LABORATORIUM NR 3

SORTOWANIE SZYBKIE (QUICKSORT)

Zadanie AL3.1
Tablicę $A = [23, 6, 11, 12, 17, 19, 7, 18, 12, 14, 15]$ sortujemy sortowaniem szybkim (`Quicksort`). W jaki sposób będzie zmieniała się ta tablica w czasie pierwszego przebiegu procedury `Partition` dokonującej podziału? Wskaż kolejne elementy, które są ze sobą zamieniane, oraz zaznacz, jak ostatecznie zostanie podzielona tablica.

Zadanie AL3.2 (5 pkt.)

- Zaimplementuj algorytm sortowania szybkiego omówiony na wykładzie. (4 pkt.)
- Zmierz i porównaj czasy działania sortowania dla dwóch rodzajów danych: dane losowe oraz dane skrajnie niekorzystne (np. liczby uprządkowane rosnąco lub malejąco). (1 pkt)

  Testy (pomiary czasu) powinny być wykonane dla różnych wielkości tablicy, ~~np. 100 elementów, 500 elementów, 1000 elementów i 2500 elementów. Ponadto w przypadku danych losowych należy wziąć średni czas (średnia arytmetyczna) z odpowiednio dużej próbki (np. 100, 500, 1000, 2500 losowań, odpowiednio, dla tablicy 100-, 500-, 1000- i 2500-elementowej).~~

  Sugerowany sposób prezentacji wyników testów.

```
rozmiar tablicy N  | (średni) czas - dane losowe | czas - dane niekorzystne
                   |                            |
                   |                            |
                   |                            |
                   |                            |
```

Specyfikacja wejścia/wyjścia tylko w przypadku realizacji pierwszej części zadania.

Wejście. Liczby (całkowite) zapisane w kolejnych wierszach pliku tekstowego.
Wyjście. Posortowane liczby z pliku wejściowego zapisane w kolejnych wierszach pliku wyjściowego.

Zadanie AL3.3 (5+1* pkt)
Zaimplementuj (5 pkt.) oraz przetestuj (1 pkt) podstawową/zmodyfikowaną wersję sortowania szybkiego, gdzie element wyznaczający podział jest:

- skrajnie prawym elementem $A[r]$ (pod)tablicy;
- wybranym (pseudo)losowo (`Randomized-Quicksort`);

- medianą (czyli elementem środkowym co do wartości) z trzech następujących elementów tablicy:
  — skrajnie lewego ($A[p]$),
  — środkowego ($A[p + \lfloor \frac{r-p}{2} \rfloor]$),
  — skrajnie prawego ($A[r]$).

Podobnie jak w zadaniu AL3.2, testy powinny być wykonane dla danych losowych i niekorzystnych, dla różnych wielkości tablicy (np. $100, 500, 1000$ i 2500). Dla tego samego rodzaju danych należy zestawić czasy działania trzech wersji algorytmów.

- Tworząc konkretną instancję I danych losowych lub niekorzystnych, każdy z tych algorytmów wykonywany jest dla tej samej instancji I, mierząc jego czas działania.
- Ze względu na „losową" naturę algorytmu `Randomized-Quicksort`, testy dla danych niekorzystnych (przy ustalonym rozmiarze tablicy) należy również wykonać wielokrotnie, a następnie uśrednić.

Sugerowany sposób prezentacji wyników testów.

```
DANE LOSOWE
rozmiar tablicy N  | algorytm 1 | algorytm 2 | algorytm 3
                   |            |            |
                   |            |            |
                   |            |            |
                   |            |            |

DANE NIEKORZYSTNE
rozmiar tablicy N  | algorytm 1 | algorytm 2 | algorytm 3
                   |            |            |
                   |            |            |
                   |            |            |
                   |            |            |
```

Zadanie AL3.4 (5+2* pkt.)

- Zaimplementuj algorytm sortowania szybkiego omówiony na wykładzie. (4 pkt.)
- Następnie zmodyfikuj ten algorytm tak, aby dla tablic o rozmiarze mniejszym od pewnej stałej $c \geq 1$ (tj. kiedy $r-p+1 < c$) nie była wykonywana procedura `Partition` i rekursywne wywołania, a tablice te pozostawały nieposortowane. Dopiero po zakończeniu tak „zepsutego" sortowania szybkiego, cała tablica ma być dodatkowo sortowana algorytmem sortowania „przez wstawianie". (2 pkt.)
- Porównaj czasy działania algorytmu podstawowego oraz jego modyfikacji dla dwóch rodzajów danych: dane losowe oraz dane skrajnie niekorzystne (ciąg malejący). Podobnie jak w zadaniu AL3.3, dla tego samego rodzaju danych należy zestawić czasy działania dwóch wersji algorytmów. (1 pkt)

Zadanie AL3.5 (5+2* pkt)

- Zaimplementuj algorytm sortowania szybkiego omówiony na wykładzie. (4 pkt.)
- Następnie zmodyfikuj ten algorytm tak, aby dla tablic o rozmiarze mniejszym od pewnej stałej $c \geq 1$ (tj. kiedy $r-p+1 < c$) nie była wykonywana procedura `Partition` i rekursywne wywołania, tylko stosowane było jakieś proste sortowanie, np. bąbelkowe czy przez wstawianie. (2 pkt.)
- Porównaj czasy działania algorytmu podstawowego oraz jego modyfikacji dla dwóch rodzajów danych: dane losowe oraz dane skrajnie niekorzystne (ciąg malejący). Podobnie jak w zadaniu AL3.3, dla tego samego rodzaju danych należy zestawić czasy działania dwóch wersji algorytmów. (1 pkt)