Część programistyczna

Należy zaimplementować algorytmy sortowania ciągu liczb naturalnych w porządku malejącym:

1. Shell Sort z algorytmem bazowym Insertion Sort oraz przyrostami Knutha,
2. Merge Sort,
3. Heap Sort,
4. Quick Sort rekurencyjny z pivotem, którym jest pierwszy element ciągu,
5. Quick Sort iteracyjny z pivotem, którym jest pierwszy element ciągu.

Dane wejściowe:

1. program powinien umożliwiać wczytanie z klawiatury *n*-elementowego ciągu liczb naturalnych (dla $n \leq 12$ – tryb demonstracyjny)
2. generowanie danych testowych przy użyciu wbudowanego generatora.

Dane wyjściowe: dla każdego uruchomienia algorytmu program powinien podawać

1. czas sortowania,
2. liczbę operacji porównań i zamian elementów (dla Merge Sort zliczamy tylko porównania),
3. dodatkowo w trybie demonstracyjnym (dla niewielkich ciągów), program powinien wyświetlać:
   - ciąg wejściowy i wyjściowy,
   - dla Quick sort – wartość pivota w każdej iteracji,
   - dla Merge sort – liczbę scaleń podzbiorów,
   - dla Shell sort – wartość przyrostu w kolejnych iteracjach.

Część eksperymentalna

Generowanie danych: dla każdego rozmiaru *n* należy wygenerować po 10 niezależnych ciągów liczb naturalnych dla każdej z następujących postaci danych: losowe, rosnące, malejące, A-kształtne (najpierw rosnące, następnie malejące) i V-kształtne (najpierw malejące, następnie rosnące). Przykład ciągu A-kształtnego: 1,3,5,7,8,6,4,2. Przykład ciągu V-kształtnego: 8,6,4,2,1,3,5,7,9.

Dwa niezależne eksperymenty:

1. Eksperyment 1 – wzrost liniowy. Należy wybrać 10–15 różnych wartości n rosnących liniowo, np.: 1000, 2000, 3000, 4000, … Zakres należy dobrać tak, aby możliwe było zaobserwowanie wzrostu czasu obliczeń wraz ze wzrostem n, a jednocześnie aby można wykonać wszystkie pomiary w rozsądnym czasie.
2. Eksperyment 2 – wzrost wykładniczy. Należy wybrać 10–15 wartości *n* rosnących wykładniczo, np.: 1000, 2000, 4000, 8000, 16000, 32000, … lub 10^3, 10^4, 10^5, 10^6, … Zakres należy dobrać tak, aby możliwe było zaobserwowanie charakteru wzrostu złożoności algorytmów, by różnice między algorytmami stały się wyraźne dla dużych danych, oraz aby eksperyment był możliwy do wykonania w rozsądnym czasie.

Procedura pomiarowa (dla obu eksperymentów): dla każdego algorytmu, typu danych oraz rozmiaru *n*, należy wygenerować 10 różnych ciągów i zmierzyć czas działania algorytmu. Następnie należy obliczyć średni czas sortowania (punkt na wykresie) oraz odchylenie standardowe. Każdy punkt pomiarowy musi być średnią z 10 uruchomień algorytmu. Wszystkie czasy należy zapisać w tabelach wyników.

Sprawozdanie

Sprawozdanie powinno zawierać:

1. Wykresy zależności czasu od liczby elementów według algorytmu. Należy przygotować 2N wykresów (N to liczba algorytmów sortowania) funkcji t=f(n) pokazującej zależność średniego czasu obliczeń t (oś Y) od liczby n elementów ciągu (oś X). Na wykresie należy zaznaczyć odchylenie standardowe (np. w postaci słupków błędów). Wykresy należy przygotować oddzielnie dla eksperymentu 1 (wzrost liniowy) i eksperymentu 2 (wzrost wykładniczy). Na każdym wykresie należy przedstawić K krzywych – po jednej dla każdego typu danych wejściowych. Celem jest pokazanie wpływu rodzaju danych na działanie danego algorytmu, porównanie kształtu wykresu dla wzrostu liniowego i wykładniczego, oraz analiza wpływu sposobu skalowania danych na czytelność różnic między algorytmami.

2. Wykresy zależności czasu od liczby elementów według rodzaju danych. Należy przygotować 2K wykresów (K to liczba typów danych wejściowych) funkcji t=f(n) zależności średniego czasu obliczeń t (oś Y) od liczby n elementów ciągu (oś X). Na wykresie należy zaznaczyć odchylenie standardowe (np. w postaci słupków błędów). Wykresy należy przygotować oddzielnie dla eksperymentu 1 (wzrost liniowy) i eksperymentu 2 (wzrost wykładniczy). Na każdym wykresie należy przedstawić N krzywych – po jednej dla każdego algorytmu sortującego. Celem jest porównanie efektywności algorytmów dla konkretnej postaci danych, jak również porównanie algorytmów przy szybkim wzroście rozmiaru danych, ocena, w którym eksperymencie różnice między algorytmami są bardziej widoczne, analiza, czy wzrost wykładniczy lepiej ujawnia charakter złożoności asymptotycznej.

3. Analizę porównawczą obu eksperymentów. Należy przeprowadzić bezpośrednie porównanie wyników uzyskanych w eksperymencie 1 i 2 oraz odpowiedzieć na pytania:
   - W którym eksperymencie łatwiej zaobserwować różnice między algorytmami?
   - Jak zmienia się kształt wykresów przy wzroście wykładniczym?
   - Czy wzrost wykładniczy lepiej ujawnia charakter złożoności obliczeniowej?
   - Czy dla małych wartości n różnice między algorytmami są istotne?
   - Jak zmienia się odchylenie standardowe wraz ze wzrostem n?

4. Analizę złożoności obliczeniowej. Dla każdego algorytmu należy podać złożoność w przypadku optymistycznym, średnim i pesymistycznym. Wyjaśnić związek między złożonością teoretyczną, liczbą wykonanych operacji i zaobserwowanym czasem działania. Odpowiedzieć na pytania: Czy wyniki eksperymentalne potwierdzają analizę teoretyczną? W jakich przypadkach występują odchylenia oraz z czego mogą wynikać? Z czego mogą wynikać różnice między teorią a pomiarem?

5. Wnioski końcowe. W podsumowaniu należy wskazać algorytm najbardziej efektywny w praktyce, określić algorytm najbardziej wrażliwy na postać danych wejściowych, porównać Quick sort rekurencyjny i iteracyjny, ocenić wpływ wyboru pivota na wyniki, wskazać, który sposób skalowania danych (liniowy czy wykładniczy) lepiej pokazuje rzeczywiste różnice między algorytmami.