AiZO - zadanie projektowe nr 1

Badanie efektywności wybranych algorytmów sortowania ze względu na złożoność obliczeniową

Należy wybrać i zaimplementować (samodzielnie) określone algorytmy sortowania, a następnie przeprowadzić analizę ich efektywności. Zakładamy, że po sortowaniu elementy powinny być uporządkowane rosnąco. Wskazane jest użycie szablonów (templates), aby łatwo można było wykorzystać zaimplementowane algorytmy do sortowania tablic zawierających elementy różnych typów (np. char, int, float).

Należy przyjąć następujące założenia:

1. Podstawowym elementem sortowanych tablic jest 4 bajtowa liczba całkowita ze znakiem (int), dla porównania należy rozpatrzyć również inne typy danych (np. char, float, double).
2. Wszystkie sortowane tablice powinny być alokowane dynamicznie (zgodnie z badanym rozmiarem tablicy).
3. Należy przeprowadzić **wstępną weryfikację poprawności sortowania** zaimplementowanych algorytmów!!! Dla małych tablic weryfikacja może być przeprowadzona wizualnie. Dla większych tablic może być przydatna procedura pomocnicza sprawdzająca poprawność uporządkowania elementów w tablicy.
4. Należy zmierzyć czasy sortowania tablic w funkcji ich rozmiaru. Ponieważ pojedynczy pomiar może być obarczony znacznym błędem oraz otrzymane wyniki mogą zależeć także od rozkładu danych, to pomiary dla konkretnego rozmiaru tablicy należy wykonać wielokrotnie (np. 100 razy – za każdym razem generując nowy zestaw danych), a wyniki uśrednić. W sprawozdaniu podajemy tylko wartości uśrednione (ewentualnie medianę i odchylenie standardowe). Badane rozmiary tablic należy dobrać eksperymentalnie w zależności od wydajności sprzętu (badania należy przeprowadzić dla 7 reprezentatywnych rozmiarów tablic np. 10000, 20000, 50000, 100000, … lub 10000, 20000, 40000, 80000, …). Wielkość tablic dobrać tak, aby pomiar czasu sortowania był na poziomie *ms* i więcej, a nie *ns* czy *us*. Pomiar powinien uwzględniać wyłącznie czas sortowania tablicy (nie powinien uwzględniać np. czasu generacji danych).
5. Ze względu na to, że czas sortowania tablicy może zależeć od początkowego rozkładu elementów należy rozpatrzyć przypadki szczególne (i umieścić w sprawozdaniu): tablica całkowicie losowa, tablica posortowana rosnąco, tablica posortowana malejąco, tablica posortowana częściowo (33% i 66% początkowych elementów już posortowanych), dodatkową funkcją programu musi być możliwość sprawdzenia poprawności

zaimplementowanych algorytmów sortowania (wczytanie danych z pliku i wyświetlenie tablicy przed i po sortowaniu).

6. Do dokładnego pomiaru czasu w systemie Windows w C++ można skorzystać z funkcji **QueryPerformanceCounter** lub ***`std::chrono::high_resolution_clock`*** (opis na stronie http://cpp0x.pl/forum/temat/?id=21331).
7. Dopuszczalnymi językami programowanie są języki kompilowane do kodu natywnego (np. C, C++), a nie interpretowane lub uruchamiane na maszynach wirtualnych (np. JAVA, .NET, Python) – możliwe jest odstępstwo od tej reguły za zgodą prowadzącego.
8. Używanie okienek nie jest konieczne i nie wpływa na ocenę (wystarczy wersja konsolowa).
9. Wszystkie algorytmy i struktury danych muszą być zaimplementowane przez studenta (nie kopiować gotowych rozwiązań).
10. Realizacja zadania powinna być wykonana w formie jednego programu.
11. Kod źródłowy powinien być komentowany.

Sprawdzenie poprawności algorytmów sortowania obejmuje:

1. Utworzenie dynamicznej tablicy na podstawie danych zapisanych w pliku tekstowym (każda liczba w osobnym wierszu). Pierwsza liczba określa rozmiar tablicy, pozostałe są kolejnymi elementami tablicy.
2. Wyświetlenie na ekranie tablicy przed sortowaniem.
3. Wyświetlenie na ekranie tablicy po sortowaniu.

W tym celu program powinien obsługiwać w trybie testowania parametry pliku konfiguracyjnego, które umożliwią następujące operacje:

a) wczytanie tablicy z pliku o zadanej nazwie,
b) wygenerowanie tablicy o zadanym rozmiarze zawierające losowe wartości,
c) wyświetlenie ostatnio utworzonej tablicy na ekranie (wygenerowanej lub wczytanej),
d) uruchomienie wybranego algorytmu na ostatnio utworzonej tablicy (do uruchomienia algorytmu używamy kopii utworzonej tablicy, program powinien umożliwić testowanie algorytmów dla tych samych danych),
e) wyświetlenie posortowanej tablicy na ekranie,

Lista dostępnych parametrów w pliku konfiguracyjnym może być rozszerzona o własne opcje.

Sprawozdanie (w formie elektronicznej) powinno zawierać:

1. ***wprowadzenie***, które powinno zawierać opis badanych algorytmów z omówieniem ich złożoności obliczeniowej (dla przypadku średniego i najgorszego) na podstawie literatury,
2. ***plan eksperymentu***, czyli założenia co do rozmiaru tablic, sposobu generowania tablic dla poszczególnych przypadków, sposobu pomiaru czasu, itp.,
3. ***omówienie przebiegu eksperymentów i przedstawienie uzyskanych wyników*** (w postaci tabel i wykresów), w przypadku wykresów jest to zawsze czas sortowania w funkcji rozmiaru tablicy (inne zmienne mogą być parametrem wykresu)
4. ***podsumowanie i wnioski*** (w przypadku niezgodności uzyskanych wyników z przewidywanymi należy spróbować wyjaśnić przyczyny),
5. ***literatura*** (materiały wykorzystane do wykonania projektu, również strony internetowe),
6. załączony ***kod źródłowy*** w formie elektronicznej (skopiować cały projekt oraz wersję skompilowaną programu),
7. jednostki w sprawozdaniu powinny być mianowane i mieć zachowaną odpowiednią precyzję.

Zakres liczby punktów za projekt (we wszystkich wersjach bada się zawsze wpływ rozmiaru tablicy oraz wstępnego ułożenia danych w tablicy na czas wykonania algorytmu):

0 – 69 – sortowanie przez wstawianie (insertion sort), przez scalanie (merge sort) i bąbelkowe (bubble sort) – dodatkowo badamy wpływ typu danych np. int i float

0 – 89 – sortowanie przez wstawianie zwykłe i binarne, przez kopcowanie (heap sort) i szybkie - program w wersji obiektowej, dodatkowo badamy wpływ typu danych np. int i float

0 – 100 – sortowanie przez wstawianie, przez kopcowanie, Shella i szybkie – program w wersji obiektowej, dodatkowo badamy wpływ typu danych np. int i float, wpływ wyboru odstępów dla algorytmu Shella (2 różne wzory tworzące algorytmy o różnych złożonościach) oraz sposób wyboru pivota (skrajny lewy, skrajny prawy, środkowy oraz losowy) dla algorytmu szybkiego.

Ze względu na nakład pracy badania wpływu typu danych można przeprowadzić tylko dla jednego wybranego algorytmu sortowania.

Literatura:

[1] Cormen T., Leiserson C.E., Rivest R.L., Stein C., Wprowadzenie do algorytmów, WNT

[2] Drozdek A., C++. Algorytmy i struktury danych, Helion