Programowanie proceduralne

Laboratorium 9-10 – Algorytmy sortowania

1. Zaimplementuj funkcję znajdującą największy element na liście. Funkcja powinna pobierać jako argument listę. Funkcja powinna zwracać wartość elementu największego.

   Aby zademonstrować działanie tej funkcji, napisz program, który wywołuje przygotowaną funkcję dla testowych danych (dane mogą być zapisane w programie lub mogą być podawane przez użytkownika).

2. Zaimplementuj funkcje działające na liście (np. zawierającej liczby całkowite):
   - wyznaczającą indeks największego elementu,
   - znajdującą najmniejszy element,
   - wyznaczającą indeks najmniejszego elementu.

   Zademonstruj działanie tych funkcji – napisz program, który wywołuje funkcje dla testowych danych.

3. Przeanalizuj działanie funkcji z zadań 1-2 wyznaczających indeks elementu maksymalnego (minimalnego). Jakie będą wyniki ich działania dla danych, w których jest więcej niż jeden element maksymalny (minimalny)? Jak możesz zmienić zachowanie tych funkcji?

4. Przygotuj plik sort.py (moduł) zawierający definicję funkcji z zadań 1 i 2. Napisz program – umieszczony w osobnym pliku, który wywołuje funkcje z modułu sort.

5. W pliku sort.py umieść i zdefiniuj funkcję **`swap`**, która w liście zamienia miejscami dwa elementy (o podanych indeksach).

6. Przeanalizuj algorytm sortowania przez wybór. Zauważ, że dysponujesz funkcjami:
   - znajdującą indeks elementu minimalnego w liście,
   - zamieniającą dwa elementy miejscami.

   Napisz funkcję **`selectsort`**, która porządkuje elementy listy, wykorzystując algorytm sortowania przez wybieranie. Zademonstruj działanie funkcji w programie.

7. Przeanalizuj algorytm sortowania bąbelkowego. Zaimplementuj funkcję **`bubblesort`**, która porządkuje elementy listy, wykorzystując ten algorytm. Zauważ, że dysponujesz funkcją zamieniającą miejscami dwa elementy tablicy. Zademonstruj działanie funkcji w programie.

8. Przeanalizuj algorytm sortowania przez scalanie. Zaimplementuj funkcję **`mergesort`**, która porządkuje elementy listy, wykorzystując ten algorytm. Zauważ, że przydatne będzie przygotowanie osobnej funkcji (**`merge`**) służącej do scalania dwóch posortowanych fragmentów listy. W funkcji **`merge`** konieczna będzie utworzenie kopii fragmentu listy. Zademonstruj działanie funkcji **`mergesort`** w programie.

9. Przeanalizuj algorytm sortowania szybkiego. Zaimplementuj funkcję **`quicksort`**, która porządkuje elementy listy, wykorzystując ten algorytm. Zauważ, że przydatne będzie przygotowanie osobnej funkcji (**`divide`**) służącej do podziału elementów listy na dwa fragmenty: elementów mniejszych oraz elementów większych/równych od wybranego elementu. W funkcji **`divide`** wykorzystaj funkcję **`swap`**. Zademonstruj działanie funkcji **`quicksort`** w programie.

Karol Tarnowski
Wrocław, 2021