Kombinatoryczna teoria liczb

Szemerédi online

Dokumentacja końcowa

Autorzy
Mateusz Bieńkowski
Kornel Żaba
Jakub Mazurkiewicz

Problem

Szemerédi online

Gra człowiek kontra komputer, gdzie człowiek to gracz pierwszy.
Dane wejściowe:

- długość ciągu arytmetycznego k
- liczba dostępnych kolorów c
- maksymalna długość rozgrywki l

W każdej turze:

- pierwszy gracz podaje liczbę naturalną
- drugi gracz koloruje podaną liczbę jednym z dostępnych kolorów,

Zwycięstwo:

- pierwszy gracz wygrywa, jeżeli wśród pokolorowanych liczb istnieje monochromatyczny ciąg arytmetyczny długości co najmniej k (tzn. ciąg postaci $i, i+j,\ i+2j, ..., i+(k-1)j$, którego wszystkie elementy są pokolorowane tym samym kolorem)
- drugi gracz wygrywa, gdy pokolorowanych jest l liczb, a taki ciąg się nie pojawił.

Głównym zagadnieniem, które zostanie zbadane w trakcie testowania aplikacji, będzie opracowanie odpowiedniej strategii (lub kilku różnych strategii, ze względu na brak wiedzy apriori jakie podejście z punktu widzenia gracza 2 będzie dla niego najlepsze) dla gracza drugiego, a przy tym także sprawdzenie warunków wystarczających do zagwarantowania wygranej gracza pierwszego (parametry $k,\ c$ i l).

W celu automatyzacji testów niezbędne będzie opracowanie również strategii dla gracza 1.

Zmiany w założeniach

W porównaniu do dokumentacji teoretycznej z 05.05.2019, zmienił się sposób uruchamiania aplikacji. Aplikacja konsolowa pobiera od zera do dwóch argumentów:

- w przypadku *zera argumentów* - aplikacja zapyta użytkownika o wartości parametrów k, c i l oraz strategie dla graczy 1 i 2. Użytkownik ma również możliwość zastosowania manualnej strategii (ręczne wykonywanie ruchów gracza). Po podaniu parametrów zostaje przeprowadzona pojedyncza rozgrywka.
  Dostępne strategie:
    - advanced - Strategia brute force
    - linear - strategia rosnąca
    - random - strategia losowego wyboru
    - manual - ręczna
- w przypadku jednego argumentu (ścieżki do pliku konfiguracyjnego) - komputer automatycznie rozegra między sobą grę według parametrów sprecyzowanych w

pliku. Wynik gry zostanie wyświetlony w konsoli oraz zapisany do pliku o nazwie "*test_res.csv*".
- w przypadku dwóch argumentów (ścieżki do pliku konfiguracyjnego oraz ścieżki do pliku wyjściowego) - testy zostaną przeprowadzone analogicznie do przypadku jednego argumentu, jednak wyniki zostaną zapisane w podanym pliku oraz na konsoli.

Uruchomienie aplikacji

Aby uruchomić aplikację należy w katalogu głównym(zawierającym plik *CombinatorialTheoryOfNumbers.AppDotNet.exe*) z poziomu wiersza poleceń wykonać polecenie wzorowane na poniższych przykładach.

```
.\CombinatorialTheoryOfNumbers.AppDotNet.exe sampletest.json
```

Przykładowe uruchomienie, testy będą wczytane z pliku "sampletest.json", a wyniki zostaną zapisane do pliku "test_res.csv"

```
.\CombinatorialTheoryOfNumbers.AppDotNet.exe sampletest.json test_results.csv
```

Przykładowe uruchomienie z podaniem ścieżki do pliku z wynikami, testy będą wczytane z pliku "sampletest.json", a wyniki zostaną zapisane do pliku "test_results.csv"

```
.\CombinatorialTheoryOfNumbers.AppDotNet.exe
```

Przykładowe uruchomienie bez parametrów, po takim uruchomieniu aplikacja zapyta o potrzebne informacje. W przypadku wprowadzenia błędnych danych np. niepoprawny format liczby lub błędnie podana nazwa strategii aplikacja zakończy działanie błędem.

Plik konfiguracyjny należy umieścić w katalogu zawierającym aplikację.

Opis pliku konfiguracyjnego

Plik konfiguracyjny jest zapisany w formacie JSON. W pliku zapisana jest tablica, w której elementami jest konfiguracja jednego testu. Konfiguracja każdej gry składa się z następujących parametrów:

- Presenter - sposób wyświetlania rozgrywki (long - wybrana liczba:numer koloru, colored - wybrana liczba pokolorowana w konsoli, empty - brak wyświetlania)
- Player1
  - Type - strategia gry gracza 1 (random - strategia losowa, linear - strategia rosnąca, advanced - strategia brute force)
  - Seed - ziarno losowości dla strategii losowej
- Player2
  - Type - strategia gry gracza 2 (random - strategia losowa, linear - strategia rosnąca, advanced - strategia brute force)
  - Seed - ziarno losowości dla strategii losowej

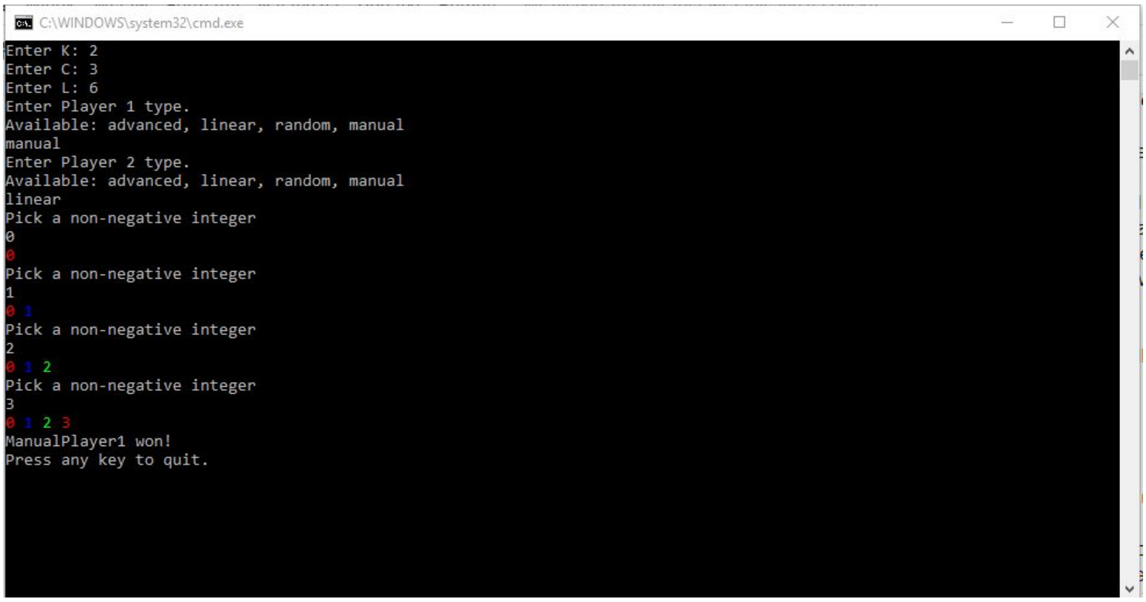
- K - docelowa długość pokolorowanego ciągu
- C - liczba kolorów
- L - długość rozgrywki
- Iterations - ile razy dana gra ma zostać powtórzona

Do aplikacji jest dołączony przykładowy plik konfiguracyjny *sampletest.json*.

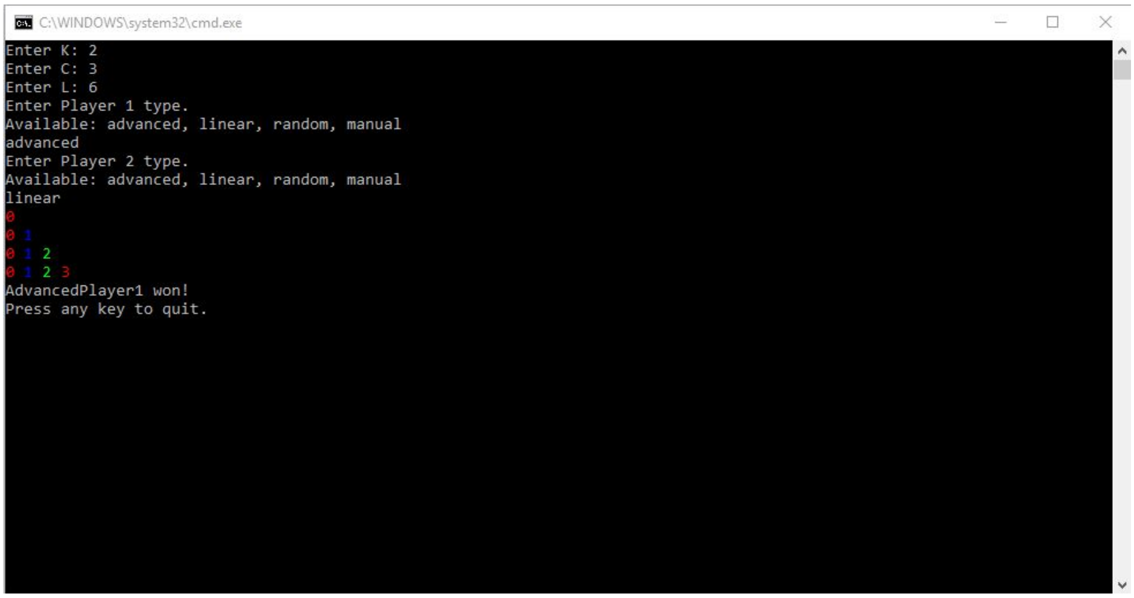
Zawartość pliku konfiguracyjnego może wyglądać następująco:

```
[
 {
  "Presenter": "long",
  "Player1": { "Type": "Linear" },
  "Player2": { "Type": "Linear" },
  "K": 6,
  "C": 4,
  "L": 30,
  "Iterations": 5
 },
 {
  "Presenter": "empty",
  "Player1": { "Type": "Linear" },
  "Player2": { "Type": "Advanced" },
  "K": 6,
  "C": 4,
  "L": 30,
  "Iterations": 5
 },
 {
  "Presenter": "empty",
  "Player1": { "Type": "Linear" },
  "Player2": { "Type": "Random" },
  "K": 6,
  "C": 4,
  "L": 30,
  "Iterations": 5
 }
]
```

Poniżej jest załączony przebieg przykładowej rozgrywki z ludzkim(manual) graczem 1.



Poniżej jest załączony przebieg przykładowej rozgrywki z 2 komputerami, graczem 1 (advanced) i graczem 2 (linear).



Opis strategii

Zostaną zrealizowane trzy strategie dla gracza 2 oraz trzy symetryczne strategie dla gracza 1. Strategie dla gracza 1, który docelowo jest sterowany przez człowieka, zostaną zaimplementowane w celu uproszczenia procesu testowania algorytmów.

Niech ZCK - zbiór zbiorów ciągów arytmetycznych danego koloru
$ZCKcr$ - zbiór ciągów arytmetycznych koloru c o różnicy r. Zakładamy, że zbiór $ZCKcr$ zawiera ciąg arytmetyczny długości 0 oznaczany jako $ZCKcr0$.
$ZCKcri$ - $i-ty$ element(ciąg arytmetyczny) zbioru $ZCKcr$

Strategie gracza 1

Strategia losowa

Gracz 1 w swojej turze wybiera liczbę od 1 do l.

Strategia rosnąca

Gracz 1 w każdej turze wybiera najmniejszą niewybraną liczbę naturalną.

Strategia brute force

Gracz 1 analizuje już istniejące ciągi arytmetyczne i stara się dobierać takie liczby naturalne, w przypadku których pokolorowanie możliwie największą liczbą kolorów powoduje przedłużenie jednego z już istniejących ciągów.

Kroki strategii brute force

1. Znajdź dwa ciągi $ZCKcri, ZCKcrj$ ze zbioru $ZCKcr \in ZCK$ (w szczególnym przypadku mogą być to ciągi długości 0 takie że:
    a. Ciąg $ZCKcri, x, ZCKcrj$ jest arytmetyczny z różnicą r dla pewnej liczby x
    b. Ciągi $ZCKcri, ZCKcrj$ są najdłuższe
2. Wybierz liczbę x.

Strategie gracza 2

Strategia losowa

Gracz 2 w swojej turze wybiera jeden z c dostępnych kolorów w losowy sposób.

Strategia *"kolejnych kolorów"*

W kolejnych ruchach gracz 2 (komputer) wybiera kolejne indeksy kolorów.
Przykład:
$c = 3$
Aktualny kolor (ak)
w poprzedniej turze został wybrany kolor $ak = 1$, w związku z tym w aktualnej zostanie wybrany kolor $ak = 2$. Kolory są indeksowane od **0** dlatego największy indeks koloru wynosi $c - 1$, w związku z tym wybranym kolorem w kolejnej rundzie będzie kolor $ak = 0$.
Ogólnie jeśli ak **wynosi** $c - 1$ to $ak = 0$, w przeciwnym wypadku $ak = ak + 1$.

Strategia brute force

Gracz 2 znajduje kolejny potencjalny ciąg arytmetyczny (tj. najbliższy ukończenia), który byłby rozszerzony przez pokolorowanie liczby wybranej przez gracza 1 na ten sam kolor, na który pokolorowany jest znaleziony ciąg. Gracz 2 wtedy koloruje znaleziony element na inny kolor. Jeśli znalezione zostanie więcej niż 1 taki ciąg, to gracz 2 koloruje rozszerzającą liczbę na kolor inny niż kolory znalezionych ciągów, jeśli to możliwe.

Kroki strategii brute force

Strategia ta jest stworzona podobnie do strategii brute force gracza pierwszego z tą różnicą, że ma ona zapobiegać powstawaniu najdłuższego ciągu.

1. Dla danej liczby x, znajdź dwa ciągi $ZCKcri, ZCKcrj$ ze zbioru $ZCKcr \in ZCK$ (w szczególnym przypadku mogą być to ciągi długości 0) takie że:
    a. Ciąg $ZCKcri, x, ZCKcrj$ jest arytmetyczny z różnicą r
    b. Ciągi $ZCKcri, ZCKcrj$ są najdłuższe
2. Wybierz losowo kolor k, który, jeśli to możliwe, nie jest jednym z kolorów na który były pokolorowane ciągi $ZCKcri$ z poprzedniego punktu.

Najbardziej kosztownym obliczeniowo elementem jest budowa, utrzymywanie i przeglądanie zbiorów zbioru ZCK w celu znalezienia odpowiednich ciągów dla strategii brute force obu graczy.

Rozważania teoretyczne

Twierdzenie Szemerédiego

Głosi ono, że dla każdego *δ*∈(0,1] oraz dowolnej liczby naturalnej *K* istnieje liczba *N(δ,K)* taka, że jeżeli *N* > *N(δ,K)*, to w dowolnym podzbiorze *{1, 2, …, N}* rozmiaru nie mniejszego niż *δN* istnieje ciąg arytmetyczny długości *K*.

Lemat 1. Bezradność *gracza 2*

Istnieją zestawy parametrów wejściowych c, k, l gry "*Szemerédi online*" dla których jeśli *gracz 1* wybiera tylko liczby ze zbioru $L = \{1, 2, 3, \ldots, l\}$, to bez względu na strategie obu graczy *gracz 1* kończy grę jako zwycięzca po l turach.

Dowód:

1. Zauważmy, że długość rozgrywki l musi być nie mniejsza niż $k * c$, aby zapewnić że jakiegoś koloru jest co najmniej k.
   $l/c >= k$
2. Z twierdzenia Szemerédiego wiemy, że w każdym podzbiorze długości większej niż δN zbioru $\{1, 2, 3, \ldots, N\}$, gdzie $N > N(\delta, k)$ istnieje ciąg arytmetyczny długości k. Wiedząc, że po l turach wszystkie liczby ze zbioru L będą pokolorowane, ponieważ z założenia gracz 1 wybiera niepokolorowane liczby ze zbioru L

Niech, $l = N$

Aby skorzystać z twierdzenia Szemerédiego, należy zapewnić, że w zbiorze jest pomalowane co najmniej δl liczb na jeden kolor.

$$\delta l = l/c \Rightarrow \delta = 1/c$$

stosując powyższą zależność do twierdzenia otrzymujemy, $l > N(1/c, k) = M$.

Dla danego c oraz k bierzemy dowolne $l > M$, tak dobrane l zapewnia to że gracz 1 zawsze wygrywa maksymalnie po l turach.

Lemat 2. Niepokonana strategia *gracza 2*

W grze "*Szemerédi online"*, jeżeli zestaw parametrów wejściowych c, k, l spełnia następującą nierówność $(k-1) * c >= l$, to istnieje co najmniej jedna niepokonana strategia *gracza 2.*

Dowód:

Niech gracz 2 korzysta ze strategii "kolejnych kolorów", to wtedy w najlepszym dla gracza 1 przypadku na koniec gry "plansza" będzie zawierała ciąg arytmetyczny jakiegoś koloru o długości co najwyżej $k-1$.

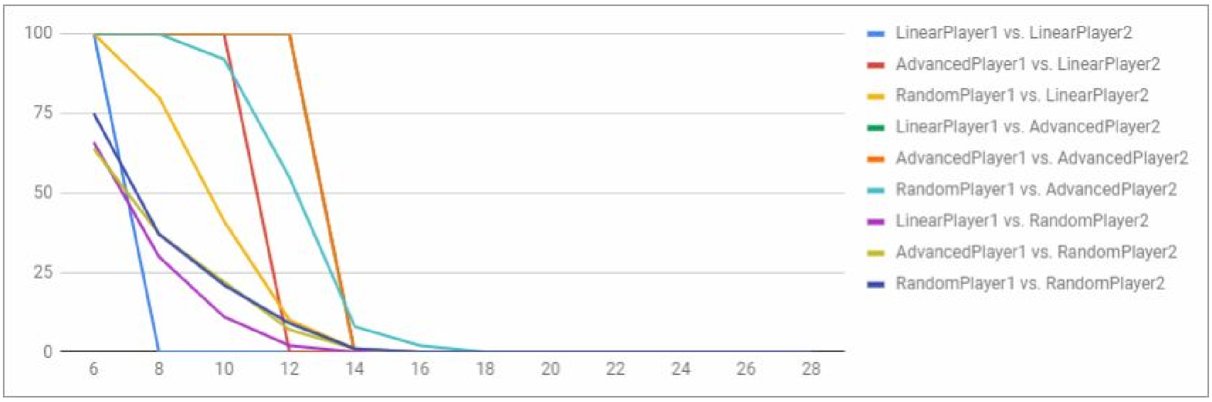
Testy

Został przygotowany zestaw konfiguracji testów, które miały na celu praktyczne sprawdzenie poprawności wyżej przedstawionych lematów.

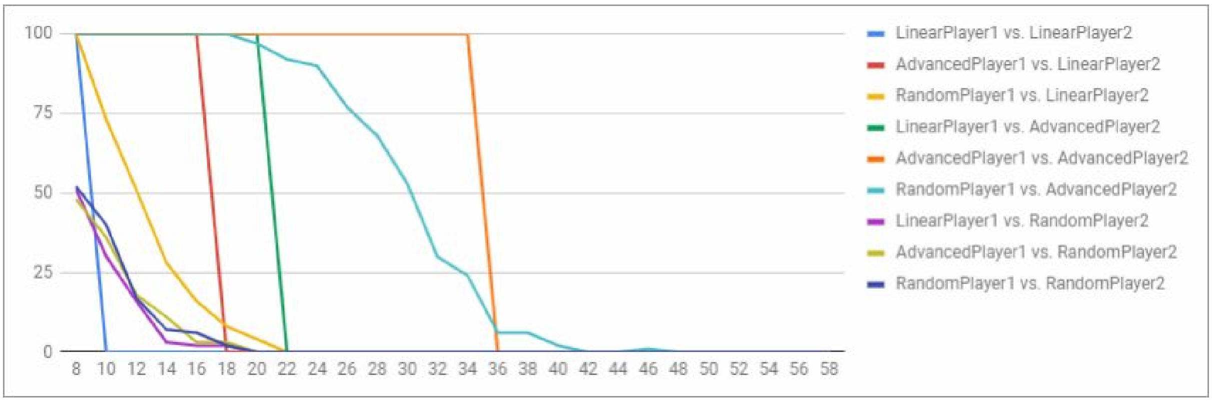
Wynik testów dla lematu o *“bezradności gracza 2”*

Poniższe wykresy przedstawiają liczbę zwycięstw gracza 2 dla różnych ustawień parametrów oraz strategii graczy.
W każdym z 3 zestawów testów zastosowano ustawione wartości parametrów K (długość ciągu) i C (liczba kolorów), natomiast manipulowano parametrem L (maksymalna długość rozgrywki), któremu nadawano coraz większe wartości aż do osiągnięcia liczby van der Waerdena N(C, K). Dla czytelności zakres danych przedstawiony na każdym wykresie został ograniczony ze względu na powtarzające się wyniki po osiągnięciu pewnego L (gracz 2 przestaje w ogóle wygrywać).

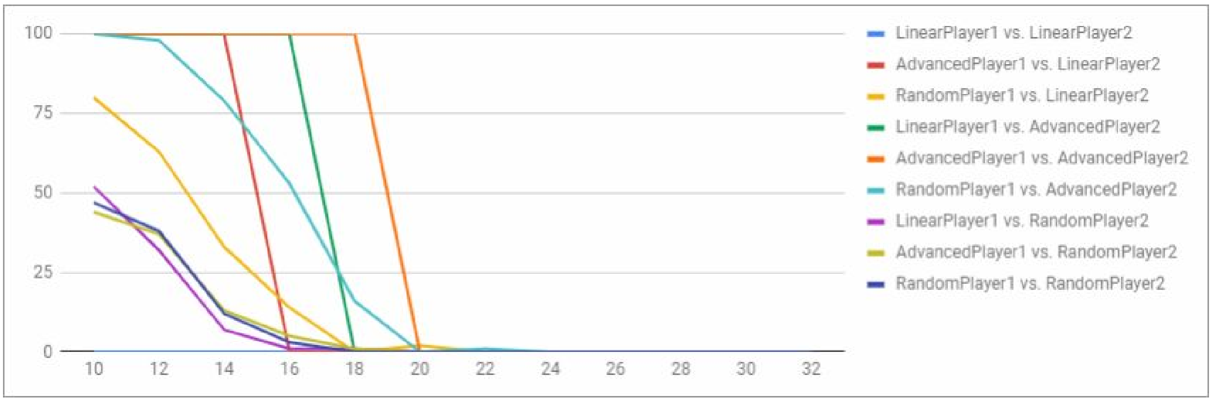
Dla każdego zestawu parametrów K, C i L przeprowadzono testy dla 9 par strategii po 100 powtórzeń dla każdej pary.



Test 1.
K = 4, C = 2, max. L = 36



Test 2.
K = 3, C = 4, max. L = 80



Test 3.
K = 5, C = 2, max. L = 180

Wynik testów dla lematu o *"niepokonanej strategii gracza 2"*

Tabele przedstawiają liczbę wygranych dla gracza danego typu. Nazwa kolumny wskazuje na strategię gracza. (Przedrostek wskazuje na strategię np. Advanced - strategia brute force, a przyrostek wskazuję na rodzaj gracza np. 1 - gracz pierwszy). Kolumna "Row Labels" opisuje odpowiednio strategię gracza 2 oraz parametr L (długość rozgrywki).
Każdy gracz 2 (gracz 2 o konretnej strategii) grał z każdym graczem 1 (gracz 1 o konkretnej strategii).

Tabele prezentują zagregowane wyniki, agregacja nastąpiła po strategii gracza 2 oraz parametrze l, tak jak zostało to już wspomniane.

Row Labels	**AdvancedPlayer1**	**AdvancedPlayer2**	**LinearPlayer1**	**LinearPlayer2**	**RandomPlayer1**	**RandomPlayer2**	**Grand Total**
Advanced		**180**					**180**
11		60					60
12		60					60
13		60					60
Linear			**20**	**159**			**179**
11				60			60
12				59			59
13			20	40			60
Random	**14**		**10**		**15**	**141**	**180**
11	3		3		6	48	60

12	5		3		3	49	60
13	6		4		6	44	60
Grand Total	**14**	**180**	**30**	**159**	**15**	**141**	**539**

Tabela 1.

Parametry k oraz c dla powyższej tabeli wynoszą odpowiednio 4, 4. Graniczna wartość dla parametru L zgodnie z lematem 2 wynosi 12.

Row Labels	**Advanced Player1**	**Advanced Player2**	**LinearPlayer1**	**LinearPlayer2**	**RandomPlayer1**	**RandomPlayer2**	**Grand Total**
Advanced		**180**					**180**
19		60					60
20		60					60
21		60					60
Linear			**20**	**159**			**179**
19				59			59
20				60			60
21			20	40			60
Random	**7**		**4**		**8**	**161**	**180**
19	1		1		3	55	60
20	2		1		1	56	60
21	4		2		4	50	60
Grand Total	**7**	**180**	**24**	**159**	**8**	**161**	**539**

Tabela 2.

Parametry k oraz c dla powyższej tabeli wynoszą odpowiednio 5, 5. Graniczna wartość parametru L wynosi 20.

Wyniki prezentowane w *Tabela 2* są bardzo podobne do wyników w *Tabela 1.*

Row Labels	*AdvancedPlayer2*	*LinearPlayer1*	*LinearPlayer2*	*RandomPlayer2*	*Grand Total*
Advanced	***180***				***180***
40	*60*				*60*
45	*60*				*60*
50	*60*				*60*
Linear		***20***	***160***		***180***
40			*60*		*60*
45			*60*		*60*
50		*20*	*40*		*60*
Random				***180***	***180***
40				*60*	*60*
45				*60*	*60*
50				*60*	*60*
Grand Total	***180***	***20***	***160***	***180***	***540***

Tabela 3.

Parametry k oraz c dla powyższej tabeli wynoszą odpowiednio 10, 5. Graniczna wartość parametru L wynosi 45.

Row Labels	AdvancedPlayer2	LinearPlayer1	LinearPlayer2	RandomPlayer2	Grand Total
Advanced	**360**				**360**
240	120				120
250	120				120
260	120				120

Linear		**40**	**319**		**359**
240			119		119
250			120		120
260		40	80		120
Random				**360**	**360**
240				120	120
250				120	120
260				120	120
Grand Total	**360**	**40**	**319**	**360**	**1079**

Tabela 4.

Parametry k oraz c dla powyższej tabeli wynoszą odpowiednio 26, 10. Graniczna wartość parametru L wynosi 250.

Wyniki potwierdzają lemat 2. Z powyższych tabel można łatwo zauważyć, że zarówno strategia liniowa (rosnąca) oraz zaawansowana (brute force) gracza 2 są strategiami niepokonanymi dla parametrów spełniających nierówność z lematu 2. ($(k-1)*c >= l$).

Pliki testowe

Testy lematu 1 są zapisane w formacie .json w następujących plikach

- *lemat1test_k4c2.json*
- *lemat1test_k3c4.json*
- *lemat1test_k5c2.json*

Testy lematu 2 są zapisane w formacie .json w następujących plikach

- *lemat2test_small.json*
- *lemat2test_medium.json*
- *lemat2test_large.json*