierzyć czasy wykonywania poszczególnych operacji w funkcji rozmiaru danej struktury (liczby przechowywanych elementów). W przypadku zbyt krótkich czasów można albo zwiększyć liczbę danych pomiarowych, albo powtórzyć pomiar na przykład 10 razy. Ponieważ wyniki zależą także od rozkładu danych, to pomiary dla konkretnego rozmiaru struktury należy wykonać wielokrotnie (na przykład 100 razy, za każdym razem generując nową populację) a wynik uśrednić. Liczbę przechowywanych elementów trzeba dobrać eksperymentalnie (może to być np. 1000, 2000, 5000, 10000, 100000 lub więcej) w zależności od wydajności sprzętu,
- należy mieć na uwadze, że czas wykonywania operacji może zależeć od wartości przechowywanych elementów, trzeba to uwzględnić w pomiarach i wnioskach,
- dodatkową funkcją programu musi być możliwość sprawdzenia poprawności zbudowanej struktury i zaimplementowanych operacji dla przykładowych danych, będzie to potrzebne przy oddawaniu projektu (szerzej na ten temat w dalszej części). Warto podkreślić, że takie sprawdzenie konieczne jest również przed wykonaniem testów czasowych (tracą one sens, jeśli będą przeprowadzone dla nieprawidłowo działających algorytmów),

- sposoby dokładnego pomiaru czasu w systemie Windows podano na stronie www:

  http://staff.iiar.pwr.wroc.pl/antoni.sterna/sdizo/SDiZO_time.pdf

- dopuszczalnymi językami programowania są języki kompilowane do kodu natywnego (na przykład C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Phyton),
- używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa),
- nie wolno korzystać z gotowych bibliotek takich jak STL, Boost lub innych przy implementacji głównych algorytmów. Można ich użyć tylko w operacjach pomocniczych (takich jak przygotowanie danych testowych, wyświetlanie kopca lub drzewa). Wszystkie algorytmy i struktury muszą być zaimplementowane samodzielnie (nie kopiować gotowych rozwiązań),
- implementacja powinna być wykonana w postaci jednego programu,
- kod źródłowy powinien być odpowiednio komentowany,
- program musi być skompilowany do wersji EXE (w takiej wersji zostanie poddany testom),
- warto pamiętać o dużych różnicach w wynikach testów czasowych pomiędzy wersjami *Debug* i *Release* (testy trzeba przeprowadzić w wersji *Release*).

Sprawdzenie poprawności zbudowanej struktury/operacji obejmuje:

- utworzenie struktury z liczb zapisanych w pliku tekstowym. Pierwsza z liczb określa rozmiar struktury danych, kolejne reprezentują wartości elementów. Liczby w pliku będą rozdzielone białymi znakami (spacja, tabulacja, przejście do nowej linii). Nie powinna mieć znaczenia liczba i rodzaj białych znaków (na przykład spacje mogą być pojedyncze lub wielokrotne, liczby mogą być dowolnie grupowane w liniach).
- wyświetlenie struktury na ekranie (dla drzew zaproponować możliwie przejrzystą formę prezentacji),
- możliwość wykonania elementarnych operacji na strukturze (dodawanie, usuwanie, wyszukiwanie) W każdym przypadku powinna być możliwość wprowadzenia odpowiednich danych (na przykład pozycja tablicy i wartość do wstawienia na tej pozycji). Opisane operacje najlepiej zrealizować w formie menu dla każdej struktury,
- program powinien być odporny, przynajmniej w podstawowym zakresie, na podanie niewłaściwych danych (na przykład ujemnego indeksu tablicy).

Sprawozdanie powinno zawierać:

- krótki wstęp, w którym powinny być przedstawione złożoności obliczeniowe operacji w implementowanych strukturach na podstawie literatury,
- plan eksperymentu, czyli założenia co do wielkości struktur, sposobu generowania ich elementów, sposobu pomiaru czasu, itp.

- zestawienie wyników w formie tabelarycznej i graficznej (czas wykonania operacji w funkcji liczby elementów), wyniki należy przedstawić osobno dla poszczególnych operacji. Trzeba pamiętać o podaniu jednostek czasu, odpowiednio dobranych do wartości (np. czasy poniżej sekundy podać jako 1,2 ms a nie 0,0012 s). Liczba cyfr po przecinku powinna być "rozsądna", uwzględniająca nieuniknione błędy pomiarów (np. 12,3 ms a nie 12,3456789 ms),
- wnioski dotyczące efektywności poszczególnych struktur w zależności od zastosowań, wielkości struktury itp. Wskazać przyczyny rozbieżności (jeśli występują) pomiędzy złożonościami uzyskanymi eksperymentalnie a teoretycznymi,
- załączony kod źródłowy w formie elektronicznej (skopiować cały projekt oraz wersję skompilowaną programu).

Ocena projektu:

3.0 – eksperymenty na tablicy, liście i kopcu binarnym

3.5 – eksperymenty na tablicy, liście i kopcu binarnym **(program w wersji obiektowej)**

4.0 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie BST (tylko podstawowe operacje: dodawanie, usuwanie i wyszukiwanie elementów, nie jest wymagane równoważenie drzewa) **(program w wersji obiektowej)**

4.5 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie BST (dodawanie, usuwanie i wyszukiwanie elementów, równoważenie drzewa na żądanie z wykorzystaniem algorytmu DSW, operacje rotacji w prawo i w lewo dla wskazanego węzła) **(program w wersji obiektowej)**

5.0 – eksperymenty na tablicy, liście, kopcu binarnym i drzewie czerwono-czarnym **(program w wersji obiektowej)**

5.5 – eksperymenty na tablicy, liście, kopcu binarnym, drzewie czerwono-czarnym i drzewie AVL (program w wersji obiektowej). Dodatkowo pomierzyć czasy operacji dla tablicy i listy implementowanej jako kontenery z biblioteki STL (odpowiednio *vector* i *list*) i porównać je z czasami dla własnej implementacji tych struktur