Gry kombinatoryczne 2021: Van der Waerden (8c)
Dokumentacja końcowa

Jakub Kosterna, Marceli Korbin, Mikołaj Malec

Maj-czerwiec 2021

Spis treści

1 Ogólnie

Celem projektu jest implementacja aplikacji konsolowej - nazywanej od teraz grą - napisanej w języku **Java**. Docelowo będzie ona programem możliwym do pobrania i uruchomienia przez każdego użytkownika, posiadającego oprogramowanie Java. Warunkiem uruchomienia gry będzie wpisanie w wiersz polecenia konkretnej komendy. Jest to gra 0-2 osobowa typu offline - wszystko odbywa się w zakresie jednego komputera.

2 Zasady

W grze niezależnie od formy zawsze bierze udział **2 graczy**. Niekoniecznie muszą być to dwaj fizyczni użytkownicy - dostępna będzie także tryb komputer vs. komputer, którego automatyczne akcje będzie mógł później zobaczyć i przeanalizować fizyczny użytkownik. Nazwijmy jednego z nich *dobrym*, a drugiego - *złym*. Na początku definiowane są cztery wartości:

- n - liczba tzw. pól (zwanych także pozycjami)
- k - liczba kolorów (zwanych także barwami) dostępnych w grze
- m - długość ciągu arytmetycznego pól pokolorowanych w ten sam sposób ku zwycięstwu gracza dobrego
- l - liczba dostępnych kolorów proponowanych przez gracza dobrego, $l <= k$

Na zmianę, począwszy od gracza dobrego, każdy z graczy wykonuje po swoim ruchu:

* dla gracza dobrego ruchem jest wybranie jednego z n pól, które nie zostały jeszcze wybrane, a także k dowolnych kolorów;
* gracz zły ma za zadanie wybrać jeden z zaproponowanych m kolorów przez gracza dobrego, co skutkuje pokolorowaniem danej pozycji ową barwą.

Gracz dobry dąży do wystąpienia ciągu arytmetycznego m tych samych kolorów, o dowolnym kroku. Gracz zły - wręcz przeciwnie.

Grę można uruchomić w dwóch trybach - demo oraz testowym. Kolejny ruch każdego gracza fizycznego zostaje wykonany po naciśnięciu klawisza.

- W trybie demo rozgrywana jest pojedyncza gra; na każdym etapie widać rezultat w postaci planszy.
- W trybie testowym rozgrywana jest zadana liczba gier, po rozegraniu których wyświetlane są wyniki, tzn. liczba wygranych i przegranych. Ponadto, gdy gra jest rozgrywana pomiędzy graczami automatycznymi, plansza w trakcie kolejnych rund jest niewidoczna.

3 Uruchomienie

W celu włączenia gry użytkownik musi być w posiadaniu oprogramowania, które jest przystosowane do uruchomienia pliku o rozszerzeniu *.jar*. Aby rozpocząć rozgrywkę, wymagane jest wykonanie w konsoli komendy:

```
java -jar VanderWaerden.jar
```

... w katalogu głównym aplikacji.

4 Rozgrywka - esencja

Na zmianę ruch wykonują obaj gracze. Wpierw dobry wpisuje numer pola, który chciałby pokolorować. Jeżeli wejście jest niepoprawne, terminal prosi o powtórzenie akcji. Następnie gracz ma za zadanie wpisać kolejne l kolorów, z których do pokolorowania wcześniej wspomnianego pola, gracz zły wybierze jedno. Gra sprawdza za każdym razem, czy ruch jest możliwy i weryfikuje, czy któryś z graczy nie wygrał. Gra kończy się w przypadku wygranej dobrego gracza, bądź złego gracza - ten drugi wypadek jest równoważny sytuacji pokolorowania wszystkich pól.

W danym momencie wyświetlana plansza złożona z kolejnych pól może być postaci przykładowo:

1234567

(gdzie kolejne barwy to: biały biały czerwony niebieski biały czerwony biały)

5 Rozgrywka - szczegółowo

5.1 Ekran początkowy

Po potwierdzeniu przez klienta operacji włączenia aplikacji, ten ujrzy interfejs główny. Na start gracz ujrzy komunikat powitalny:

```
Witaj, epicki graczu!! :D
```

5.2 Konfiguracja gry

...żeby zostać poproszonym o podanie pierwszych danych wejściowych:

```
Masz kolegę, z którym pragniesz zagrać?
Czy może wolałbyś ujrzeć symulację komputerową?
Wpisz: "1vs1" dla rozgrywki z kumplem,
"p1" dla gry z komputerem jako gracz 1,
"p2" dla gry z komputerem jako gracz 2
lub "comp" dla przeprowadzenia gry automatycznej
```

Po wprowadzeniu odpowiedniego ciągu znaków, określane jest także, którzy gracze będą dokonywać swoich akcji automatycznie (czyli komputer sam wybiera, które pola i jak optymalnie jest mu odznaczyć), a którzy - manualnie (czyli standardowo). Użytkownik wpisuje wartości wejściowe z klawiatury, a następnie gra kieruje gracza (graczy) do głównej rozgrywki.

Uwaga! W przypadku wprowadzenia niepoprawnego łańcucha znaków, wyświetla się komunikat:

```
Wpisz jedną z czterech wartości: 1vs1, p1, p2, comp
```

...i aplikacja czeka na korektę.

Jeżeli został wybrany którykolwiek tryb gry z udziałem komputera, aplikacja zadaje pytanie o stopień zaawansowania gracza automatycznego (lub obu graczy w wypadku wybrania `comp`):

```
Wpisz poziom gracza pierwszego: "weak" lub "strong" //(tryby p2, comp)
Wpisz poziom gracza drugiego: "weak" lub "strong" //(tryby p1, comp)
```

Poziom `weak` oznacza gracza, który dokonuje wszystkich swoich akcji w sposób losowy; poziom `strong` nadaje graczowi strategię, która jest szerzej opisywana w punktach 6.4, 6.5. (Niepoprawny ciąg znaków jest korygowany analogicznie, jak poprzedni).

Następnie po kolei komputer prosi o cztery liczby, które są poprzedzone następującymi wyjściami:

```
Wpisz proszę, ile w grze ma być pól (pozycji do zakolorowania)
Wpisz proszę, ile w grze ma być kolorów dostępnych w grze
Wpisz proszę, ile w grze ma być pól w ciągu arytmetycznym o tej
```

```
    samej barwie, ku zwycięstwu gracza pierwszego (nie białej)
Wpisz proszę, ile w grze ma być kolorów dostępnych do kolorowania
    przez gracza drugiego w każdej turze
```

W pierwszej kolejności wymagane jest podane podstawowych danych wejściowych, opisujących specyfikację naszej rozgrywki: n, k, m i l. Obowiązkowe jest, aby te wartości były dodatnimi liczbami całkowitymi, a wartość l powinna ponadto być mniejsza lub równa od k. Kolejne pola od teraz będą wyświetlane i przetwarzane jako kolejne liczby naturalne [począwszy od 1]. Na początku gry wszystkie liczby mają kolor biały, zaś w zależności od pokolorowania - na dalszych etapach zmieniają swoje barwy. Aplikacja posiada także obsługę błędów w przypadku podania danych nieliczbowych bądź niespełniających powyższych kryteriów - gracz otrzymuje wtedy stosowny komunikat i proszony jest o ponowne wpisanie danych.

Pozostaje jeszcze kwestia wyświetlania wyników - na sam koniec konfiguracji wyświetla się jeszcze wiadomość z pytaniem:

```
No dobrze, ale czy masz czas i serce do oglądania całej rozgrywki?
Jeśli tak - spróbuj trybu "demo"! W przeciwnym wypadku doskonały
    dla Ciebie będzie "test" :)
Wpisz: "demo" lub "test"
```

Tak jak i w wyjaśnieniu - dla trybu (demo)nstracyjnego wyświetlany jest każdy kolejny etap gry, natomiast (test)owego - przede wszystkim finalny efekt, z zupełnym pominięciem przebiegu rozgrywki w grze między graczami komputerowymi.

5.3 Właściwa rozgrywka

Grę zawsze rozpoczyna gracz "dobry". Niezależnie od sytuacji czy za jego rolę odpowiada fizyczny użytkownik, czy też automat, za każdym ruchem dostaje dwie wskazówki:

```
Graczu 1, podaj numer pola
Podaj kolory
```

Po tej komendzie program czeka na wpisanie jednej liczby całkowitej ¡= n (i jeszcze niepokolorowanej), a następnie ciągu l kolorów z przedziału ¡1; k¿. Wymagane jest podanie dokładnej liczby w zadanym przedziale - w innym wypadku wyświetlany jest komunikat:

```
Nieprawidłowe numery! Podaj jeszcze raz kolory
```

... i komputer czeka na ponowne wprowadzenie danych wejściowych.

Następnie dla gracza "złego" sytuacja jest jeszcze prostsza - ten dostaje tylko jeden komunikat:

```
Graczu 2, podaj numer koloru
```

... i wymagane jest wprowadzenie tylko jednej liczby - oczywiście jednej z tych, które wcześniej zaproponował gracz dobry.

5.4 Koniec gry

Gra może zakończyć się w jednym z dwóch przypadków:

1) Został utworzony ciąg arytmetyczny m tych samych kolorów, co skutkuje wygraną gracza "dobrego"

2) Wszystkie ruchy zostały wykonane, a powyższy warunek zwycięstwa "dobrego" nie został spełniony - skutkuje tu triumfem gracza "złego"

W zależności od scenariusza, zostaje wyświetlony jeden z dwóch prostych komunikatów:

```
Gracz 1 wygrywa!
Gracz 2 wygrywa!
```

W trybie demonstracyjnym, w wypadku wygranej gracza pierwszego, zostaje dodatkowo pokazana tablica z zaznaczonym ciągiem wygrywającym.

6 Strategia

Zdecydowaliśmy się na implementację algorytmu typu *brute-force*. Algorytm wybiera najlepszy ruch lokalnie, co tak naprawdę jest strategią globalną. Zostało to dowiedzione symulacyjne, co dowiódł przygotowany przez nas skrypt napisany w języku R.

Ważną uwagą jest fakt, że mamy tu styczność z typową **grą deterministyczną bez tajnych informacji**. W praktyce oznacza to, że istnieje strategia wygrywająca. Efektem tego jest fakt, że dla każdej gry istnieje graniczna wartość długości planszy, która determinuje wygraną albo przegraną.

6.1 Liczby Van der Waerdena

Kluczowym ku zrozumieniu strategii i jej efektywności jest wiedza o **twierdzeniu *Van der Waerdena***. Mówi ono, że dla każdych całkowitych liczb $c > 0$ i $n > 0$, istnieje pewna liczba V, taka że jeśli ciąg złożony z następujących po sobie V liczb jest pokolorowany na c różnych kolorów, wówczas ciąg ten musi zawierać podciąg o postępie arytmetycznym o długości n, taki, że jego elementy są tego samego koloru. W efekcie, kolorując kolejne liczby na pewną liczbę kolorów, przy pewnej długości ciągu uzyskamy swego rodzaju regularność.

Liczby będące minimalnymi V dla takich c kolorów i n długości ciągów arytmetycznych nazywamy **liczbami *Van der Waerdena***. Co ciekawe, wartości V rosną stosunkowo szybko wraz ze wzrostem c i n i już dla małych wartości (parenaście x parenaście) ich znajdowanie jest bardzo ciężkie nawet komputerowo.

Dla małych c i n prezentują się one następująco:

n/c	2 kolory	3 kolory	4 kolory	5 kolorów	6 kolorów
3	9	27	76	> 170	> 223
4	35	293	> 1048	> 2254	> 9778
5	178	> 2173	> 17705	> 98740	> 98748
6	1132	> 11191	> 157209	> 786740	> 1555549
7	> 3703	> 48811	> 2284751	> 15993257	> 111952799
8	> 11495	> 238400	> 12288155	> 86017085	> 602119595
9	> 41265	> 932745	> 139847085	> 978929595	> 6852507165

Warto wspomnieć, że swego czasu **Timothy Gowers** dowiódł nierówność: $w(c,n) \leqslant 2^{2^{r^{2^{2^{k+9}}}}}$

Dla naszej gry jest to ważne o tyle, że dla gry o długości większej lub równej liczbie Van der Waerdena, niezależnie od strategii graczy, gracz dobry zawsze wygra. Dlatego nasze poszukiwane "graniczne" liczby będą od nich mniejsze.

Ponadto warto zwrócić uwagę, że jeśli plansza była by nieskończona, to niezależnie od wielkości (skończonych) pozostałych parametrów gry gracz dobry by zawsze wygrał. Jest to bezpośredni wniosek z tw. Van der Waerdena.

6.2 Wstępne uwagi dotyczące gry

Zgodnie z dokumentacją wstępną rozważamy dane wejściowe złożone z czterech liczb naturalnych [uwaga! nieco inne oznaczenia]:

- n - liczba pól planszy
- *max_colors* - liczba kolorów, które wybiera gracz dobry, a następnie z tych wybranych gracz zły wybiera ostateczny kolor
- k - wymagana długość ciągu arytmetycznego ku wygranej
- *palete_game* - liczba kolorów do wyboru przez gracza "dobrego"

Nie ma sensu rozważać gier, których:

- max_colors jest równa 1
- k jest równe 1 lub 2
- n nie jest znacząco większe niż k

Nie ma też sensu rozważać gier, w których n jest większe/równe liczbie Van der Waerdena dla danych max_colors i k, ponieważ wtedy gracz dobry oferując ciągle te same kolory niezależnie od kolejności oferowanych punktów zawsze będzie wygrywał. Nie jest to ciekawy przykład.

W dalszej części pracy zakładamy, że **żadne z powyższych punktów nie zachodzi**.

6.3 Gra, gdzie palete_game = max_colors

Dlaczego gra zakłada, że gracz dobry wybiera z palety kila kolorów, a następnie zły wybiera? Ponieważ dla gry, w której gracz dobry daje do wyboru te same kolory, **skończy się to dla niego porażką**. Ponieważ n jest mniejsze od liczby Van der Waerdena, zatem istnieje taki "wzorzec", który nie zawiera ciągu arytmetycznego. Wystarczy, że zły będzie wybierał kolory z tego wzorca, to niezależnie od kolejności wybieranych pozycji, dobry przegra.

6.4 Strategia opisowo

Czy zatem dobry może wygrać? Może, ale musi "popsuć" plan złego. W tym celu w odpowiednim momencie oferuje złemu zestaw, który nie zawiera potrzebnego przez złego koloru, a zamiast niego "nowy" kolor. Zły może wybrać nowy kolor, ale dobry wybrał taki moment, że nawet z dodatkowym kolorem niezależnie co zrobi zły, gra będzie musiała utworzyć ciąg arytmetyczny.

Jako przykład podamy prostą grę o parametrach n=7, k=3, max_colors=2, palete_game=3

---

1 2 3 4 5 6 7
0 0 0 0 0 0 0
Good pick: 1 Colors to chose: 1 — 2
Bad pick: 1

---

1 2 3 4 5 6 7
1 0 0 0 0 0 0
Good pick: 2 Colors to chose: 1 — 2
Bad pick: 2

---

1 2 3 4 5 6 7
1 2 0 0 0 0 0
Good pick: 3 Colors to chose: 1 — 2
Bad pick: 1

---

1 2 3 4 5 6 7
1 2 1 0 0 0 0
Good pick: 6 Colors to chose: 2 — 3
Bad pick: 3

---

1 2 3 4 5 6 7
1 2 1 0 0 3 0
Good pick: 4 Colors to chose: 1 — 3
Bad pick: 3

---

1 2 3 4 5 6 7
1 2 1 3 0 3 0

Good pick: 5 Colors to chose: 1 — 3

W tym momencie niezależnie co wybierze zły, gra zakończy się zwycięstwem dobrego. Ta gra pokazuje, że nawet pomimo istnienia ciągu niearytmetycznego (np. 1 2 2 1 1 2 2), dobry jest w stanie wygrać.

6.5 Strategia szczegółowo

Nasze rozwiązanie zdecydowanie nie jest najszybsze i będzie działało wręcz bardzo wolno dla "dużych" danych wejściowych, ale jego brutalność zapewnia optymalną strategię dla obu graczy i owocuje w sukces, o ile jest on wskazany przy bezbłędnie inteligentnej grze zawodników.

Biorąc pod uwagę założenie o bezbłędności obojga graczy, dla każdego ruchu obaj są w stanie przewidzieć potencjalne scenariusze w zależności od wziętego pod lupę rozwiązania i tym samym rozważyć je pod względem opłacalności przez pryzmat warunku własnej wygranej.

Drzewo działa na zasadzie rekurencji. Każda pozycja może być oznaczona jako wygrywająca (wygrywa dobry), lub przegrywającą (wygrywa zły). Gracz dobry wybiera taką pozycję i takie ruchy, które będą prowadziły do ruchów wygrywających. Jeśli z danej pozycji istnieje ruch na pozycję wygrywającą, to ta pozycja też jest pozycją wygrywającą. Jeśli dobry jest zmuszony dać złemu możliwość wyboru ruchy do pozycji przegrywającej, to zły to wykorzysta, zatem ten ruch jest przegrywający. W ten sposób wszystkie możliwe pozycje są oznaczane jako wygrywające lub jako przegrywające (dla dobrego). Jeśli pusta plansza została oznaczona jako wygrywająca, to dla danych parametrów gry zawsze wygra dobry, jeśli przegrywająca, zawsze wygra zły.

Dzięki temu można zrobić tabele podobną do powyższej, aczkolwiek złożoność obliczeniowa jest znacznie większa od "zwykłych" liczb Van der Waerdena. Przeprowadziliśmy kilka eksperymentów, które pozwoliły wyznaczyć najmniejsze z tych liczb.

n/c	3 kolory, wybiera 2	4 kolory, wybiera 3
3	7	< 23
4	< 25	

Z naszych dalszych obserwacji wynika, że możliwość wyboru z większa palete_game nie wpływa na strategie graczy, lecz jest to temat do dalszej analizy.

6.6 Trochę refleksji o optymalnej strategii z *Math Overflow*

Problem był rozważany już wcześniej przez profesjonalnych matematyków i amatorów. Ciekawa dyskusja wywiązała się na wspomnianym serwisie, gdzie można lepiej zaznajomić się z tematem i wyciągnąć kilka dodatkowych wniosków. Link do sformułowanego pytania i komentarze internautów można znaleźć pod drugim linkiem, natomiast owa witryna prowadzi do wniosków takich jak:

- dla wspomnianej wariacji gry dla każdej gry typu k dozwolonych pozycji od gracza złego dla dobrego oraz ciągu geometrycznego długości 3, graczowi dobremu wystarczy nieco lekko po nad k tur, by wygrać.

- w owej wariacji warto przyjrzeć się potęgom dwójki - ze względu na ich niearytmetyczność przynoszą one średnio sprzyjające sytuacje graczowi złemu.

- niezależnie od liczby pól i kolorów, dla skończonych gier typu k pozycji do wyboru i wymagany ciąg geometryczny długości 3, można stosunkowo łatwo wyliczyć maksymalną liczbę gier, które się odbędą.

6.7 Dowód, że liczba Van der Waerdena dla 2 kolorów i k=3 jest mniejsza niż 325

Weźmy dowolny ciąg kolorów $(a_i)_i$, gdzie $a_i \in \{0,1\}$ dla $i \in \{1,\cdots,325\}$.

Podzielmy $\{1,\cdots,325\}$ na 65 pięciu elementowych podciągów, takich, że przyjmują formę $(5b+1, 5b+2, \cdots, 5b+5)$ dla $b \in \{0,\cdots,64\}$. Ponieważ 5 elementowy ciąg z dwoma kolorami można ustawić na 32 sposoby, to w pierwszych 33 z zasady szufladkowej muszą istnieć 2 identyczne pokolorowania podciągów, czyli istnieją takie b_1 i b_2, że $a_{5b_1+k} = a_{5b_2+k}$ dla każdego $k \in \{1,\cdots,5\}$.

Pomiędzy kolorami a_{5b_1+1}, a_{5b_1+2}, a_{5b_1+3} muszą istnieć z zasady szufladkowej 2 takie same, czyli, że istnieje takie c_1 i c_2, gdzie $c_1 < c_2$, że $a_{5b_1+c_1} = a_{5b_1+c_2}$.

Niech $c_3 := 2 * c_2 - c_1$. Jeśli $a_{5b_1+c_1} = a_{5b_1+c_3}$ to koniec dowodu, w przeciwnym przypadku weźmy $b_3 := 2 * b_2 - b_1$. Warto zwrócić uwagę, że $b_3 \leqslant 64$. Jeśli $a_{5b_1+c_3} = a_{5b_3+c_3}$ to $(5b_1 + c_3, 5b_2 + c_3, 5b_3 + c_3)$ tworzy ciąg arytmetyczny, w przeciwnym wypadku $(5b_1 + c_1, 5b_2 + c_2, 5b_3 + c_3)$ nie tworzy ciągu arytmetycznego.

7 Testowanie

7.1 Przykładowa gra - wygrana dobrego

Uwaga! Niżej zmienimy sposób kolorowania. Ku lepszemu poglądowi, niżej roboczo: zamiast liczb na kolejnych miejscach w różnych kolorach, i-ta pozycja od lewej będąca "_" oznacza, że i-te pole jest niepokolorowane, zaś kolejne liczby dla i-tego miejsca oznaczają kolejne barwy owych pól.

Dane wejściowe:
- n = 9
- k = 3
- l = 3
- m = 2

Stan początkowy planszy:
_____
(1): pozycja: 3 kolory: 2,3
(2): 2
__2_____
(1): pozycja: 4 kolory: 1,3
(2): 1
__21____
(1): pozycja: 2 kolory: 2,3
(2): 3
_321____
(1): pozycja: 1 kolory: 1,2
(2): 1
1321____
(1): pozycja: 5 kolory: 1,2
(2): 2
13212___
(1): pozycja: 7 kolory: 1,2
(2): 1
13212_1__

Końcowy rezultat: Gracz (1) wygrał!

7.2 Przykładowa gra - wygrana złego

Dane wejściowe:
- n = 6
- k = 3
- l = 3
- m = 2

Stan początkowy planszy:
_____
(1): pozycja: 3 kolory: 2,3
(2): 2
__2___
(1): pozycja: 4 kolory: 1,3
(2): 1
__21__
(1): pozycja: 2 kolory: 2,3
(2): 3
_321__
(1): pozycja: 1 kolory: 1,2
(2): 1
1321__
(1): pozycja: 5 kolory: 1,2
(2): 2
13212_
(1): pozycja: 6 kolory: 1,2
(2): 1
132121

Końcowy rezultat: Gracz (2) wygrał!

7.3 Statystyki na podstawie serii gier między komputerami

Dla czterech kombinacji poziomów graczy komputerowych (słaby-słaby, mocny-słaby, słaby-mocny, mocny-mocny) przeprowadziliśmy po 10000 gier w trybie testowym, z parametrami: n=8, k=3, m=3 i l=2. Oto obserwacje:

- W rozgrywce dwóch graczy słabych, wybierających swoje ruchy całkowicie losowo, gracz dobry wygrał 7059 razy, a zły 2941.
- W rozgrywce mocnego gracza dobrego (korzystającego ze strategii) ze słabym złym, gracz dobry wygrał wszystkie 10000 gier.
- Słaby gracz dobry z mocnym graczem złym wygrał 368 razy; gracz drugi wygrał 9632 razy.
- W wypadku obu graczy komputerowych grających strategicznie, gracz dobry ponownie wygrał 10000 razy.

Na tej podstawie można stwierdzić, że strategia gracza dobrego jest bardziej skuteczna od strategii gracza złego. Widać to nie tylko w przypadku gry mocnego gracza dobrego z mocnym złym; gracz zły, nawet jeśli ma plan gry i mierzy się z graczem dobrym grającym losowo, dalej może z nim przegrać.