Algorytmy i struktury danych

Laboratorium - lista 2

Termin wysłania: ~~2023-03-27~~ 2023-04-02

Zadanie 1. [40 p.]

Celem zadania jest zaimplementowanie i przetestowanie następujących algorytmów sortowania:

- INSERTION SORT ,
- MERGE SORT ,
- QUICK SORT ,

Zaimplementuj po jednym programie implementującym każdy z tych algorytmów.
Elementy sortowanej tablicy nazywamy *kluczami*.
Wejściem (przez standardowy strumień wejściowy) dla programu są kolejno:

- liczba *n* — długość sortowanej tablicy,
- tablica *n kluczy* do posortowania.

Ponadto zaimplementuj programy generujące na standardowym wyjściu dane zgodne z powyższym opisem w postaci:

- ciąg losowych kluczy (zadbaj o dobry generator pseudolosowy),
- ciąg posortowany rosnąco,
- ciąg posortowany malejąco.

Każdy taki generator, jako argument z linii poleceń przyjmuje rozmiar tablicy *n.*
Testy uruchamiaj jako potok postaci: `generator_danych n | program_sortujący`
Na potrzeby testów przyjmijmy, że *klucze* są liczbami całkowitymi od zera do 2^n-1.

Program sortujący powinien sortować tablicę wybranym algorytmem i wypisywać na standardowym wyjściu:

- Dla rozmiaru danych $n < 40$:
    - tablicę wejściową,
    - stany sortowanej tablicy w istotnych momentach (np. w MERGE SORT - po zakończeniu każdego scalania),
    - tablicę po sortowaniu.
      (Dla czytelności drukujmy klucze jako liczby dwucyfrowe.)
- Dla dowolnego rozmiaru danych, na końcu:
    - łączną liczbę porównań między *kluczami*,
    - łączną liczbę przestawień *kluczy*.
      (Warto zaimplementować osobne funkcje/procedury do porównywania i przestawiania *kluczy*, które dodatkowo zwiększają swój globalny licznik odpowiednio porównań lub przestawień.)

Finalnie, program sam sprawdza, czy wynikowy ciąg jest posortowany.

W prezentacji wykonaj testy dla długości tablicy *n* ($n \in \{8, 16, 32\}$) dla danych ciągów:

- losowego,
- posortowanego malejąco,
- posortowanego rosnąco.

Zadanie 2. [20 p.]

Wykorzystaj programy z zadania 1., aby porównać złożoności algorytmów.
Dla rozmiarów danych *n*, wykonaj po *k* niezależnych powtórzeń:

- sortowania ciągu każdym algorytmem,
- zapisania w pliku wykonanych liczb porównań i przestawień.

Wykorzystując zebrane wyniki, przedstaw na wykresach za pomocą wybranego narzędzia (np. numpy, Matlab, Mathematica):

- średnią liczbę wykonanych porównań (*c*) w zależności od *n*,
- średnią liczbę przestawień kluczy (*s*) w zależności od *n*,
- iloraz *c/n* w zależności od *n*,
- iloraz *s/n* w zależności od *n*.

Zadbaj o to, by dane dotyczące różnych algorytmów sortujących można było nakładać na te same osie i porównywać.
Sprawdź, jak wykresy zmieniają się dla różnych wartości *k* (np. $k = 1$, $k = 10$, $k = 100$).

- Dla wszystkich algorytmów przeprowadź eksperymenty dla $n \in \{10, 20, 30, \ldots, 200\}$.
- Dla algorytmów różnych od INSERTION SORT, przeprowadź eksperymenty dla $n \in \{1000, 2000, 3000, \ldots, 20000\}$.

Zadanie 3. [20 p.]

Uzupełnij Zadania 1. i 2. o algorytm DUAL-PIVOT QUICKSORT używając strategii COUNT :

- Mamy dwa pivoty *p* i *q* oraz załóżmy, że $p < q$.
- Załóżmy, że w procedurze PARTITION klasyfikując *i*-ty element tablicy mamy s_{i-1} elementów małych (mniejszych od *p*) oraz l_{i-1} elementów dużych (większych od *q*).
- Jeśli $l_{i-1} > s_{i-1}$, to porównuj *i*-ty element w pierwszej kolejności z q, a następnie, jeśli jest taka potrzeba, z *p*.
- Jeśli $l_{i-1} \leq s_{i-1}$, to porównuj *i*-ty element w pierwszej kolejności z *p*, a następnie, jeśli jest taka potrzeba, z *q*.

Dokonaj szczegółowych porównań otrzymanych statystyk dla algorytmu QUICKSORT i DUAL-PIVOT QUICKSORT. Eksperymentalnie wyznacz stałą stojącą przy czynniku $n \ln(n)$ dla liczby porównań między kluczami.

W prezentacji wykonaj testy dla długości tablicy *n* ($n \in \{8, 16, 32\}$) dla danych ciągów:

- losowego,
- posortowanego malejąco,
- posortowanego rosnąco.

Zadanie 4. [20 p.]

Uzupełnij Zadania 1. i 2. o algorytm *hybrydowy*, będący połączeniem dwóch wybranych algorytmów sortowania (z Zadania 1.).
Eksperymentalnie sprawdź, kiedy warto jest zmieniać moment przełączania się pomiędzy łączonymi algorytmami.
Zaproponowany algorytm powinien skutkować lepszymi statystykami od wcześniej zaimplementowanych algorytmów sortowania.
Na wykresach porównaj wyniki tego algorytmu z algorytmami, z których został stworzony.

W prezentacji wykonaj testy dla długości tablicy n ($n \in \{8, 16, 32\}$) dla danych ciągów:

- losowego,
- posortowanego malejąco,
- posortowanego rosnąco.