Programistyczne zadanie domowe nr II — Sortowanie

termin oddania: 21.04 (piątek, do końca dnia)

Celem tego zadania jest przećwiczenie praktycznej umiejętności konwersji pseudokodu na kod zapisany w rzeczywistym języku programowania. Państwa zadanie nie jest napisanie optymalnego algorytmu, a konwersja pseudokodu z podręcznika *Wprowadzenie do algorytmów* na program w wybranym języku.

Algorytmy do zaimplementowania

Poniżej znajduje się lista algorytmów do zaimplementowania. Zakładamy, że sortowaniu podlegają tablice liczb zmiennoprzecinkowych (double, do dwóch miejsc po przecinku). W nawiasach podano numery stron w polskiej wersji podręcznika wydanej przez PWN (zielona okładka). W innych wersjach podręcznika zastosowano inny podział na rozdziały, a interesujące nas algorytmy mogą się znajdować na innych stronach.

- Sortowanie przez wstawianie (str. 25),
- Sortowanie bąbelkowe (str. 39),
- Sortowanie kubełkowe (str. 198; zamiast list można zastosować zwykłe tablice/wektory/listy Pythona),
- Sortowanie przez scalanie (str. 34),
- Sortowanie szybkie (str. 169).

Wejście

Sposób obsługi wejścia nie zmienia się w stosunku do poprzedniego zadania. Zmianie uległy jedynie dane.
Po uruchomieniu programu przez narzędzie sprawdzające na standardowym wejściu pojawi się następujący ciąg znaków określający zawartość tablicy do posortowania (poszczególne elementy są oddzielone od siebie spacjami):

$$-1\ rozmiar\ element1\ element2\ element3\ ...$$

- $rozmiar$ — liczba elementów zapisanych w tablicy,
- $argument1\ argument2\ argument3\ ...$ — Kolejne elementy tablicy (liczby zmiennoprzecinkowe, typ double) przekazane w kolejności ich występowania

Przykład:

- $-1\ 3\ 2\ 3\ 4$ — Tablica $[2, 3, 4]$
- $-1\ 5\ 1\ 1\ 1\ 0\ 8$ — Tablica $[1, 1, 1, 0, 8]$ W tym zadaniu rozmiar tablicy pokrywa się z liczbą jej elementów.

Po wczytaniu danych, na standardowym wyjściu powinna wyświetlić się przekazana do programu tablica. Wartości w tablicy powinny być od siebie oddzielone przecinkami i spacjami (ten sposób wyświetlania to domyślny sposób prezentacji zawartości listy/tablicy przez funkcję *print* w Pythonie):

$$[element, element, None, element, ...]$$

Po wyświetleniu zawartości tablicy program przechodzi do trybu interaktywnego, w którym przyjmuje na standardowym wejściu następujące polecenia w postaci: $cyfra\ argument$.

Polecenia w trybie interaktywnym

Polecenia z poprzedniego zadania:

- $-1\ 0$ – Koniec działania programu. Na ekranie powinno się pojawić słowo *koniec*
- $0\ 0$ – wyświetlanie danych przekazanych na wejściu (czyli tablicy).

Polecenia do dodania:

- $1\ 0$ – Sortowanie tablicy algorytmem sortowania przez wstawianie,
- $2\ 0$ – Sortowanie tablicy algorytmem sortowania bąbelkowego,
- $3\ 0$ – Sortowanie tablicy algorytmem sortowania kubełkowego,
- $4\ 0$ – Sortowanie tablicy algorytmem sortowania przez scalanie,
- $5\ 0$ – Sortowanie tablicy algorytmem sortowania szybkiego.

Wyjście

Aby umożliwić sprawdzenie poprawności konwersji pseudokodu, w Państwa programie muszą się znaleźć dodatkowe instrukcje wyświetlające wynik działania fragmentów algorytmów. Wejściem każdego algorytmu jest tablica liczb. Wartości wszystkich dodatkowych parametrów muszą być obliczone przez kod na podstawie przekazanej do programu tablicy. Po zakończeniu sortowania na ekranie powinna pojawić się posortowana tablica.

Sortowanie przez wstawianie, sortowanie bąbelkowe

Program musi dodatkowo (poza tablicą wynikową) wyświetlić na ekranie zawartość sortowanej tablicy po każdym obiegu zewnętrznej pętli *for*.

Sortowanie kubełkowe

Przed wyświetleniem wynikowej tablicy, należy wyświetlić zawartość tablicy pomocniczej (po sortowaniu wewnętrznych list, w Cormenie jest to tablica B). Tablica pomocnicza powinna być wyświetlona jako „lista list” (tak jak robi to funkcja *print* w Pythonie). Na przykład tablica z rysunku 8.4 z Cormena powinna być wyświetlona jako

$$[[], [0.12, 0.17], [0.21, 0.23, 0.26], [0.39], [], [], [0.68], [0.72, 0.78], [], [0.94]]$$

Brak elementu na danej pozycji jest reprezentowany przez pustą listę ([]).

Sortowanie przez scalanie

Przed wyświetleniem wynikowej tablicy program wyświetla na ekranie liczbę wykonanych wywołań funkcji *mergeSort* oraz liczbę wywołań funkcji *merge*. Wartość te należy wyświetlić w jednej linii najpierw liczbę wywołań *mergeSort*, a następnie liczbę wywołań *merge*. Wartości te należy oddzielić od siebie spacją. W liczbie wywołań *mergeSort* należy uwzględnić również oryginalne wywołanie w celu posortowania tablicy.

Sortowanie szybkie

Przed wyświetleniem wynikowej tablicy program wyświetla na ekranie liczbę wykonanych wywołań funkcji *quickSort* oraz liczbę wywołań funkcji *partition*. Wartość te należy wyświetlić w jednej linii najpierw liczbę wywołań *quickSort*, a następnie liczbę wywołań *partition*. Wartości te należy oddzielić od siebie spacją. W liczbie wywołań *quickSort* należy uwzględnić również oryginalne wywołanie w celu posortowania tablicy. **W implementacji należy zastosować metodę partycjonowania Lomuty (metoda PARTITION z Cormena)**.

W przypadku tego zadania prace zostaną ocenione po terminie oddania. Aby umożliwić Państwu weryfikację rozwiązania (głownie sposobu wyświetlania wyników), udostępnione zostanie zadanie bez oceny, w którym znajdą się proste przypadki testowe umożliwiające sprawdzenie, czy program wyświetla dane we właściwy sposób. To, że program przechodzi to zadanie, nie oznacza automatycznie, że programowi uda się rozwiązać wszystkie przypadki testowe w zadaniu na ocenę (zadanie testowe nie będzie sprawdzało wszystkich algorytmów/skomplikowanych przypadków).

W razie wątpliwości proszę o kontakt mailowy lub poprzez MS Teams.