Gry kombinatoryczne 2022
Gra Szemeredi'ego
Dokumentacja Teoretyczna

Aleksander Malinowski
Jakub Piwko
Malwina Wojewoda

marzec 2022

1 Treść gry

Dane wejściowe:

- liczba naturalna k
- liczba naturalna x

Na początku rozgrywki komputer losuje zbiór x liczb naturalnych X. Każdy gracz ma swój własny kolor. Ruch polega na wybraniu niepokolorowanej liczby ze zbioru X i pokolorowaniu jej swoim kolorem. Gracze ścigają się kto pierwszy ułoży monochromatyczny ciąg arytmetyczny o zadanej długości k. Gra człowiek kontra komputer.

2 Teoria

Aby zrozumieć problem kolorowania ciągów arytmetycznych w podzbiorach zbiorów liczb naturalnych, przyjrzeliśmy się Teorii Ramseya. Ten dział matematyki zajmuje się poszukiwaniem regularności, które pojawiają się w przypadkowych nieuporządkowanych obiektach, jeśli te zwiększają się. Szczególnie bliskie naszego problemu są prace matematyków: B. L. van Der Waerdena i E. Szemeredi'ego, których twierdzenia skupiają się na poszukiwaniu ciągów arytmetycznych w specyficznych podzbiorach zbioru liczb naturalnych.

Na początek sformułujmy Twierdzenie Ramseya. Najpierw jednak warto przedstawić wykorzystywaną w nim notację „strzałkową" Erdôsa i Rado: $n \to (l_1, \ldots, l_r)^k$, gdy dla każdego podziału $\binom{[n]}{k} = X_1 \cup \cdots \cup X_r$ istnieją $i \in [r]$ oraz $T \subseteq [n]$ mocy $|T| = l_i$ takie, że $\binom{T}{k} \subseteq X_i$.

Gdy $l_1 = \cdots = l_r$, to piszemy $n \to (l)^k_r$. Gdy $r = 2$ lub $k = 2$, to opuszczamy ten indeks.

Twierdzenie Ramseya Dla wszystkich $r, l_1, \ldots, l_r$ istnieje n takie, że $n \to (l_1, \ldots, l_r)$.

Wydaje się, że może przydać się nam także Twierdzenie Schura, więc je także sformułujemy, a następnie udowodnimy na podstawie wcześniej przywołanego Tw. Ramseya.

Twierdzenie Schura

Dla wszystkich r istnieje s takie, że dla każdego $\chi : [s] \to [r]$ istnieją $x, y, z \in [s]$ takie, że $\chi(x) = \chi(y) = \chi(z)$ i $x + y = z$.

Dowód:

Niech $s + 1 \to (3)_r$ (takie s istnieje na podstawie Tw. Ramsey'a). Dowolne kolorowanie $\chi : [s] \to [r]$ generuje kolorowanie χ' par zbioru $\{0, 1, \ldots, s\}$ w ten sposób, że $\chi'(ij) = \chi(|i - j|)$.

Na podstawie Tw. Ramseya istnieją $a, b, c \in \{0, 1, \ldots, s\}$ takie, że $\chi'(ab) = \chi'(ac) = \chi'(bc)$, a więc $\chi(|a - b|) = \chi(|a - c|) = \chi(|b - c|)$. Niech $a < b < c$. Wtedy $x = b - a, y = c - b$ i $z = c - a$ spełniają tezę twierdzenia.

Przydatne do rozwiązaniu problemu wydaje się być także następujące twierdzenie:

Twierdzenie van der Waerdena

Dla wszystkich k, l istnieje istnieje taka najmniejsza liczba $n = W(k, l) \in \mathbb{Z}_+$, że dla każdego $\chi : [n] \to [k]$ istnieje monochromatyczny ciąg arytmetyczny długości l.

Twierdzenie van der Waerdena można rozumieć tak, że dla każdego kolorowania skończonego zbioru liczb całkowitych dodatnich istnieje skończony, monochromatyczny ciąg arytmetyczny dowolnej długości.

Liczby $W(k, l)$ opisane w twierdzeniu nazywane są liczbami van der Waerdena. Ich poszukiwanie jest złożonym problemem i do tej pory dla kolorowania dwoma kolorami znane są wartości tylko dla k wynoszących 3, 4, 5 i 6. Natomiast dla liczb od 7 do 11 znane są jedynie dolne ograniczenia.

Dowód:

Dowód przeprowadzimy jedynie dla konkretnych przypadków liczb k i l, ponieważ w ogólności dowód jest długi i skomplikowany.

Dowód dla $k = 2$ i $l = 3$:

Pokażemy, że $n = W(2, 3)$ istnieje oraz jej ograniczenie górne to 325, a więc $n \leq 325$. Na początek zdefiniujmy funkcję $\chi : [n] \to [r]$. Wiadomo, że zbiór 5-elementowy można pokolorować na $2^5 = 32$ kolorów. Aby mieć gwarancję, że co najmniej 2 zbiory będą pokolorowane w ten sam sposób należy wziąć 33 zbiory 5-elementowe. Oznaczymy je jako $B_1 = \{1, 2, 3, 4, 5\}, \ldots, B_{33} = \{161, 162, 163, 164, 165\}$.

Załóżmy, że te tak samo pokolorowane bloki to B_i i B_j, gdzie $1 \leq i < j \leq 33$. Spójrzmy na $B_i = \{5i - 4, \ldots, 5i\}$. Z zasady szufladkowej Dirichleta, wśród liczb $5i - 4, 5i - 3, 5i - 2$ są dwie tego samego koloru, bez straty ogólności powiedzmy, że ten kolor oznaczony jest numerem 1, czyli $\chi(5i - a) = \chi(5i - b) = 1$, gdzie $2 \leq b < a \leq 4$.

Wówczas mamy 2 przypadki: jeśli $\chi(5i - b + (a - b)) = \chi(5i + a - 2b) = 1$ to mamy już monochromatyczny ciąg arytmetyczny długości 3: $5i - a$, $5i - b$ $5i + a - 2b$, co kończy dowód.

Rozważmy więc przeciwny przypadek, kiedy $\chi(5i + a - 2b) = 2$. Wiemy, że skoro bloki B_i i B_j są pokolorowane tak samo to $\chi(5i - a) = \chi(5j - a) = \chi(5i - b) = \chi(5j - b) = 1$ oraz $\chi(5i + a - 2b) = \chi(5j + a - 2b) = 2$

Rozważmy $5i - a$, $5j - b$, które są tego samego koloru ($\chi(5i - a) = \chi(5j - b) = 1$).

Różnica między tymi liczbami wynosi $5j - b - (5i - a) = 5j - 5i + a - b$, a więc kolejna liczba w tym ciągu to $5j - b + (5j - 5i + a - b) = 10j - 5i + a - 2b$.
Rozważmy też $5i + a - 2b$, $5j + a - 2b$, które także są tego samego koloru, innego niż poprzednio ($\chi(5i + a - 2b) = \chi(5j + a - 2b) = 2$). Różnica między tymi liczbami wynosi $5j-5i$, a więc kolejna liczba w tym ciągu to $5j+a-2b+(5j-5i) = 10j-5i+a-2b$.

Zauważmy, że z wcześniejszego warunku na liczby i, j, a oraz b wynika, że ograniczenie górne na n to 325, ponieważ $10j - 5i + a - 2b \leq 330 - 5 + 4 - 2 \times 2 = 325$.
Widać więc, że jeśli $\chi(10j - 5i + a - 2b) = 1$ to otrzymamy monochromatyczny ciąg arytmetyczny długości 3 pierwszego koloru, którego wyrazami są $5i - a$, $5j - b$, $10j-5i+a-2b$, a jeśli $\chi(10j-5i+a-2b) = 2$ to otrzymamy monochromatyczny ciąg arytmetyczny długości 3 drugiego koloru, którego wyrazami są $5i+a-2b$, $5j+a-2b$, $10j - 5i + a - 2b$, co kończy dowód.

Twierdzenie Van Der Weardena było prekursorem do sformułowania Twierdzenia Szemeredi'ego, które dotyczy innego rodzaju podzbiorów $\mathbb{N}$. Zanim podamy to twierdzenie, podamy definicję asymptotycznej gęstości zbioru.

Definicja: asymptotyczna gęstość zbioru
Niech A stanowi podzbiór $\mathbb{Z}_+$. Asymptotyczną gęstością górną zbioru A nazywamy liczbę $\bar{d}$, zdefiniowaną następująco:

$$\limsup_{n\to\infty} \frac{|A \cap \{1, ..., n\}|}{n}$$

Gęstość zbioru można interpretować jak prawdopodobieństwo trafienia liczby ze zbioru A spośród zbioru liczb naturalnych. Łatwo zauważyć, że zbiory skończone posiadają gęstość równą 0. Z kolei jeśli gęstość zbioru jest niezerowa, to zbiór jest nieskończony i przeliczalny. Jako przykład możemy rozważyć zbiór $A = \{2n : n \in \mathbb{N}\}$, czyli zbiór liczb parzystych. Dla tego zbioru granica z definicji przy $n \to \infty$ wynosi $\frac{1}{2}$. Jest to dość intuicyjne, bo można naiwnie powiedzieć, że liczby parzyste stanowią połowę liczb naturalnych, czyli z prawdopodobieństwem $\frac{1}{2}$ możemy wylosować z $\mathbb{N}$ liczbę parzystą.

Twierdzenie Szemeredi'ego
Każdy podzbiór $\mathbb{Z}_+$ o niezerowej asymptotycznej gęstości górnej zawiera dowolnie długie, skończone ciągi arytmetyczne.

Twierdzenie to określa bardzo ciekawe własności nieskończonych podzbiorów liczb naturalnych. Jednak nasza gra, ze względu na obecność parametru x, który użytkownik na początku wybiera, będzie operowała jedynie na zbiorach skończonych. Bardzo możliwa będzie więc sytuacja, w której w wylosowanym zbiorze nie wystąpi ciąg arytmetyczny o długości k. Dlatego tak kluczowe w naszej aplikacji będzie sprawdzanie na każdym etapie gry, czy jest możliwe ułożenie szukanego ciągu. Jako że nasze liczby są losowe, nie będziemy mogli skorzystać bezpośrednio z twierdzenie van der Waerdena.

3 Strategia komputera

Komputer w naszej grze będzie musiał wykazywać się umiejętnością przewidywania i oceny aktualnej sytuacji. Jako że rozgrywka odbywa się między maszyną a człowiekiem, a ruchy kolorowania liczb wykonywane są naprzemiennie, komputer będzie musiał operować dwiema strategiami - ofensywną i defensywną. **Strategia ofensywna** - będzie naturalnie wybierana przez komputer w przypadku gdy to do niego będzie należeć pierwszy ruch. Głównym celem jest jak najszybsze pokolorowanie swojego ciągu arytmetycznego. **Strategia defensywna** - komputer obierze ją od razu, gdy przeciwnik człowiek będzie pierwszy wybierał liczbę do kolorowania. W takim przypadku komputer będzie starał się zablokować możliwości powiększania ciągu rywala. Na bazie wybranych już liczb będzie przewidywał najbardziej optymalny ruch człowieka, tak aby uniemożliwić mu zwycięstwo.

Zacznijmy jednak od początku gry. Gracz wybiera długość ciągu n oraz długość szukanego ciągu k. Jako, że charakter naszej gry jest losowy, może zdarzyć się, że rozgrywka przy wylosowanym zbiorze liczb nie będzie możliwa, chociażby przez brak w zbiorze X ciągu arytmetycznego długości k. W takim przypadku losowany będzie nowy zbiór, aż do momentu, kiedy taki ciąg będzie w nim istniał. To, jak sprawdzamy czy ciąg danej długości istnieje opisane jest dalej, w sekcji 3.1: Szukanie ciągów arytmetycznych długości k w zbiorze X. W celu zapewnienia, że ciągi takie będą istnieć możemy rozważyć ucięcie zbioru liczb naturalnych zależnie od podanych na wejściu parametrów. Na przykład, możemy wykorzystać parametr x, aby wylosować liczby ze zbioru $\{1, 2, \ldots, 10x-1, 10x\}$, co zwiększy szanse na wystąpienie w zbiorze X ciągów arytmetycznych długości k. Liczby w wylosowanym zbiorze będą unikalne.

Jeśli mamy już wyznaczone wszystkie występujące ciągi wróćmy do strategii gry. W ramach naszej aplikacji będzie znajdowała się specjalna klasa *MySet*. Przechowywać będzie wybrany ciąg, jego różnicę, długość, a także zbiory liczb z tego ciągu, które komputer i gracz już pokolorowali. Każdemu znalezionemu wcześniej ciągowi będzie odpowiadał jeden obiekt tej klasy, gdyż jego pola posłużą nam w określaniu kolejnych ruchów.

Początkowo, bez względu czy pierwszy ruch należy do człowieka czy nie, komputer szuka jaka liczba znajduje się w największej liczbie znalezionych ciągów arytmetycznych, czyli przykładowo, jeśli zostały znalezione ciągi:$[1, 2, 3]$, $[3, 6, 9]$, $[3, 12, 21]$ to tą liczbą jest 3. Koloruje ją, ponieważ ma szanse na ułożenie z nią kilku ciągów, czyli nawet jeśli gracz zablokuje ułożenie jednego, to może jeszcze kolorować inne.
Jeśli występuje sytuacja, że dwie liczby występują tyle samo razy, to wybierana jest losowa spośród nich.

Jeśli w żadnych dwóch ciągach nie występuje ta sama liczba, to szukany jest znaleziony ciąg arytmetyczny o największej długości i z niego wybierany środkowa liczba (lub w przypadku parzystej długości ciągu pierwsza z środkowych), czyli np. mamy ciągi $[1, 2, 3]$, $[8, 12, 16, 20]$, $[30, 50, 70]$ to zostanie wybrana liczba 12.

Planujemy posługiwać się pewną miarą, która wskazuje na postępy graczy. Załóżmy, że szukamy 8-elementowych ciągów. Przykładowo, komputer ma 4-elementowy ciąg w swoim kolorze, natomiast człowiek 6-elementowy, czyli człowiek jest bliżej zwycięstwa. Taką miarą postępu może być liczba ruchów potrzebna w tej chwili do najszybszego pokolorowania takiego ciągu, czyli w tym przypadku dla komputera to

4 ruchy, a dla gracza 2 ruchy. Inna sytuacja to gdy komputer ma w swoim kolorze jeden ciąg 5-liczbowy, ale nie ma już możliwości przedłużenia go (jest zablokowany), ale ma też ciąg 4-liczbowy, do którego ze zbioru niepokolorowanych liczb można dobrać 4 liczby, aby wygrać, czyli brakuje jeszcze 4 ruchów.

I tak oto dzięki połączeniu miar oraz możliwości zmiany strategii, komputer może dobierać swoje ruchy w danej chwili, sprawdzając po każdym ruchu, który gracz jest bliżej zwycięstwa. Gdy jest w defensywie, ale zarejestruje brak postępów u rywala, może przejść do ofensywy. I odwrotnie, gdy komputer będzie budował swój ciąg, ale odnotuje, że człowiek pokolorował już dłuższy ciąg, może zacząć blokować ruchy przeciwnika. W momencie, kiedy gracz zostanie zablokowany, ciąg, który próbował ułożyć znika z listy możliwych ciągów arytmetycznych.

Blokujące ruchy w trakcie gry mogą też prowadzić do wyczerpania możliwości zbudowania takich ciągów. Taka ewentualność kończyłaby rozgrywkę z wynikiem remisu, jako że żaden z zawodników nie będzie w stanie zrealizować zadania i zostanie wyświetlony odpowiedni komunikat, gra się zakończy.

Warunek, czy wystąpił remis będzie sprawdzany po każdym wykonanym ruchu. Oprócz tego, po każdym ruchu, w obiektach klasy *MySet*, gdzie w ciągu arytmetycznym występuje właśnie wybrana liczba będzie aktualizowane to, że została ona pokolorowana przez gracza, który wykonywał ten ruch. Następnie będzie aktualizowana miara postępu gracza (czyli tego, ile najmniej pozostało mu po pokolorowania całego ciągu). Jeśli wybrana liczba była w ciągu, który do tej pory miał najwięcej pokolorowanych liczb, oznaczamy, że gracz jest o 1 bliżej do zwycięstwa. Jeśli był na innej z list, nie dzieje się nic. Natomiast jeśli był to ruch blokujący przeciwnika, zmniejszamy miarę postępu przeciwnika.

Dodatkowo, przypadku ruchu gracza, najpierw będzie sprawdzane, czy taka liczba występuje w wylosowanym ciągu oraz czy można ją pokolorować (czy nie została wcześniej pokolorowana przez niego samego lub komputer).

Jeśli po ruchu gracza okaże się, że jego miara postępu do zwycięstwa jest równa 0, to gra się zakończy, wyświetlając komunikat, który gracz wygrał i jaki ciąg pokolorował

3.1 Szukanie ciągów arytmetycznych długości k w zbiorze X

Naszym pomysłem, jak sprawdzić istnienie ciągu arytmetycznego jest policzenie wartości bezwzględnej z różnicy między każdymi dwiema liczbami. Przykładowo, dla ciągu: 3 12 54 32 90 76 43 60 71 33 30 55 4 89 27 62 91
wyglądałoby to następująco:

-	3	12	54	32	90	76	43	60	71	33	30	55	4	89	27	62	91
3		9	51	29	87	73	40	57	68	30	27	52	1	86	24	59	88
12			42	20	78	64	31	48	59	21	18	43	8	77	15	50	79
54				22	36	22	11	6	17	21	24	1	50	35	27	8	38
32					58	44	11	28	39	1	2	23	28	57	5	30	59
90						14	47	30	19	57	60	35	86	1	63	28	1
76							33	16	5	43	46	21	72	13	49	14	15
43								17	28	10	13	12	39	46	16	19	48
60									11	27	30	5	56	29	33	2	31
71										38	41	16	67	18	44	9	20
33											3	22	29	56	6	29	58
30												25	26	59	3	32	61
55													51	34	28	7	36
4														85	23	58	87
89															62	27	2
27																35	64
62																	29
91																	

Następnie policzymy jakie różnice się powtarzają więcej niż *k-1* razy, czyli takie, które mają szanse utworzyć ciąg arytmetyczny zadanej długości. Będziemy mieć dla każdej różnicy listę par liczb (a, b), gdzie $a < b$, między którymi ona zachodzi.
W dalszej kolejności sprawdzimy, czy liczby dla konkretnej różnicy układają się w ciąg arytmetyczny. Będziemy to sprawdzać dodając do liczby b z każdej kolejnej pary rozważaną różnicę i sprawdzając, czy taka liczba znajduje się w którejś z par jako liczba a. Wówczas sprawdzamy, czy dla liczby b z tej pary istnieje gdzieś kolejna w ciągu liczba a. Jeśli uda nam się utworzyć ciąg co najmniej o zadanej długości k to zapisujemy go na specjalną listę. W ten sposób otrzymamy listę wszystkich ciągów arytmetycznych, co najmniej wymaganej długości k, które znajdują się w wygenerowanym zbiorze. Posłuży ona nam do decydowania które liczby będzie kolorował komputer oraz do sprawdzania, czy gracz koloruje któryś z ciągów i ewentualnego blokowania, o czym wcześniej wspomnieliśmy.