Zadanie Robot – miniLOGIA 16 (2017/18), etap 3

Treść zadania

Robot porusza się od lewego do prawego końca korytarza. Został tak zaprogramowany, że wykonuje tylko dwie podstawowe czynności: idzie naprzód o 1 krok lub zawraca, gdy nie może pójść naprzód. Na początku robot stoi w lewym końcu korytarza, jest skierowany w kierunku prawego i jest naładowany.

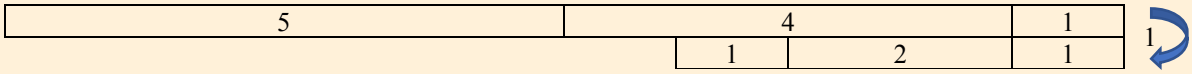
Po uruchomieniu porusza się do drugiego końca, a potem zawraca. Wykonując czynności (ruch lub zawracanie) traci energię, więc co jakiś czas musi zostać doładowany. Doładowanie nie jest jednak pełne, każde kolejne umożliwia wykonanie o jedną czynność mniej niż poprzednie.

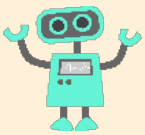
Napisz dwuparametrową funkcję **robot**. Pierwszy parametr określa liczbę czynności, jaką robot może wykonać na pełnym naładowaniu, drugi długość korytarza wyrażoną w krokach robota. Oba parametry są liczbami naturalnymi od 1 do 1000. Wynikiem funkcji jest odległość robota od lewego końca po wykonaniu wszystkich możliwych czynności.

Przykład 1:

Dla wartości pierwszego naładowania równej 5 i długości korytarza 10.

Robot po pierwszym naładowaniu idzie 5 kroków, po drugim naładowaniu 4 kroki, po trzecim 1 krok, zawraca i idzie 1 krok, po czwartym naładowaniu 2 kroki, a po ostatnim 1 krok. Odległość od lewego końca wynosi **6**.

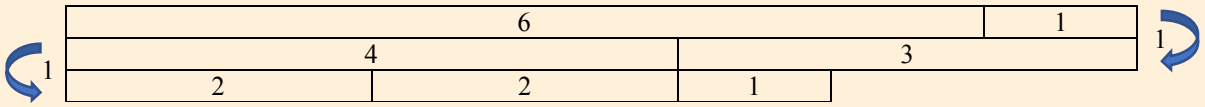




Przykład 2:

Dla wartości pierwszego naładowania równej 6 i długości korytarza 7.

Robot po pierwszym naładowaniu idzie 6 kroków, po drugim naładowaniu 1 krok, zawraca, a potem idzie 3 kroki, po trzecim naładowaniu idzie 4 kroki, po czwartym zawraca i idzie 2 kroki, po piątym idzie 2 kroki, a po ostatnim 1. Odległość od lewego końca wynosi **5**.



Logo:

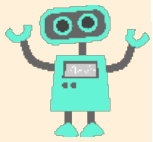
wynikiem **robot 5 10** jest **6**,

wynikiem **robot 6 7** jest **5**.

Python:

wynikiem **robot(5,10)** jest **6**,

wynikiem **robot(6,7)** jest **5**.



Omówienie rozwiązania

Zadanie można rozwiązać implementując symulację ruchu robota. Zapamiętujemy bieżące położenie robota i rozpatrujemy krok po kroku jego pozycję. Jest to rozwiązanie wynikające wprost z treści zadania, ale pracochłonne do implementacji.

Można jednak zauważyć pewną zależność. Jeśli pełne naładowanie wynosi 6, to kolejne wartości naładowania wynoszą: 6, 5, 4, 3, 2, 1. Można liczby zapisać w innej kolejności 6, 1, 5, 2, 4, 3. Teraz łatwo zauważyć, że sumując liczby parami, otrzymamy takie same wartości 7, a średnio (n+1)/2. Takich liczb jest n. Wobec tego wzór na łączną energię można zapisać n*(n+1)/2. Podobnie będzie w przypadku nieparzystej wartości pełnego naładowania: 5, 4, 3, 2, 1; średnio otrzymujemy (5+1)/2=3. Jeśli długość korytarza wynosi dl, to pełny cykl – ruch tam i z powrotem z obrotami – wynosi 2*dl+2.

Obliczamy położenie robota ze wzoru:

energia = n*(n+1)/2, gdzie n - początkowe naładowanie
polozenie = *energia* **mod** (2*dl+2), gdzie dl – długość korytarza, a **mod** oznacza resztę z dzielenia.

i rozpatrujemy cztery przypadki:

- robot idzie z lewej do prawej
  Jeśli `polozenie < dl + 1`, wynikiem jest wartość `polozenie`
- robot jest na prawym końcu
  Jeśli `polozenie = dl + 1`, wynikiem jest wartość `polozenie - 1`
- robot wraca – idzie z prawej do lewej
  Jeśli `polozenie > dl + 1` i `polozenie < 2 * dl + 2`,
  wynikiem jest `wartość 2 * dl + 1 - polozenie`
- robot jest w położeniu początkowym (lub po pełnym cyklu)
  Jeśli `polozenie = 2 * dl + 2`, wynikiem jest wartość `0`

Rozwiązanie w języku Python

Warunki zapisane w języku Python są w wersji uproszczonej w stosunku do zapisu w języku naturalnym, gdyż polecenie **return** nie tylko przekazuje wynik, ale kończy działanie funkcji.

```
1.  def robot(n, dl):
2.      polozenie = (n* (n + 1) // 2  ) % (2 * dl + 2)
3.
4.      if polozenie < dl + 1:
5.          return polozenie
6.
7.      if polozenie == dl + 1:
8.          return polozenie - 1
9.
10.     if polozenie < 2 * dl + 2:
11.         return 2 * dl + 1 - polozenie
12.
13.     return 0
```

Rozwiązanie w języku Logo

Warunki zapisane w języku Logo są w wersji uproszczonej w stosunku do zapisu w języku naturalnym, gdyż polecenie **wynik** nie tylko przekazuje wynik, ale kończy działanie funkcji.

```
1.  oto robot :n :dl
2.    niech "polozenie  reszta ((:n + 1) * :n / 2  )  (2 * :dl + 2)
3.
4.    jeśli :polozenie < :dl + 1
5.      [wy :polozenie]
6.
7.    jeśli :polozenie = :dl + 1
8.      [wy :polozenie - 1]
9.
10.   jeśli :polozenie < 2 * :dl + 2
11.     [wy 2 * :dl + 1 - :polozenie]
12.
13.   wy 0
14. już
```

Testy

Najpierw rozwiązanie sprawdzamy dla najprostszych scenariuszy, gdy robot idzie tylko do przodu i nie zawraca, potem dla bardziej skomplikowanych. W testach nie powinno zabraknąć przykładów dla wszystkich przypadków: robot jest w drodze z lewej do prawej, na końcu, w drodze z prawej do lewej i na samym początku. Sprawdzamy działanie funkcji dla małych i dużych wartości ze szczególnym uwzględnieniem wartości brzegowych.

Python	Logo	wynik
robot(42, 100) robot(36, 45)	robot 42 100 robot 36 45	95 22
robot(60, 35) robot(700, 500)	robot 60 35 robot 700 500	30 139
robot(34, 5) robot(423, 7)	robot 34 5 robot 423 7	4 3
robot(42, 6) robot(404, 100)	robot 42 6 robot 404 100	6 0
robot(500, 1000) robot(1000, 1000)	robot 500 1000 robot 1000 1000	875 0