# Gra MasterMind

środa, 10 stycznia 2018 01:00

# Zadanie - gra MasterMind

## Informacje o grze:

- Informacje na Wikipedii: https://pl.wikipedia.org/wiki/Mastermind_(gra_planszowa)
- Zagraj w wersji on-line: http://www.zagrajsam.pl/dzialy_gier.php?gra=3
- Zaawansowana analiza matematyczna gry w czasopiśmie Delta: http://www.deltami.edu.pl/temat/matematyka/gry_zagadki_paradoksy/2013/05/26/mastermind.pdf

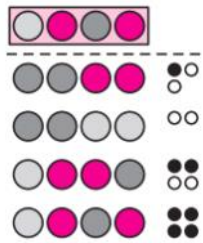
mastermind

## Założenia (gra klasyczna):

Kody kolorów: {red (r), yellow (y), green (g), blue (b), magenta (m), cyan (c) } = {r, y, g, b, m, c}

## Scenariusz gry:

1) Pierwszy gracz ustala kod (sekretny) składający się z 4 kolorów - kolejność ma znaczenie, kolory mogą się powtarzać.
2) Drugi gracz próbuje odgadnąć sekretny kod:
   a. Podaje swoją kombinację 4 kolorów w odpowiedniej kolejności
   b. Pierwszy gracz weryfikuje propozycję i odpowiada, podając liczbę trafień dokładnych (tzn. właściwy kolor na właściwej pozycji - odpowiednia liczba kółek czarnych) oraz trafień niedokładnych (tzn. właściwy kolor na niewłaściwej pozycji - odpowiednia liczba kółek białych)
   c. Kroki powyższe powtarzane są wielokrotnie
3) Gra kończy się, gdy drugi gracz odgadnie sekretny kod
4) Dobry gracz po 4-5 próbach potrafi odgadnąć sekretny kod (w zależności od stosowanej strategii - strategia bazuje na dobrym początku i odpowiednim wnioskowaniu w dalszych krokach).



## Analiza:

- Pojedynczy kod = 4-elementowy ciąg ze zbioru 6-elementowego {r, y, g, b, m, c}. Zatem kolejność jest ważna.
- Przestrzeń wszystkich kodów - wariacje z powtórzeniami $\{(x_1, x_2, x_3, x_4): x_i \in \{r, y, g, b, m, c\}\}$, jest ich $6^4 = 1296$.
- Ocena próby: $f(x_1, x_2, x_3, x_4) \rightarrow (n_d, n_n)$, gdzie:
  - $n_d$ - liczba dokładnych trafień $(0..4)$
  - $n_n$ - liczba niedokładnych trafień $(0..4)$
  - $n_n + n_d \leq 4$
- Przykład:

| Propozycja | $n_d$ | $n_n$ |
|---|---|---|
| rrrr | 1 | 0 |
| yyyy | 1 | 0 |
| gggg | 0 | 0 |
| bbbb | 1 | 0 |
| mmmm | 0 | 0 |
| cccc | 1 | 0 |

Czy potrafisz ustalić sekretny kod?
Jeszcze kilka podpowiedzi:

| Propozycja | $n_d$ | $n_n$ |
|---|---|---|
| rybc | 1 | 3 |
| rycb | 0 | 4 |
| rcyb | 1 | 3 |
| cryb | 0 | 4 |

Czy teraz potrafisz ustalić sekretny kod?

## Strategia 1 - brute force:

1. Dla każdej możliwej wariacji kodu
    a. Sprawdź, czy funkcja $f$ zwróci wynik (4, 0)
    b. Jeśli tak, gra zakończona.

Złożoność obliczeniowa: W pesymistycznym przypadku odgadniesz sekretny kod w 1296 ruchu.

## Strategia 2

1. Dla każdego koloru sprawdź XXXX
2. Teraz wiesz, które kolory zostały użyte w sekretnym kodzie (cztery lub mniej)
    a. Dla każdej 4-elementowej wariacji ze zbioru użytych kolorów sprawdź, czy funkcja $f$ zwróci wynik (4, 0)

W kroku 1 wykonasz 6 lub mniej ruchów. Przeprowadź analizę liczby ruchów w kroku 2. O ile "lepsza" jest Strategia 2 od Strategii 1?

## Strategia 3

1. Wygeneruj listę $L$ wszystkich możliwych kodów (przestrzeń potencjalnych rozwiązań gry)
2. Powtarzaj do momentu, aż trafisz:
    a. Wybierz dowolny element z listy $L$
    b. Oceń wynik za pomocą funkcji $f$
    c. Zaktualizuj listę $L$ (usuń te kody, które są niespójne z oceną)

Ta strategia przypomina strategię, którą stosuje człowiek.
Zbadaj złożoność obliczeniową (i pamięciową).

## Strategia 4

1. Poddaj ocenie następujące kombinacje:
    a. `rryy`
    b. `rgrg`
    c. `bbmm`
    d. `bcbc`

    Jeśli trafiłeś, to koniec gry.
2. Utwórz listę $L$ wszystkich możliwych kodów niesprzecznych z odpowiedziami z kroku 1.
3. Powtarzaj do momentu, aż trafisz:
    a. Wybierz dowolny element z listy $L$
    b. Oceń wynik za pomocą funkcji $f$
    c. Zaktualizuj listę $L$ (usuń te kody, które są niespójne z oceną)

Spróbuj rozegrać grę stosując podaną strategię. Będziesz zaskoczony, jak małoliczna będzie lista $L$ z kroku 2.
Zbadaj (pesymistyczną) złożoność obliczeniową dla tej strategii (w różnych wariantach).

# ZADANIA

## Zadanie 1

Napisz w C# program interaktywny, który umożliwi rozegranie klasycznej wersji gry między człowiekiem a komputerem, gdzie komputer losuje kod i udziela odpowiedzi, a człowiek próbuje odgadnąć sekretny kod. Przyjmij $\mathrm{n} = 6$ kolorów, kod o długości $\mathrm{k} = 4$ oraz ograniczenie czasowe $\mathrm{t} = 9$ ruchów, w których należy udzielić poprawnej odpowiedzi. Zakładamy, że komputer udzielając odpowiedzi się nie myli i nie oszukuje.

a) Napisz program w wariancie konsolowym (CLI), wykorzystaj możliwości klasy Console (np. kolorowanie wyprowadzanych tekstów)
b) Napisz program w wariancie GUI (WinForms lub lepiej WPF lub lepiej UWP)
c) Jeśli potrafisz, zrób aplikację na urządzenia mobilne (Xamarin)

Podpowiedzi:

1) Utwórz klasę Game, która będzie "silnikiem" dla gry -- dostarczy stosownego API, z którego skorzystasz w implementacji interfejsu użytkownika (CLI lub GUI). Klasa Game ma być niezależna od zastosowanego wariantu interfejsu użytkownika.
2) Obiekt klasy Game musi pamiętać stan gry (wyobraź sobie, że robisz snapshot gry, zapamiętujesz jej stan, by móc np. jutro go przywrócić i grę kontynuować - ta prosta analiza myślowa pozwoli Ci ustalić pola klasy).
3) Zaprogramuj w klasie Game metody, które mogą przydać się interfejsowi użytkownika.
4) Jednym z elementów stanu gry jest sekretny kod. Pamiętaj, że "świat zewnętrzny" (np. interfejs

użytkownika) nie może go znać podczas rozgrywki (może go poznać dopiero po jej zakończeniu lub przerwaniu -- poddaniu się). Natomiast w trakcie rozgrywki musisz dostarczyć mechanizm oceny propozycji kodu (czyli funkcję $f$).

5) W języku C# od wersji 7.1 (.Net Core 2.0) funkcja może zwracać więcej niż jedną wartość (np. parę wartości - jak w naszym przypadku). Możesz (ale nie musisz) zaimplementować tę funkcjonalność.

## Zadanie 2.

Rozbuduj program z zadania 1 tak, aby role zostały odwrócone: człowiek wymyśla sekretny kod, a komputer próbuje go odgadnąć (stosując jedną z opisanych strategii). Musisz rozważyć sytuację, że człowiek może pomylić się przy ocenie kodu zaproponowanego przez komputer - komputer w swojej analizie powinien wykryć taką sytuację

## Zadanie 3.

Rozbuduj/sparametryzuj programy z poprzednich zadań tak, aby:

a) Liczba kolorów $\mathrm{n} = 6..8$
b) Długość kodu $\mathrm{k} = 4..6. \mathrm{k} < \mathrm{n}$

Użytkownik na początku gry określa jej parametry początkowe.

## Zadanie 4.

Istnieje wariant gry MasterMind, w którym dopuszczalne jest oszukiwanie (np. jedno lub dwie błędne odpowiedzi). Spróbuj zaimplementować taką grę.

## Zadanie 5.

Istnieje wariant gry, w którym zamiast kolorów stosowane są cyfry $0..9$ ($\mathrm{n} = 10$). Spróbuj zaimplementować grę dla $\mathrm{k} = 4$. Oczywiście liczba prób musi być wieksza.