AISDI - ćwiczenie 3

9 maja 2026

1 Sprawy organizacyjne

1. Ćwiczenie realizowane jest samodzielnie.
2. Ćwiczenie wykonywane jest w języku Python.
3. Ćwiczenie powinno zostać oddane najpóźniej do 10.06.2026. W ramach oddawania ćwiczenia należy zademonstrować prowadzącemu działanie kodu oraz utworzyć pull request (z kodem oraz raportem) który prowadzący będzie mógł komentować.
4. Rozwiązanie powinno wykorzystywać optymalne dla tego problemu struktury danych oraz algorytmy.
5. Kod oraz raport powinien być umieszczony na repozytorium `gitlab-stud.elka.pw.edu.pl`, a pull request utworzony do konta prowadzącego: **@djagodzi**
6. Raport powinien być w postaci pliku .pdf, .html albo być częścią notebooka jupyterowego. Powinien zawierać koncepcję rozwiązania i testowania, opis eksperymentów, uzyskane wyniki wraz z komentarzem oraz wnioski.
7. Na ocenę wpływa poprawność oraz jakość kodu i raportu.
8. Implementacja algorytmu powinna być ogólna.

2 Maszyna Turinga

2.1 Wprowadzenie

Funkcję przejścia dla maszyny Turinga można zapisać w postaci tekstu złożonego z wierszy o postaci:

current_state current_symbol new_symbol direction new_state gdzie:

- current_state jest identyfikatorem (ciąg liter) aktualnego stanu maszyny;
- current_symbol jest symbolem odczytywanym przez głowicę maszyny (może to być dowolna litera, cyfra, albo znak _ traktowany jako symbol pusty maszyny);
- new_symbol jest symbolem zapisywanym przez głowicę maszyny (jak wyżej);
- direction jest literą L,R lub znakiem *, co oznacza odpowiednio przesunięcie głowicy w lewo, w prawo lub brak przesunięcia;
- new_state jest identyfikatorem kolejnego stanu maszyny.

Początkowo maszyna jest w stanie init, a stanem końcowym maszyny jest dowolny stan, którego identyfikator zaczyna się od halt (np. halt, haltOK). Wielkość liter w identyfikatorach stanów oraz odczytywanych i zapisywanych przez głowicę ma znaczenie.

2.1.1 Przykład 1

Załóżmy, że zawartość taśmy i położenie głowicy (zaznaczone symbolem ^) są następujące (reszta taśmy wypełniona jest symbolem _):

```
1011
^
```

Funkcja przejścia dodająca jeden do liczby binarnej mogłaby mieć postać:

```
init 1 1 R init
init 0 0 R init
init _ _ L carry
carry 1 0 L carry
carry 0 1 * halt
carry _ 1 * halt
```

To znaczy, w stanie `init` głowica przesuwa się w prawo na sam koniec liczby, by potem w stanie `carry` przenieść jedynkę w odpowiednie miejsce. Zapisując jedynkę maszyna przechodzi w stan `halt`.

2.1.2 Przykład 2

Załóżmy, że zawartość taśmy i położenie głowicy (zaznaczone symbolem ^) są następujące (reszta taśmy wypełniona jest symbolem _):

```
101101
^
```

Funkcja przejścia sprawdzająca czy liczba binarna kończy się na `01` mogłaby mieć postać:

```
init 1 1 R init
init 0 0 R init
init 0 0 R zero
zero 1 1 R one
one _ _ * halt
```

To znaczy, w stanie `init` głowica przesuwa się w prawo nad znakami liczby, aby za każdym razem, gdy widzi znak `0` sprawdzić, czy nie rozpoczyna on końcowego ciągu `01`. Zauważmy, że widząc znak `0` maszyna może zarówno pozostać w stanie `init`, jak i zmienić stan na `zero`.

Uwaga: Przyjmujemy, że maszyna kończy pracę, gdy którekolwiek z możliwych wykonań doprowadzi do stanu końcowego.

2.2 Ćwiczenie

Zadanie polega na napisaniu programu, który:

1. wczyta zawartość taśmy oraz funkcję przejścia (zawartość taśmy i nazwa pliku z funkcją przejścia przekazane jako argumenty wywołania programu);

2. wyświetli zawartość taśmy, położenie głowicy (głowica jest początkowo umieszczona nad pierwszym symbolem) i identyfikator stanu (init);

3. znajdzie i wyświetli sekwencję zmian zawartości taśmy, położenia głowicy i stanu zgodnie z wczytaną funkcją przejścia, prowadzącą do stanu końcowego.

Rezultatem powinny być kod źródłowy programu oraz 3 pliki tekstowe z przykładowymi funkcjami przejścia dla maszyny Turinga (innymi niż z treści zadania).

Przykładowo, zakładając że pliki: `przyklad_1.txt` i `przyklad_2.txt` zawierają pokazane wcześniej funkcje przejścia, wynikiem działania programu powinno być:

```
$ python program.py 1011 przyklad_1.txt
1011 init
^
1011 init
 ^
1011 init
  ^
1011 init
   ^
1011_ init
    ^
1011 carry
   ^
1010 carry
  ^
1000 carry
 ^
1100 halt
 ^
```

oraz:

```
$ python program.py 101101 przyklad_2.txt
101101 init
^
101101 init
 ^
101101 init
  ^
101101 init
   ^
101101 init
    ^
101101 zero
     ^
101101_ one
      ^
101101_ halt
      ^
```