Informatyka w Edukacji, XVI
UMK Toruń, 2019

WYSZUKIWANIE, MASTERMIND I SZTUCZNA INTELIGENCJA

Agnieszka Borowiecka, Katarzyna Olędzka
Ośrodek Edukacji Informatycznej i Zastosowań Komputerów
w Warszawie, Raszyńska 8/10
{agnieszka.borowiecka, katarzyna.oledzka}@oeiizk.waw.pl

Abstract. Learning programming helps us to better understand the world around us. While teaching searching algorithms in an unstructured and structured set, we can choose different programming languages.The Mastermind game is an example of appliance a genetic algorithm – nondeterministic way of solving problems.

1. Wprowadzenie

Nauka programowania pomaga nam lepiej zrozumieć otaczający nas świat. Uczymy się zapisywania w języku formalnym różnych problemów, a następnie poprzez implementację algorytmu opisujemy rozwiązanie, które zrealizuje komputer. Nauka programowania to tak naprawdę nauka sposobu myślenia. Trzeba umieć biegle analizować problemy, dzielić je na części oraz starać się je rozwiązać w optymalny sposób. Metodami wywodzącymi się z informatyki rozwiązujemy różne problemy, nie tylko informatyczne, potrzebna jest jednak twórcza praca i krytyczna postawa. Pokażemy trzy grupy problemów: wyszukiwanie w zbiorze nieuporządkowanym, uporządkowanym i odgadywanie kodu na przykładzie gry Mastermind. Każde z tych zagadnień można zaimplementować korzystając z różnych narzędzi programistycznych, wykorzystując twórczo możliwości danego języka programowania i środowiska.

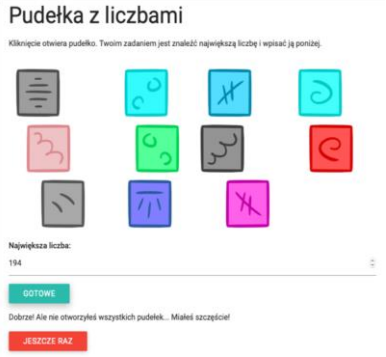
2. Wyszukiwanie w zbiorze nieuporządkowanym

Rozpatrywanym zadaniem jest znalezienie elementu, który ma określoną własność np. jest zielony, najmniejszy, najcięższy itp. Struktura zbioru nie jest określona w sposób ułatwiający wyszukiwanie, czyli mamy do czynienia ze zbiorem nieuporządkowanym. Metoda rozwiązywania polega na systematycznym przeszukaniu całego zbioru danych. To tak jak wtedy, gdy stajemy przed biblioteczką zawierającą bogaty księgozbiór. Jej właściciel powiedział, że co najmniej jedna z pozycji zawiera dedykację od

przyjaciela, ale nigdy wcześniej nie widzieliśmy żadnej z tych książek. Jak zatem mamy je odszukać? Zastanówmy się chwilę nad tym, jak szukać konkretnego elementu wśród wielu innych, podobnych do niego, jeśli nie jest znana żadna cecha wyróżniająca poszczególne elementy w zbiorze, żaden sposób ich uporządkowania. Oczywiście może się zdarzyć, że trafimy już za pierwszym razem, ale jest to bardzo mało prawdopodobne. Poza tym, nie mamy gwarancji, że znaleźliśmy każdą książkę z dedykacją. Jeśli chcemy mieć pewność, że znajdziemy to czego szukamy, musimy obejrzeć wszystkie.

Podobnych problemów w życiu codziennym jest więcej, mamy różnego typu kody, np. do zabezpieczenia walizki, w blokadzie rowerowej, sejfach czy jako pin do kart bankowych, telefonów lub tabletów. Czy łatwo jest odgadnąć taki kod? Dla czterocyfrowych pinów mamy 10 000 możliwości. Jeśli nie będziemy mieli szczęścia, to trzeba będzie sprawdzić je wszystkie. Jednak, gdy mamy trochę czasu lub komputer, który może to za nas zrobić, to nie jest to wcale zbyt trudne. Możemy przedyskutować z uczniami bezpieczeństwo stosowania takich kodów oraz dodatkowe zabezpieczenia polegające na ograniczeniu liczby możliwych prób.

O wyszukiwaniu informacji mówimy dostosowując język przekazu do możliwości uczniów. Nie musimy programować skomplikowanych algorytmów, warto zacząć od zabawy – posłużyć nam mogą kartki papieru, kartonowe pudełka zawierające różne ukryte przedmioty lub prosta aplikacja. Pod rysunkami pudełek kryją się liczby. Zadanie polega na znalezieniu największej z nich. Odkrywamy kolejno pudełka i odczytujemy liczby zapamiętując największą. Łatwo przekonać się, że jedyna pewna metoda rozwiązania zadania wymaga zajrzenia do każdego pudełka.



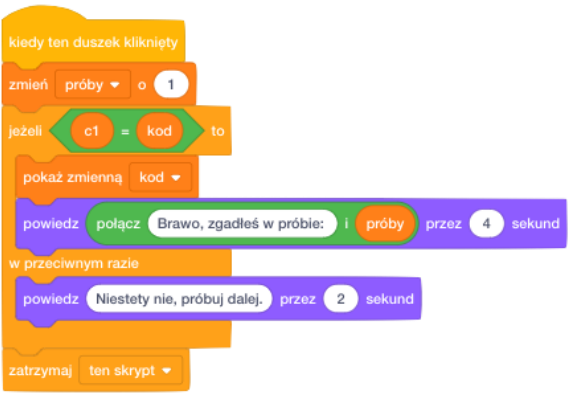
Rysunek 1 Wyszukiwanie największej liczby – aplikacja interaktywna

Powyższy zrzut ekranu aplikacji **Pudełka z liczbami** pochodzi z polskiej wersji **Przewodnika po informatyce** [4]. Znajdziemy tam również dwa przykłady zapisu algorytmu znajdowania największej liczby w nieuporządkowanej tablicy skierowane do starszych uczniów, zapisane w językach Scratch i Python.

Można też samodzielnie przygotować aplikację w Scratchu pozwalającą odgadnąć liczbę jednocyfrową. W projekcie wykorzystujemy trzy duszki: narratora – kota, duszka cyfrę oraz przycisk uruchamiający sprawdzanie.

Rysunek 2 Odgadywanie liczb jednocyfrowych

Cyfra ma 10 kostiumów, po kliknięciu w nią następuje ich cykliczna zmiana: z 0 na 1, z 1 na 2, …, z 8 na 9 i z 9 na 0. Po naciśnięciu przycisku *Sprawdź* porównywana jest wartość liczby wybranej przez użytkownika z odgadywanym kodem i zwiększany licznik prób. Jeśli liczby są takie same, to wyświetlany jest komentarz zawierający gratulacje oraz informacja, w której próbie udało nam się zgadnąć.



Rysunek 3 Skrypt wykonywany po kliknięciu w przycisk Sprawdź

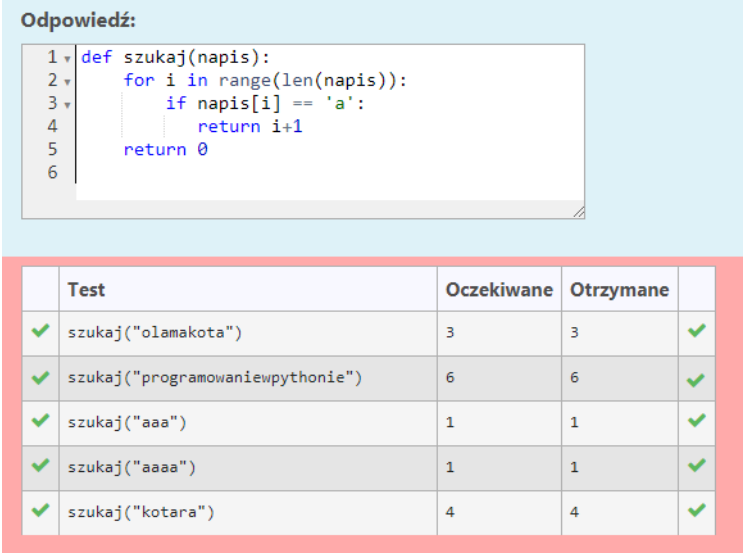
Przeanalizujmy działanie aplikacji i zastanówmy się z uczniami nad sposobem odgadywania liczby. Znajdźmy odpowiedzi na kilka pytań: *Ile prób należy wykonać, by odgadnąć liczbę? Jak jest najmniejsza możliwa liczba prób? Jak zmienią się te wartości w przypadku liczb dwucyfrowych? A czterocyfrowych?*

Opisany projekt łatwo rozbudować dodając kolejne duszki cyfry. Ciekawym zadaniem będzie wyliczenie wartości kodu, jaki został ustawiony poprzez klikanie

w poszczególne cyfry. Możemy przy okazji odwołać się do matematyki i sposobu interpretowania zapisu dziesiętnego liczb.

Rysunek 4 Wyliczanie wartości kodu ustawionego przez użytkownika

Omówiony problem wyszukiwania w zbiorze nieuporządkowanym można implementować z uczniami w różnych językach. Poniżej przykład z kursu programowania w Pythonie. Uczeń wpisuje rozwiązanie i uruchamia skrypt, który jest odpowiedzialny za sprawdzanie. Rozwiązanie jest badane na kilku zadanych testach.



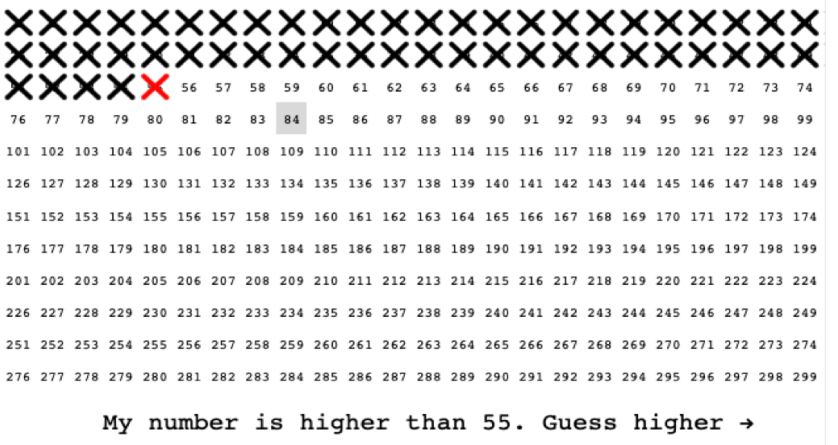
Rysunek 5 Wyszukiwanie litery „a” – rozwiązanie i testy ze sprawdzarki na platformie Moodle

3. Wyszukiwanie w zbiorze uporządkowanym

Gdy w przeszukiwanym zbiorze zostanie wprowadzony pewien porządek, możemy zastosować inny algorytm. Oczywiście przejrzenie wszystkich elementów pozwoli

odnaleźć właściwy, ale nie jest to konieczne do osiągnięcia celu. Należy jednak opracować strategię i dobrze ją zaimplementować.

W kursie **Informatyka** na portalu **Akademia Khana** [2] znajduje się ciekawy dział poświęcony algorytmom. W części **Gra w zgadywanie liczb** dostępne są dwie interaktywne aplikacje pozwalające odgadywać liczbę z podanego zakresu. Wyszukiwanie w zbiorze uporządkowanym nie wymaga rozpatrywania wszystkich liczb, ale zastosowania sprytnej metody odgadywania.



Rysunek 6 Odgadywanie liczby z przedziału od 1 do 300 w Akademii Khana

W rozdziale **Algorytmy** wspomnianego wcześniej **Przewodnika po informatyce** dostępna jest interaktywna aplikacja **Przeszukiwanie pudełek**. Tym razem szukamy konkretnej liczby w zbiorze uporządkowanym.

W Scratchu możemy zrealizować podobny projekt. Duszek najpierw losuje liczbę. Następnie czeka na podanie liczby przez użytkownika, porównuje ją z wylosowaną i informuje czy jest ona większa, mniejsza, czy równa odgadywanej wartości. Jest to interesujący projekt również ze względu na wykorzystane w nim konstrukcje programistyczne. Poza użyciem zmiennej do przechowywania odgadywanej liczby, w sposób naturalny wprowadzamy złożoną instrukcję warunkową i pętlę **powtarzaj aż**.

W środowisku Processing została przygotowana interaktywna wizualizacja. Przy jej opracowaniu wykorzystano łatwość programowania elementów multimedialnych w tym środowisku. Górny pasek na poniższym rysunku przedstawia aktualny dostępny zakres wartości, możemy myszką przesuwać wskaźnik wybieranej wartości. Dolny pasek ilustruje położenie i zakres niewykreślonych liczb w stosunku do całego przedziału [1, 1000]. Dodatkowo w projekcie zliczana jest liczba podejmowanych prób.

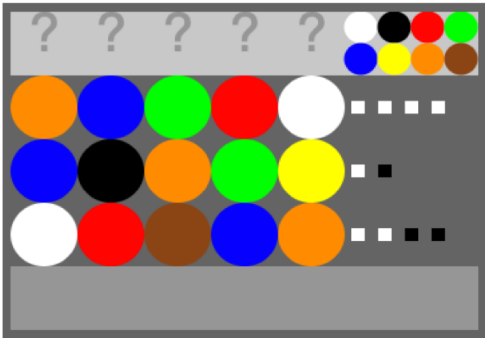
Rysunek 7 Skrypt w Scratchu – odgadywanie liczby z zakresu od 1 do 100

Rysunek 8 Wizualizacja odgadywania liczby z zakresu od 1 do 1000

Dzięki opisanym przykładom uczniowie mogą w sposób intuicyjny zrozumieć pojęcie wyszukiwania binarnego. Metodą prób i błędów znajdują najbardziej efektywny sposób wyszukiwania wartości w zbiorze uporządkowanym. Jednocześnie porównując wcześniejsze wnioski dotyczące szukania w zbiorze nieuporządkowanym zrozumieją znaczenie wprowadzania porządku w przypadku dużych zbiorów danych. Warto także zwrócić uwagę starszych uczniów na praktyczne zastosowania poznanej przez nich metody na przykładzie znajdowania miejsca zerowego funkcji metodą bisekcji.

4. Mastermind

Mastermind to gra polegająca na odgadywaniu kodu – kombinacji różnokolorowych kulek. W pojedynczym ruchu gracz próbuje odgadnąć kod poprzez zapytanie – ustawienie przykładowego układu. Jeśli dana kulka jest na dobrym miejscu, to otrzymujemy za nią czarny pionek, jeśli w dobrym kolorze, ale na złym miejscu – biały. Pełna odpowiedź jest sumą czarnych i białych pionków. Łatwo znaleźć różne implementacje tego problemu, gdzie zadaniem jest odgadnięcie kodu złożonego z 4, 5 lub większej liczby elementów. Kody mogą składać się z różnobarwnych pionków, liczb lub liter, a nawet symboli albo obrazków.



Rysunek 9 Przykładowa implementacja gry Mastermind w Processingu

W przypadku tej aplikacji komputer zajmuje się jedynie wyświetlaniem gry i sprawdzaniem, czy udało nam się odgadnąć kod. Wyobraźmy sobie jednak sytuację odwrotną, kiedy komputer znajduje układ pionków wymyślony przez człowieka. Algorytm najkorzystniejszej strategii nie jest zagadnieniem łatwym, gdyż mamy nie tylko wiele możliwości (liczba kolorów do potęgi długość kodu), ale w każdym kroku dodajemy wiele ograniczeń. Intuicja podpowiada nam, że dobrze, gdy w każdym ruchu ograniczana jest liczba możliwych rozwiązań, tak długo aż nie zgadniemy kodu. Przy okazji tego problemu można wprowadzić zagadnienie algorytmów genetycznych, które nie są deterministycznym rozwiązaniem, lecz opierają się na mechanizmach probabilistycznych. Jak sama nazwa wskazuje, są odzwierciedleniem procesów ewolucyjnych, które zachodzą w przyrodzie.

Podobnie jak u organizmów żywych, implementując algorytm genetyczny mamy do czynienia z krzyżowaniem, mutacją i permutacją chromosomów. W przypadku gry Mastermind poprzez chromosom będziemy rozumieć pojedynczy kod. Krzyżowanie to proces, w którym dla dwóch chromosomów wybieramy punkt krzyżowania, a następnie zamieniamy część pierwszego z drugim.

Rysunek 10 Przykład krzyżowania dla 5 elementowej gry

Mutacja to operacja, w której jeden gen jest losowo zastępowany przez inny, a permutacja polega na zamianie kolejności.

Rysunek 11 Przykład mutacji i permutacji dla 5 elementowej gry

```
Combination.prototype.crossover = function(other) {
        var crossoverPoint = getCrossoverPoint(this.code.length);
        var firstChild = this.code.slice(0, crossoverPoint).concat(other.code.slice(crossoverPoint));
        var secondChild = other.code.slice(0, crossoverPoint).concat(this.code.slice(crossoverPoint));
        return [new Combination(firstChild), new Combination(secondChild)];
};

Combination.prototype.mutate = function() {
        var position = Math.floor(Math.random() * NUM_FIELDS);
        var newValue = _.sample(_.without(SYMBOLS, this.code[position]));
        this.code[position] = newValue;
};

Combination.prototype.permutate = function() {
        var positions = _.range(NUM_FIELDS);
        var toPermute = _.sample(positions, 2);
        this.code[toPermute[1]] = [
                this.code[toPermute[0]],
                this.code[toPermute[0]] = this.code[toPermute[1]]
        ][0];
}
```

Rysunek 12 Fragment implementacji w JavaScript

Schemat algorytmu dla gry Mastermind wygląda następująco:

1. Wygeneruj populację złożona z różnych możliwych wartości kodów – chromosomów.
2. Wybierz z nich jeden i sprawdź jego ocenę – liczbę czarnych i białych pionków. Jeśli liczba czarnych jest równa długości kodu, to kod został znaleziony.
3. Promując osobniki lepiej przystosowane, za pomocą krzyżowania, mutacji i permutacji zmodyfikuj populację.
4. Powróć do punktu 2.

Przy implementacji trzeba jeszcze uściślić poszczególne kroki. W punkcie pierwszym określić maksymalną wielkość populacji, a w drugim – sposób wyboru elementu.

```
NUM_SYMBOLS = 6;
NUM_FIELDS = 4;

SYMBOLS = ["A", "B", "C", "D", "E", "F"];

MUTATION_PROBABILITY    = 0.2; // 0.6
PERMUTATION_PROBABILITY = 0.1; // 0.4
INVERSION_PROBABILITY   = 0.05;

MAXGEN = 250;
POPULATION_SIZE = 150;
SET_SIZE = 110;
FIT_A = 1;
FIT_B = 2;
```

Rysunek 13 Fragment implementacji w JavaScript

Punkt trzeci wymaga określenia prawdopodobieństwa krzyżowania, mutacji, permutacji oraz sposobu wyboru osobników lepiej przystosowanych. Wybór funkcji dostosowania jest bardzo ważny. Jeśli obliczamy dopasowanie chromosomu, można porównywać go z każdą poprzednią próbą – obliczamy liczbę czarnych i białych pionków, które zdobyłyby te próby, gdyby dany kod był kodem do odgadnięcia [1].

Okazuje się, że w praktyce tak zaprojektowany algorytm genetyczny daje dobre rezultaty. Konkretne wyniki zależą od doboru parametrów metodą prób i błędów.

Rysunek 14 Wynik działania gry dla przykładowych danych [3]

5. Podsumowanie

W informatyce rozważając problemy chcemy otrzymać rozwiązania, które są poprawne i efektywne, a najlepiej jeszcze zrozumiałe. Większość klasycznych algorytmów stosuje deterministyczną procedurę znajdowania rozwiązania, która gwarantuje znalezienie rozwiązania optymalnego. Inaczej jest z algorytmami genetycznymi, które opierają się na prawdopodobieństwie. Pozwalają one sobie radzić również tam, gdzie rozwiązanie deterministyczne jest skomplikowane lub czasochłonne. Jeśli za jeden z celów nauczania informatyki stawiamy sobie poznanie świata, to te dwie grupy pro-

blemów to odzwierciedlają. Nie sposób zrozumieć zachodzących w świecie zjawisk bez podejścia deterministycznego i probabilistycznego.

Literatura

1. Berghman L., Goossens D., Leus R., *Efficient solutions for Mastermind using genetic algorithms*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.496.276&rep=rep1&type=pdf, ostatni dostęp 28.05.2019 roku.
2. *KhanAcademy,* kurs *Informatyka – Algorytmy*, https://pl.khanacademy.org/computing/computer-science/algorithms, ostatni dostęp 28.05.2019 roku.
3. Mastermind AI, https://github.com/ognjenvucko/mastermind-ai, ostatni dostęp 28.05.2019 roku.
4. *Przewodnik po informatyce. Wersja dla nauczyciela*, https://bezkomputera.wmi.amu.edu.pl/ppi/teacher, ostatni dostęp 28.05.2019 roku.