Algorytmy i struktury danych

Lista nr 3

Waga listy: 2

(termin oddania: 3 laboratorium)

Zadanie 1 (2 pkt) Celem zadania jest zaimplementowanie i przetestowanie następujących algorytmów sortowania:

- SelectSort,
- InsertionSort,
- HeapSort,
- QuickSort.

Program przyjmuje dwa parametry wejściowe `--type select|insert|heap|quick` – określający wykorzystywany algorytm sortowania oraz `--asc|--desc` – określający porządek sortowania. Wejściem dla programu są kolejno:

- liczba n – liczba elementów do posortowania,
- ciąg elementów do posortowania (niech elementy tej listy zostaną nazwane kluczami).

Program powinien sortować ciąg wybranym algorytmem i wypisywać na standardowym wyjściu błędów wykonywane operacje (porównania, przestawienia). Po zakończeniu sortowania, na standardowym wyjściu błędów powinna zostać wypisana łączna liczba porównań między kluczami, łączna liczba przestawień kluczy oraz czas działania algorytmu sortującego. Finalnie, program sprawdza, czy wynikowy ciąg jest posortowany zgodnie z wybranym porządkiem, a następnie wypisuje na standardowe wyjście liczbę posortowanych elementów oraz posortowany ciąg.

Przykładowe wywołanie:

```
./main --type quick --desc
5
9 1 -7 1000 4
```

Zadanie 2 (2 pkt) Uzupełnij program z **Zadania 1** o możliwość wywołania go z parametrem uruchomienia `--stat nazwa_pliku k`, wtedy pomija on wczytywanie danych i dla każdego $n \in \{100, 200, 300, \ldots, 10000\}$ wykonuje po k niezależnych powtórzeń:

- generowania losowego ciągu n elementowego (zadbaj o dobry generator pseudolosowy),
- sortowania kopii wygenerowanego ciągu każdym algorytmem,
- dla każdego z sortowań, zapisania do pliku `nazwa_pliku` statystyk odnośnie rozmiaru danych n, liczby wykonanych porównań między kluczami, liczby przestawień kluczy oraz czasu działania algorytmu sortującego.

Po zakończeniu programu, korzystając z zebranych danych przedstaw na wykresach, za pomocą wybranego narzędzia (np. numpy, Matlab, Mathematica):

- średnią liczbę wykonanych porównań kluczy (c) w zależności od n,
- średnią liczbę przestawień kluczy (s) w zależności od n,
- średni czas działania algorytmu w zależności od n,
- iloraz $\frac{c}{n}$ w zależności od n,
- iloraz $\frac{s}{n}$ 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 k (np. $k = 10, k = 1000$).

Zadanie 3 (1 pkt) Dodaj do poprzednich zadań modyfikację algorytmu QuickSort (ModyfiedQuickSort, opcja `--type mquick`), która wykonuje następujące czynności

- jeśli zostało co najwyżej 16 elementów to QuickSort zastąp przez InsertionSort,
- wybiera element dzielący jako medianę pierwszego, środkowego i ostatniego elementu (pamiętaj o porównaniach między tymi elementami).