Zagadka Wież Hanoi w języku Java

Zagadka Wież Hanoi w języku Java

Źródło: Thuan Pham, domena publiczna.

Zagadka Wież Hanoi jest problemem, który polega na przeniesieniu pojedynczo pewnej liczby krążków o różnych średnicach z jednego słupka na drugi i ułożeniu ich w kolejności od największego do najmniejszego. Łamigłówka ta oparta jest na starej legendzie o buddyjskich mnichach, którzy mają przenieść w ten sposób 64 krążki. W celu rozwiązania problemu muszą wykonać aż $2^{64} - 1$ ruchów. W e-materiale Zagadka Wież Hanoi przedstawiliśmy najważniejsze informacje dotyczące tego zagadnienia.

W tym e-materiale poznasz implementację algorytmu rozwiązywania tego problemu w języku Java.

Więcej przykładów, ćwiczeń i rozwiązań znajdziesz w e-materiale Zagadka Wież Hanoi – zadania maturalne.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Zagadka Wież Hanoi w języku C++,
- Zagadka Wież Hanoi w języku Python.

O tym, jak zagadnienie rekurencji wyjaśnia matematyka, przeczytasz w e-materiałach:

- Ciąg określony rekurencyjnie,
- Ciąg geometryczny określony rekurencyjnie,
- Wzór ogólny ciągu określonego rekurencyjnie,
- Ciąg arytmetyczny określony wzorem rekurencyjnym.

Twoje cele

- Zaimplementujesz algorytm rozwiązania zagadki Wież Hanoi w języku Java.
- Przeanalizujesz złożoność czasową i pamięciową omawianego algorytmu.
- Rozwiążesz samodzielnie zadania związane z zagadką Wież Hanoi.

Przeczytaj

Polecenie 1

Zapisz program wykorzystujący rekurencję, który wypisze kolejne kroki, jakie trzeba wykonać, aby przenieść wieżę składającą się z n krążków o różnej średnicy ze słupka `A` na słupek `C`. Swój program przetestuj dla 5 krążków.

Zasady gry:

- Gra składa się z trzech słupków i zestawu krążków o różnych średnicach.
- Na jednym słupku umieszczona jest wybudowana wieża.
- Celem gry jest odbudowanie wieży na trzecim słupku.
- Nie wolno położyć krążka o większej średnicy na krążku o mniejszej średnicy.
- Na raz można przenosić tylko jeden krążek (znajdujący się na szczycie wieży).
- Do przenoszenia można wykorzystać dodatkowy słupek.

Specyfikacja problemu:

Dane:

- n – liczba krążków w grze, liczba naturalna
- `A`, `B`, `C` – symboliczne nazwy słupków, znaki

Wynik:

Program, na wyjściu standardowym, wypisuje w kolejnych linijkach kroki, które prowadzą do rozwiązania zagadki Wież Hanoi.

```
1 public class Main {
2     public static void main(String[] args){
3         // Tutaj dodaj własny kod. Użyj funkcji
```

```
        // System.out.print();
    }
}
```

```
```

Polecenie 2

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/R1bVBYyDekoQo

Film nawiązujący do treści materiału zagadki wież Hanoi.

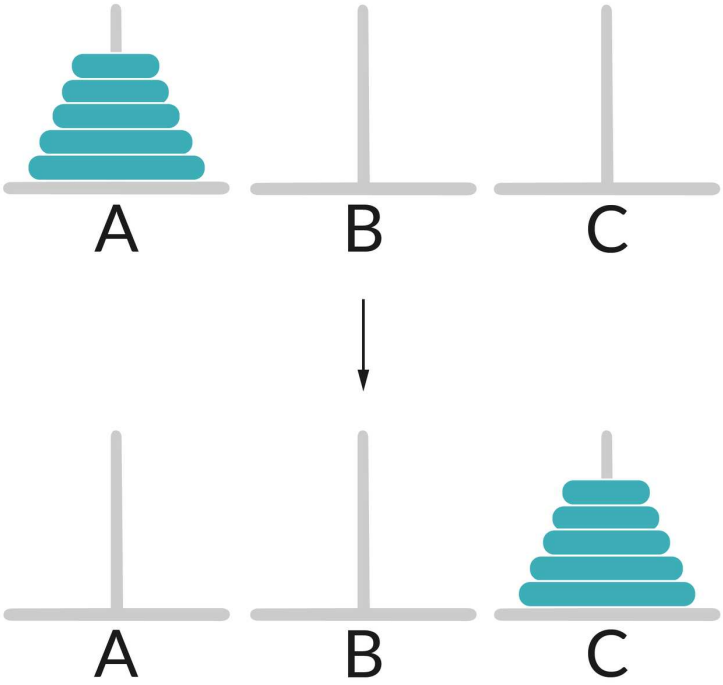
Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 1.36 KB w języku polskim

Przedstawienie problemu

Wiesz już, że zagadka Wież Hanoi polega na przeniesieniu krążków o różnych rozmiarach z jednego słupka na drugi. Warunkiem poprawnego rozwiązania zadania jest pojedyncze przenoszenie każdego krążka, a także brak możliwości położenia większego krążka na mniejszym. Do dyspozycji mamy jednak pomocniczy słupek, na który możemy tymczasowo odkładać krążki.

Mamy zatem trzy słupki: `A`, `B`, `C` oraz `n` krążków. Początkowo wszystkie krążki położone są na słupku `A`, w kolejności od największego do najmniejszego. Docelowo wszystkie krążki mają znaleźć się na słupku `C` w tej samej kolejności, w jakiej znajdowały się na słupku A. Pomocniczym słupkiem jest słupek `B`. Celem naszego zadania jest przedstawienie ciągu ruchów, opisującego rozwiązanie zagadki Wież Hanoi.



Źródło: Contentplus.pl sp. z o.o., licencja: CC BY-SA 3.0.

Implementacja rozwiązania w języku Java

Problem Wież Hanoi można rozwiązać w sposób iteracyjny, czyli za pomocą pętli. Jest on jednak bardzo dobrym przykładem sytuacji, gdzie możemy zastosować rozwiązanie rekurencyjne, czyli takie, w którym używamy metody „dziel i zwyciężaj". Operacje niezbędne do umieszczenia coraz mniejszych krążków na słupku docelowym (`C`) są takie same jak w przypadku pierwszego, największego krążka – z tym zastrzeżeniem, że zmienia się słupek pomocniczy. Najpierw (czyli po przeniesieniu największego krążka na słupek `C`) będzie nim `A`. Później rolę słupka pomocniczego będą na przemian odgrywać słupki `B` i `A`. Zauważmy, że słupek docelowy się nie zmienia.

Przypadkiem rekurencyjnym będzie sytuacja, w której na słupku `A` pozostał co najmniej jeden krążek. Instrukcje składające się na funkcję rekurencyjną `hanoi()` będą wówczas wykonywane. Przypadkiem bazowym będzie sytuacja, w której na słupku A nie ma już żadnego krążka. Aktualne wywołanie funkcji `hanoi()` przestanie być wtedy wykonywane.

> **Ważne!**
>
> Należy pamiętać, aby w funkcji rekurencyjnej zawsze opisać przypadek bazowy, który określi, kiedy funkcja powinna się zatrzymać. Bez zdefiniowanego przypadku bazowego funkcja teoretycznie będzie wywoływać samą siebie w nieskończoność. W praktyce nastąpi moment, w którym zostaną wyczerpane zasoby przydzielone programowi, co zakończy się jego awarią. Taką sytuację nazywamy przepełnieniem stosu.
>
> **Przepełnienie stosu** (ang. *Stack overflow*) polega na wyczerpaniu miejsca zarezerwowanego na stos. Spowodowane jest ono właśnie nadmierną bądź nieskończoną rekurencją. Kiedy używamy języka Java, mamy domyślnie do dyspozycji stos o rozmiarze

> 512 KB, a przekroczenie tej wartości sygnalizowane jest zgłoszeniem błędu `StackOverflowError`.

Oto implementacja algorytmu w języku Java dla 5 krążków:

```java
public class Main {
    // n – liczba krążków, A – miejsce początkowe
    // B – słupek pomocniczy, C – miejsce docelowe
    public static void hanoi(int n, char A, char B, char C) {
        if (n > 0) {
            //wywołanie rekurencyjne
            hanoi(n - 1, A, C, B);
            System.out.println("Przeniesienie z " + A + " do " +
            //wywołanie rekurencyjne
            hanoi(n - 1, B, A, C);
        }
    }

    public static void main(String[] args) {
        hanoi(5, 'A', 'B', 'C');
    }
}
```

Po uruchomieniu programu otrzymamy następujący wynik:

```
Przeniesienie z A do C
Przeniesienie z A do B
Przeniesienie z C do B
Przeniesienie z A do C
Przeniesienie z B do A
Przeniesienie z B do C
Przeniesienie z A do C
Przeniesienie z A do B
Przeniesienie z C do B
Przeniesienie z C do A
Przeniesienie z B do A
Przeniesienie z C do B
Przeniesienie z A do C
Przeniesienie z A do B
Przeniesienie z C do B
Przeniesienie z A do C
```

```
Przeniesienie z B do A
Przeniesienie z B do C
Przeniesienie z A do C
Przeniesienie z B do A
Przeniesienie z C do B
Przeniesienie z C do A
Przeniesienie z B do A
Przeniesienie z B do C
Przeniesienie z A do C
Przeniesienie z A do B
Przeniesienie z C do B
Przeniesienie z A do C
Przeniesienie z B do A
Przeniesienie z B do C
Przeniesienie z A do C
```

Jest to gotowe rozwiązanie problemu Wież Hanoi dla 5 krążków.

Złożoność czasowa i pamięciowa

Jak możemy zauważyć, do rozwiązania problemu Wież Hanoi dla 5 krążków potrzebne było 31 ruchów. Gdybyśmy uruchomili ten program dla 10 krążków, potrzebne byłyby aż 1023 ruchy.

Aby rozwiązać problem Wież Hanoi dla n krążków, należy przenieść jeden krążek oraz rozwiązać dwukrotnie ten problem dla $n-1$. Oznacza to, że liczba wykonanych przeniesień jest równa $2^n - 1$. Algorytm ma zatem złożoność wykładniczą $O(2^n)$.

Złożoność pamięciowa tego algorytmu jest liniowa, czyli równa $O(n)$. Wynika to z faktu, że funkcje wywołane w jednej funkcji rekurencyjnej nie wywołają się jednocześnie — drugie wywołanie pojedynczej funkcji będzie miało miejsce dopiero po zakończeniu całości pierwszego wywołania. Zatem pamięć ta może być użyta ponownie. W tym samym momencie aktywnych będzie maksymalnie $O(n)$ funkcji.

Słownik

> **funkcja rekurencyjna**
>
> funkcja, która wywołuje samą siebie, aż do momentu osiągnięcia stanu początkowego
> **myślenie rekurencyjne**
>
> myślenie polegające na tym, że wartość pewnej wielkości obliczana jest poprzez odwołanie się do tej samej wielkości określonej dla mniejszych wartości parametrów

przypadek bazowy

warunek w funkcji rekurencyjnej, po którego spełnieniu funkcja nie wywołuje samej siebie po raz kolejny

przypadek rekurencyjny

warunek w funkcji rekurencyjnej, po którego spełnieniu funkcja ponownie wywołuje siebie samą

zagadka Wież Hanoi

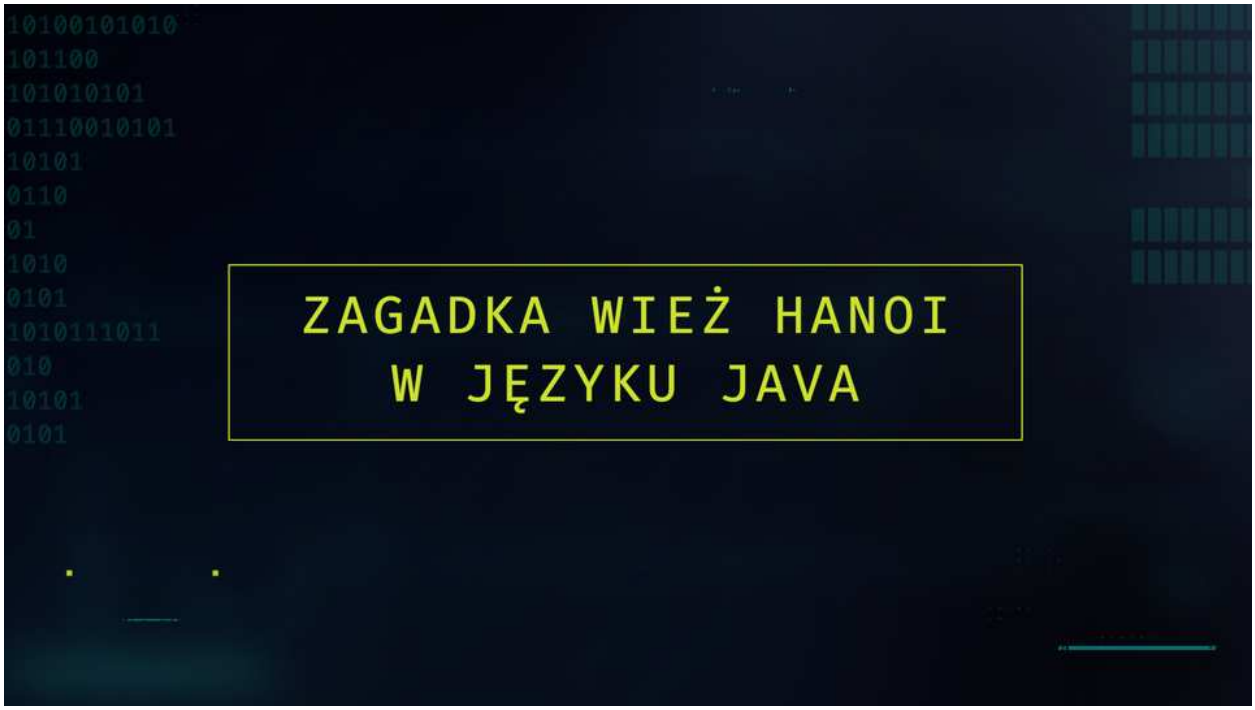
problem polegający na przeniesieniu wieży z krążków o różnej średnicy z jednego słupka na drugi, przy czym można przenosić tylko po jednym krążku z wierzchu stosu; nie jest dozwolone położenie większego krążka na mniejszym

złożoność wykładnicza

złożoność, która jest określona za pomocą funkcji wykładniczej, rośnie wykładniczo w zależności od rozmiaru danych wejściowych n i można ją zapisać jako $O(k^n)$

Animacja

> **Polecenie 1**
>
> Zapoznaj się z animacją, dotyczącą implementacji rozwiązania zagadki Wież Hanoi w języku Java.



Film dostępny pod adresem /preview/resource/RPkqPu3Fo4VAH

Film nawiązujący do treści materiału zagadka Wież Hanoi w języku Java.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 1.36 KB w języku polskim

> **Polecenie 2**
>
> Poszukaj informacji na temat mocy obliczeniowej najszybszego superkomputera. Ile czasu zajęłoby mu rozwiązanie zagadki Wież Hanoi dla 64 krążków?

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Uzupełnij podaną funkcję w taki sposób, aby rozwiązywała ona problem Wież Hanoi za pomocą rekurencji. Dodaj również zapisywanie, ile krążków jest na poszczególnych słupkach w trakcie każdego wykonania funkcji. Krążki początkowo znajdują się na słupku `A` i mają docelowo zostać przeniesione na słupek `C`.

Przykład: jeżeli krążek zdejmujemy ze słupka `A` i umieszczamy na słupku `C`, wówczas zmienną `krazkiNaA` powinniśmy zmniejszyć o 1, a `krazkiNaC` zwiększyć o 1.

Swoje rozwiązanie przetestuj dla 5 krążków: `n = 5`.

Specyfikacja problemu:

Dane:

- n – liczba krążków w grze, liczba naturalna

Wynik:

Program, na standardowym wyjściu, w kolejnych liniach wypisuje, ile krążków jest na poszczególnych słupkach w każdym kroku wykonania funkcji. Wartości na kolejnych słupkach powinny zostać oddzielone pojedynczym znakiem odstępu.

Twoje zadania

1. Program oblicza, ile krążków jest na poszczególnych słupkach na każdym etapie wykonania funkcji. Liczby krążków powinny zostać wypisane w oddzielnych linijkach, przed dokonaniem ruchu.

```
1 public class Main {
2     public static int n = 5;
3     public static int krazkiNaA = n;
4     public static int krazkiNaB = 0;
5     public static int krazkiNaC = 0;
6     public static void hanoi(int n, char A, char B, char C) {
7                 //tutaj wpisz swój kod
8     }
```

```
9
10      public static void main(String[] args) {
11          hanoi(countA, 'A', 'B', 'C');
12      }
13  }
```

```
1
```

Ćwiczenie 2

Uzupełnij podany kod tak, aby rozwiązywał zagadkę Wież Hanoi. Dodatkowo program powinien wyświetlać liczbę przesunięć krążków potrzebnych do rozwiązania tej zagadki dla n krążków. Swoje rozwiązanie przetestuj dla 11 krążków.

Specyfikacja problemu:

Dane:

- n – liczba krążków, liczba naturalna

Wynik:

Program, na standardowym wyjściu, wypisuje liczbę przesunięć krążków potrzebnych do rozwiązania zagadki Wież Hanoi.

Twoje zadania

1. Program rozwiązuje zagadkę Wież Hanoi i wypisuje liczbę przesunięć krążków potrzebnych do wykonania tego zadania.

```java
public class Main
{
    static int licznik = 0;

    public static void hanoi(int n, char A, char B, char C)
    {
        // Tutaj wpisz kod
    }

    public static void main(String[] args)
    {
        int n = 11;
        hanoi(n, 'A', 'B', 'C');
        System.out.println(licznik);
```

```
1
```

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Zagadka Wież Hanoi w języku Java

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

b) rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),

c) metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Zaimplementujesz algorytm rozwiązania zagadki Wież Hanoi w języku Java.
- Przeanalizujesz złożoność czasową i pamięciową omawianego algorytmu.
- Rozwiążesz samodzielnie zadania związane z zagadką Wież Hanoi.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Zagadka Wież Hanoi w języku Java". Nauczyciel prosi uczniów o zapoznanie się z multimedium w sekcji „Animacja".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. Uczniowie w formie burzy mózgów wymieniają najważniejsze informacje, które zapamiętali z animacji. W razie potrzeby nauczyciel dopowiada brakujące kwestie.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj" wyświetlone na tablicy.
2. **Praca z multimedium.** Uczniowie zapoznają się z treścią polecenia 1 z sekcji „Przeczytaj". Pracując w parach, tworzą program. Następnie porównują swoje rozwiązania z przedstawionym w filmie.
3. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Na koniec zajęć z programowania w Javie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują ćwiczenie 2 z sekcji „Sprawdź się".
2. Uczniowie wykonują polecenie 2 z sekcji „Animacja".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.
- Nauczyciel może wykorzystać film samouczek z sekcji „Przeczytaj" do pokazania zastosowań list w języku Java i omówienia operacji na listach.