Implementacja algorytmu Insertion Sort w podejściu TDD

Grzegorz Rozdzialik

23 maja 2021

Streszczenie

Sprawozdanie zadania 2. zawiera omówienie zaimplementowanego algorytmu Insertion Sort w języku Rust w podejściu TDD (ang. Test-Driven Development). Omówiony zostaje algorytm, utworzone testy automatyczne w kolejnych cyklach TDD, a także sposób uruchomienia testów.

Jest to rozwiązanie zadania 2. z przedmiotu *Testowanie i Weryfikacja Oprogramowania* studiach magisterskich OKNO 2020/2021.

Spis treści

Opis algorytmu Insertion Sort

Algorytm Insertion Sort ma na celu posortowanie listy elementów. Realizuje to poprzez podzielenie listy na 2 podlisty:

1. Lista elementów już posortowanych
2. Lista elementów oczekujących na posortowanie

A następnie wybieraniu kolejnych elementów z listy elementów oczekujących na posortowanie i umieszczaniu ich w odpowiednim miejscu w liście elementów posortowanych.

Początkowo lista elementów posortowanych zawiera wyłącznie pierwszy element listy, a pozostałe są uznawane jako nieposortowane.

Szerszy opis oraz przykład działania algorytmu został zaprezentowany na stronie internetowej https://www.tutorialspoint.com/data_structures_algorithms/insertion_sort_algorithm.htm

Stabilność algorytmu sortowania

W testach sprawdzana będzie również *stabilność* algorytmu sortowania.

Algorytm sortowania jest stabilny jeżeli elementy o tym samym kluczu sortowania są w tej samej kolejności względem siebie w danych wejściowych oraz w posortowanym wyniku.

Przykładowo, dla następujących par liczb `(x, y)`, gdzie `x` jest kluczem sortowania:

```
(0, 1), (-1, 1), (0, 5)
```

Posortowaniem stabilnym względem klucza `x` będzie jedynie:

```
(-1, 1), (0, 1), (0, 5)
```

Natomiast poniższy wynik nie będzie wynikiem stabilnego algorytmu sortowania:

```
(-1, 1), (0, 5), (0, 1)
```

Pomimo, że elementy są posortowane względem klucza `x` (pierwszego elementu z każdej pary), to elementy o kluczu 0 występują w wynikowej liście w odwrotnej kolejności niż w liście wejściowej.

Podejście Test-Driven Development

W podejściu Test-Driven Development (TDD) implementacja algorytmu lub funkcjonalności następuje w kolejno następujących po sobie cyklach składających się z następujących kroków:

1. Napisanie testu dla niezrealizowanej funkcjonalności. Wykonanie tego testu powinno zakończyć się porażką.
2. Implementacja funkcjonalności. Po zakończeniu implementacji nowo napisany test, jak i testy napisane w poprzednich cyklach powinny zakończyć się powodzeniem.
3. Czyszczenie, ulepszenie implementacji (ang. refactoring). Po tym kroku wszystkie testy nadal powinny kończyć się powodzeniem.

Implementacja algorytmu z zadania

Algorytm z zadania zaimplementowano w języku Rust (https://www.rust-lang.org/).

Zaimplementowana funkcja obsługuje nie tylko liczby typu `i32` (jak było to w pierwszym zadaniu), ale dowolne elementy, które można ze sobą porównywać. W praktyce oznacza to, że funkcja `insertion_sort` przyjmuje wektor elementów, które implementują cechę (ang. trait) `Ord`, która pozwala je ze sobą porównać.

Ponadto, zaimplementowana funkcja wykonuje sortowanie stabilne, bo zostało potwierdzone w jednym z testów.

Stworzony projekt jest biblioteką (ang. library) i nie dostarcza aplikacji wykonywalnej (ang. executable binary). Udostępniona została jedynie funkcja `insertion_sort`, która wykonuje sortowanie. Użytkownicy tej biblioteki mogą użyć ją do sortowania w swoich projektach w języku Rust (przykładowym zastosowaniem byłoby stworzenie programu będącego odpowiednikiem komendy `sort` znanej z systemów UNIX).

Do uruchomienia testów użyto standardowej komendy dla języka Rust:

```
cargo test
```

Komenda ta jest dostępna po instalacji narzędzi dla języka Rust: https://www.rust-lang.org/tools/install.

Kod źródłowy projektu został udostępniony w serwisie GitHub: https://github.com/Gelio/tiwo-sorting-tdd. Korzystając z historii zmian można przejrzeć kolejne cykle bezpośrednio w serwisie (nazwy zmian zaczynają się od `(cycle X)`, gdzie `X` jest liczbą całkowitą), a także pobrać repozytorium i uruchomić testy we własnym środowisku.

Interfejs algorytmu sortowania

Cykle TDD dla implementacji algorytmu

Implementacja funkcji `insertion_sort`

Wynik wykonania testów

Bibliografia

1. Opis algorytmu Insertion Sort
   https://www.tutorialspoint.com/data_structures_algorithms/insertion_sort_algorithm.htm

2. Slajdy do przedmiotu *Testowanie i Weryfikacja Oprogramowania* na studiach magisterskich OKNO
3. Dokumentacja pisania testów w języku Rust https://doc.rust-lang.org/book/ch11-01-writing-tests.html
4. *Clean Code - Uncle Bob / Lesson 4* (sekcja o TDD rozpoczyna się około 21:41) https://youtu.be/58jGpV2Cg50?t=1300
5. *Klasyfikacja algorytmów sortowania* - Wikipedia https://pl.wikipedia.org/wiki/Sortowanie#Klasyfikacja