Algorytmy i struktury danych

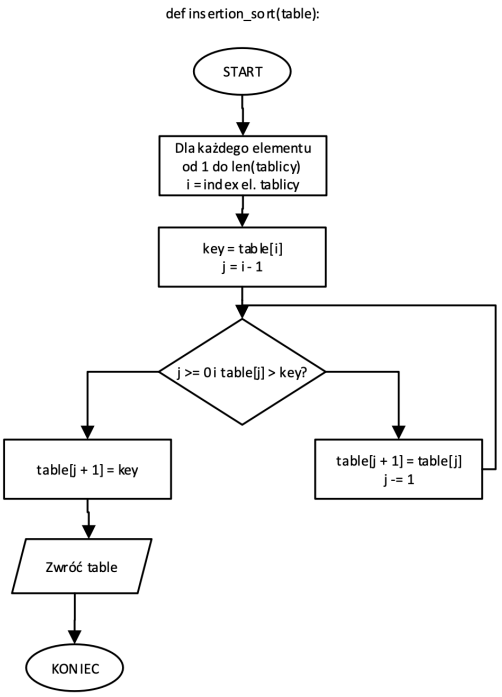
Sprawozdanie z Zadania nr: 7
Imię i Nazwisko: Bartosz Ochnik
Data: 27.12.2024

Wstawianie (insertsort)

Opis teoretyczny: Sortowanie przez wstawianie polega na iteracyjnym wstawianiu kolejnych elementów na właściwą pozycję w posortowanej części tablicy. W każdej iteracji bierze się jeden element z nieposortowanej części i przesuwa w posortowanej części tak, aby zachować porządek.

- **Złożoność czasowa**:
  - Średnia: $O(n^2)$
  - Najgorsza: $O(n^2)$
  - Najlepsza: O(n) (gdy dane są wstępnie posortowane)
- **Złożoność pamięciowa**: O(1) (sortowanie odbywa się w miejscu).
- **Stabilność**: Stabilne (nie zmienia kolejności równych elementów).

Opis schematem blokowym:



Przykłady wykorzystania:

- Sortowanie niewielkich zbiorów danych.
- Kiedy dane są wstępnie posortowane lub prawie posortowane.
- W systemach czasu rzeczywistego, gdzie prosta implementacja jest kluczowa.

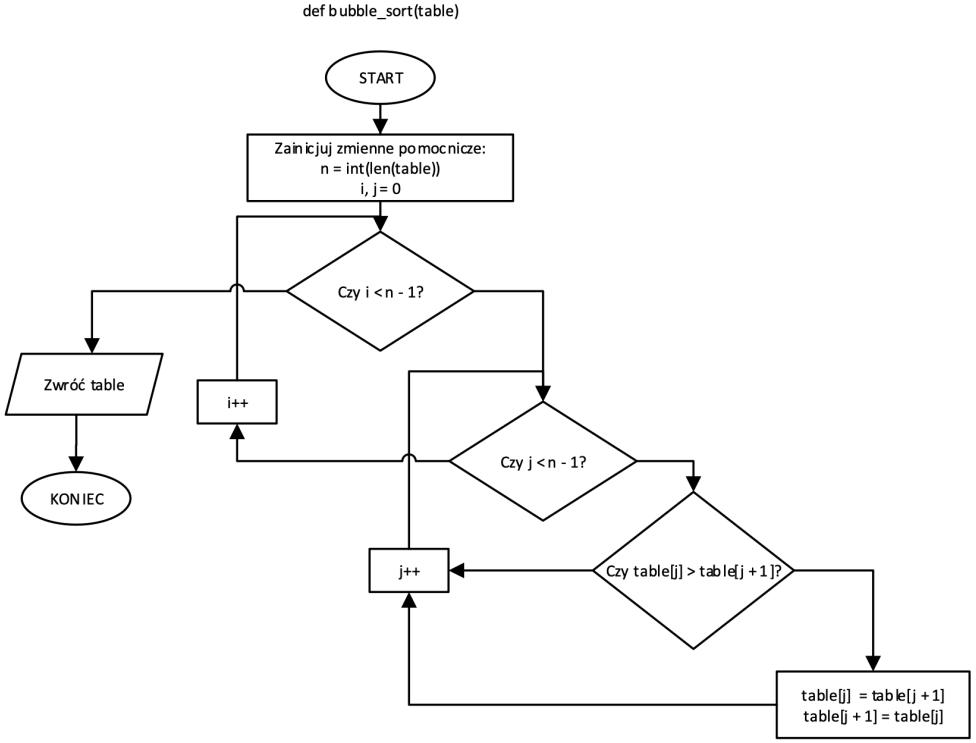
Algorytmy i struktury danych

Bąbelkowe (bubblesort)

Opis teoretyczny: Algorytm sortowania bąbelkowego polega na wielokrotnym porównywaniu sąsiednich elementów tablicy i zamianie ich miejscami, jeśli są w niewłaściwej kolejności. Proces powtarza się, aż tablica zostanie posortowana.

- **Złożoność czasowa**:
  - Średnia: $O(n^2)$
  - Najgorsza: $O(n^2)$
  - Najlepsza: O(n) (przy użyciu optymalizacji zatrzymującej sortowanie, gdy dane są już posortowane).
- **Złożoność pamięciowa**: O(1)
- **Stabilność**: Stabilne.

Opis schematem blokowym:



Przykłady wykorzystania:

- Edukacja (uczenie podstaw algorytmów sortowania).
- Niewielkie i proste zbiory danych, gdzie wydajność nie jest priorytetem.

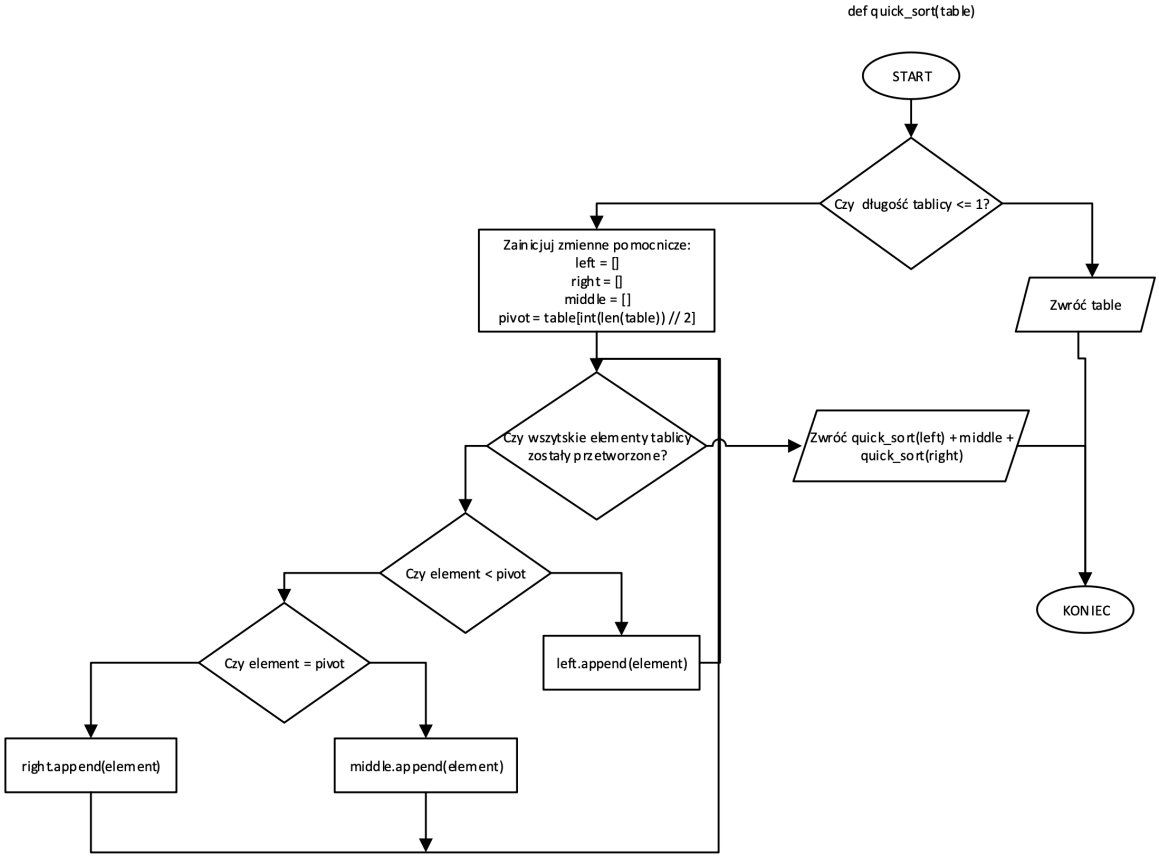
Algorytmy i struktury danych

Szybkie (quicksort)

Opis teoretyczny: Sortowanie szybkie to algorytm dziel i zwyciężaj. Wybierany jest element (tzw. pivot), względem którego reszta elementów jest podzielona na dwie grupy: mniejsze od pivota i większe od pivota. Następnie proces jest rekurencyjnie powtarzany dla obu grup.

- **Złożoność czasowa**:
  - Średnia: O(nlogn)
  - Najgorsza: O(n^2) (gdy pivot jest zawsze najmniejszym lub największym elementem).
  - Najlepsza: O(nlogn)
- **Złożoność pamięciowa**: O(logn) (dla optymalnego wyboru pivota).
- **Stabilność**: Niestabilne.

Opis schematem blokowym:



Przykłady wykorzystania:

- Sortowanie dużych zbiorów danych.
- W systemach, gdzie ważna jest złożoność średnia O(nlogn), np. bazy danych.
- W sortowaniu tablic w wielu językach programowania (np. C++ std::sort).

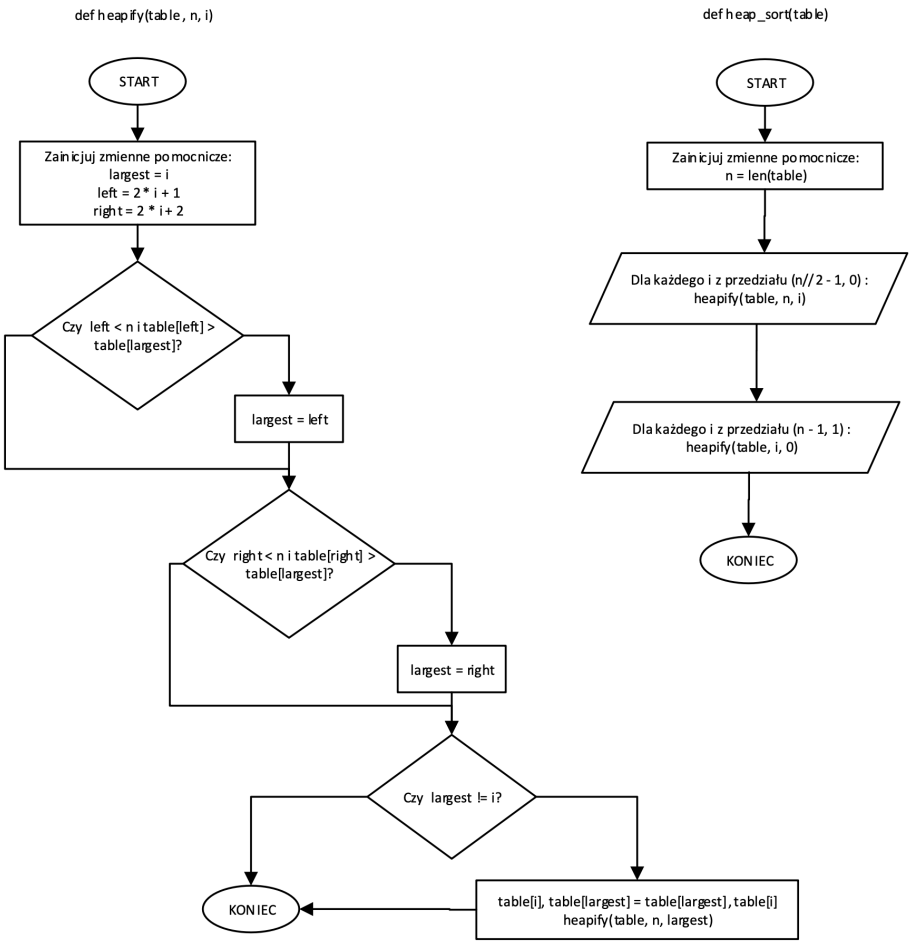
Algorytmy i struktury danych

Stogowe (heapsort)

Opis teoretyczny: Sortowanie stogowe wykorzystuje strukturę kopca (stogu). Algorytm tworzy kopiec maksymalny, gdzie największy element znajduje się na szczycie, a następnie usuwa ten element, przesuwając go na koniec tablicy. Proces jest powtarzany dla pozostałych elementów.

- **Złożoność czasowa**:
  - Średnia: O(nlogn)
  - Najgorsza: O(nlogn)
  - Najlepsza: O(nlogn)).
- **Złożoność pamięciowa**: O(1)
- **Stabilność**: Niestabilne.

Opis schematem blokowym:



Przykłady wykorzystania:

- Gdy potrzebne jest sortowanie w miejscu i stała złożoność pamięciowa.
- Algorytmy opierające się na priorytetach, np. kolejki priorytetowe.
- Sortowanie w systemach wbudowanych.

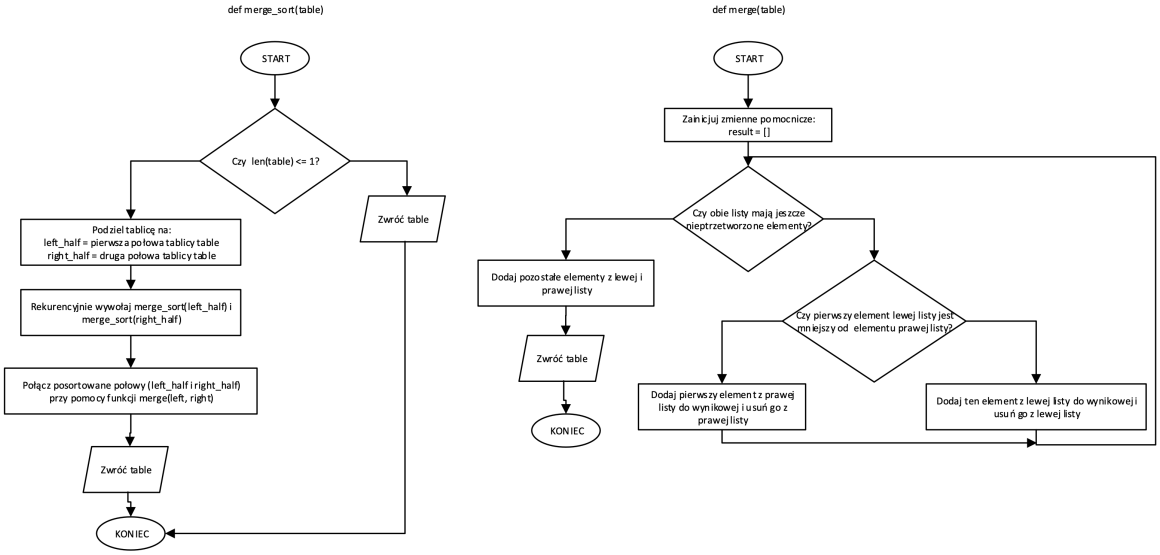
Algorytmy i struktury danych

Przez scalanie (mergesort)

Opis teoretyczny: Sortowanie przez scalanie to algorytm dziel i zwyciężaj. Dane są dzielone na mniejsze podzbiory, które są sortowane rekurencyjnie, a następnie scalane w jedną, posortowaną tablicę.

- **Złożoność czasowa**:
  - Średnia: O(nlogn)
  - Najgorsza: O(nlogn)
  - Najlepsza: O(nlogn).
- **Złożoność pamięciowa**: O(n) (ze względu na dodatkową pamięć na scalanie).
- **Stabilność**: Stabilne.

Opis schematem blokowym:



Przykłady wykorzystania:

- Sortowanie bardzo dużych zbiorów danych, szczególnie na dysku (algorytmy zewnętrzne).
- W systemach, gdzie stabilność sortowania jest kluczowa.
- Gdy dane muszą być przetwarzane w sposób równoległy (łatwe do zrównoleglenia).