Raport końcowy

Piotr Gołębiewski, Michał Jaruga, Michał Gromadzki

19 marca 2023

1 Instrukcja obsługi

Aplikacja została stworzona w programie Python i wszystkie rozgrywki naszej gry sa właśnie wyświetlane w konsoli tego programu. Każda rozgrywka rozpoczyna się od podania następujących wartości w podanej kolejności:

1. $n \in \mathbb{N}$ - długość planszy,
2. $k \in \mathbb{N}$ - długość wygrywającego monochromatycznego ciągu arytmetycznego,
3. kto zaczyna grę? (1 - gracz, 2 - komputer)
4. strategia komputera: 1 - losowy, 2 - defensywny, 3 - ofensywny, 4 - zoptymalizowany, 5 - zmodyfikowany. W przypadku strategii zmodyfikowanej dodatkowo:
   (a) α - współczynnik ofensywności strategii komputera (dla strategii zoptymalizowanej: 1)
   (b) β - współczynnik defensywności strategii komputera (dla strategii zoptymalizowanej: 1)

Dla estetycznego wyświetlania planszy, zachęcamy aby liczba n była nie większa niż 35, ale oczywiście dla większych wartości gra również działa.

Po wpisaniu wartości wstępnych rozpoczyna się właściwa rozgrywka. Zaczyna użytkownik, którego wskazaliśmy w trzecim pytaniu. Jeśli ruch wykonuje gracz, to program prosi o podanie numeru wyrazu ciągu i, który chcemy pokolorować. Oczywiście $i \in \{1, 2, ...n\}$. W przypadku podania niepoprawnej wartości (indeks i jest za mały, za duży lub jest pozycją zajętą), program ponownie spyta o wartość i. Następnie ruch wykonuje komputer zgodnie z typem strategii, który wskazaliśmy w pytaniu czwartym.

Rozgrywka trwa do momentu, gdy zostanie ułożony ciąg monochromatyczny o zadanej długości k lub do przypadku, gdy nikt nie wygrał - remisu.

2 Zmiany względem poprzednich wersji

- Stworzyliśmy pięć strategii dla komputera. Są to strategie: losowa, defensywna, ofensywna, zoptymalizowana i zmodyfikowana. W pierwotnej wersji projektu były tylko trzy pierwsze.
- Początkowo w konsoli nie wyświetlały się różne kolory dla użytkowników, lecz w wersji finalnej już tak.
- Po wygranej użytkownika, program na samym końcu wyświetla jakim ciągiem wygrywającym osiągnięto zwycięstwo.

3 Implementacja

W programie stworzyliśmy następujące funkcje:

1. `find_arytm_sqnc_all(idx, k)`
   - Funkcja zwraca listę list dwuelementowych, każda mniejsza lista odpowiada jednemu ciągowi arytmetycznemu o długości k, który można ułożyć z liczb zawartych w liście liczb idx. Pierwszy element odpowiada liczbie, od której zaczyna się ciąg, a drugi reprezentuje jego różnicę. Funkcja ta będzie wykorzystywana przy wyznaczaniu optymalnego ruchu komputera.

2. `find_arytm_sqnc(idx, k)`

   - Funkcja zwraca prawdę, jeśli w liście liczb idx istnieje ciąg arytmetyczny o długości k. W przeciwnym wypadku zwraca fałsz. Funkcja ta jest wykorzystywana w funkcji `check_if_win`.

3. `check_if_win(Board, k)`

   - Funkcja sprawdza warunki zwycięstwa przez użytkowników, tj. bada, czy istnieje ciąg wygrywający. Wypisuje, ktory użytkownik wygrał oraz wypisuje jakim ciągiem osiągnięto zwycięstwo. W przypadku remisu, wyświetla komunikat o grze zakończonej remisem.

4. `player_turn(Board)`

   - Funkcja prosi gracza o podanie indeksu i do pokolorowania pola i pobiera tę wartość. W przypadku podania niepoprawnej wartości, program ponownie pyta. Jeżeli podano poprawną wartość, program zajmuje podaną pozycję.

5. `pc_random_turn(Board)`

   - Funkcja ta odpowiada za wybór ruchu przez komputer w strategii losowej. Komputer do losowania rozpatruje pozycje na planszy niezajęte ani przez gracza, ani przez siebie.

6. `pc_opt_turn(Board, k, alpha, beta)`

   - Funkcja odpowiada za wykonanie przez komputer optymalnego ruchu. Działanie funkcji można opisać w 3 krokach. Najpierw ocenia każdy możliwy do ułożenia planszy ciąg, potem wybiera te najlepiej ocenione, a na końcu wybiera liczbę, która najczęściej w nich występuje, oczywiście o ile nie jest ona jeszcze pokolorowana.
     Ciągi oceniane są w następujący sposób: każdy ciąg należy do dokładnie jednego z 4 następujących przypadków:
     (a) Jego część jest pokolorowana przez gracza
     (b) Jego część jest pokolorowana przez komputer
     (c) Jest pokolorowany zarówno przez gracza, jak i komputer
     (d) Jest niepokolorowany przez żadnego z graczy

     Wówczas te cztery sytuacje implikują kolejno, że:
     (a) Ciąg otrzymuje βp_G punktów, gdzie p_G to liczba pokolorowanych przez gracza liczb w tym ciagu.
     (b) Ciąg otrzymuje αp_K punktów, gdzie p_K to liczba pokolorowanych przez komputer liczb w tym ciągu.
     (c) Ciąg otrzymuje -1 punktów - jest wykluczony.
     (d) Ciąg otrzymuje 0 punktów.

     Następnie zostają wybrane te ciągi, które otrzymały jak najwięcej punktów. Na koniec sprawdzamy, która wsród jeszcze niepokolorowanych liczb występuje w tych ciągach najczęściej - jest to optymalnych ruch dla komputera.
     Dodatkowo przed rozpoczęciem wyboru optymalnego ruchu funkcja sprawdza czy nie istnieje oczywisty ruch, który należy wykonać w danej pozycji. Istnieją 2 takie przypadki (kolejność ma znaczenie):
     (a) Komputer ma już ułożony ciąg o długości $k-1$ wtedy oczywistym ruchem jest pokolorowanie ostatniej liczby i wygranie gry, oczywiście o ile to możliwe.
     (b) Gracz ma już ułożony ciąg o długości $k-1$, wtedy oczywistym ruchem jest pokolorowanie takiej liczby, aby uniemożliwić graczowi zakończenie rozgrywki w następnym ruchu.

7. `game()`

   - Funkcja łącząca pozostałe funkcje w całość. Odpowiada za kolejność, w jakiej użytkownicy grają, którą strategią gra komputer oraz za wyświetlanie planszy.

8. `print_board(Board)`

   - Funkcja wyświetla planszę: pokazuje które pozycje są zajęte przez gracza w kolorze niebieskim i komputer w kolorze czerwonym oraz pozycje dostępne.

9. `preproc()`

   - Funkcja, której celem jest zadanie pytań użytkownikowi na początku rozgrywki i pobranie danych, tj. wartości n i k, kto zaczyna rozgrywkę oraz jaki typ strategii dla komputera (w tym dla strategii zmodyfikowanej wartości α i β).

4 Przykładowa rozgrywka

```
Podaj n: 12
Podaj k: 3
Wybierz kto zaczyna: 1 - gracz, 2 - komputer: 1
Wybierz tryb gry komputera: 1 - losowy, 2 - defensywny, 3 - ofensywny, 4 - zoptymalizowany,
5 - zmodyfikowany: 4
------
Wybiera Gracz. Podaj liczbę do pokolorowania: 7
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   |   |   |   |   | G |   |   |    |    |    |
------
Komputer wybiera pozycję 5.
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   |   |   | K |   | G |   |   |    |    |    |
------
Wybiera Gracz. Podaj liczbę do pokolorowania: 3
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   | G |   | K |   | G |   |   |    |    |    |
------
Komputer wybiera pozycję 11.
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   | G |   | K |   | G |   |   |    | K  |    |
------
Wybiera Gracz. Podaj liczbę do pokolorowania: 8
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   | G |   | K |   | G | G |   |    | K  |    |
------
Komputer wybiera pozycję 6.
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   | G |   | K | K | G | G |   |    | K  |    |
------
Wybiera Gracz. Podaj liczbę do pokolorowania: 4
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
  |   | G | G | K | K | G | G |   |    | K  |    |
------
Komputer wybiera pozycję 1.
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
K |   | G | G | K | K | G | G |   |    | K  |    |
------
Komputer wygrał!
Wygrywający ciąg to: 1 6 11
```

5 Testy aplikacji i wnioski

Po przeprowadzeniu wiele rozgrywek, doszliśmy do wniosku, że najskuteczniejszym dla komputera typem strategii jest strategia zbalansowana, czyli taka w której współczynnik ofensywności α i defensywności β są sobie równe. Najgorszą jest oczywiście strategia losowa - tutaj komputer właściwie nie posiada żadnej sensownej strategii, a jedynie wybiera pozycje w sposób całkowicie losowy. Mimo to, pozostałe strategie również są skuteczne:

- Strategia defensywna komputera jest skuteczna, kiedy gracz również gra defensywnie, tj. najbardziej skupia się na tym, aby komputer nie ułożył ciągu wygrywającego. Odwrotnie, gdy gracz gra ofensywnie, strategia defensywna komputera może nie być najelpszą strategią.
- Strategia ofensywna komputera - analogicznie jak poprzednio - jest skuteczna, gdy gracz również gra ofensywnie, tj. zależy mu najbardziej na ułożeniu ciągu wygrywającego oraz nieskuteczna, gdy gracz gra defensywnie.
- Strategia zmodyfikowana komputera, czyli taka, w której strategie ofensywna i defensywna są połączone w stosunku $\alpha : \beta$, jest najbardziej skuteczna, gdy gracz również podążą taką strategią (czyli upraszczając, na $\alpha+\beta$ ruchów, α ruchów jest ofensywnych i β defensywnych).

Ogółem, komputer ma największe szanse na zwycięstwo, gdy naśladuje strategie gracza. Oczywiście nie możemy z góry założyć, ile gracz wykona ruchów ofensywnych, a ile defensywnych. Poza tym, nie zawsze można łatwo zdecydować, czy konkrenty ruch gracza był bardziej defensywny czy ofensywny - mógł być zarówno taki i taki. Dlatego najbezpieczniej przyjąć opcję, że gracz gra optymalnie, czyli w stosunku $1 : 1$. Stąd strategia zbalansowana okazuje się być najlepszą strategią na zwycięstwo komputera.

Użytkownik, który zaczyna rozgrywkę ma przeważnie większą szanse na wygraną (o ile jest możliwa dla podanych wartości n i k), w szczególności dla małych wartości k, np. $k = 3$. W przypadku $k = 1, 2$ oczywiście pierwszy użytkownik zawsze wygrywa, bo jedna liczba i dwie liczby zawsze tworzą ciąg arytmetyczny.

Zauważyliśmy też, że zaczęcie rozgrywki od wyboru pozycji bliżej środka planszy zamiast bliżej brzegów zwiększa szanse wygranej. Jest tak, ponieważ im jesteśmy bliżej środka, to tym więcej mamy możliwych pozycji do wybrania na lewo i prawo do utworzenia ciągu.

6 Podsumowanie

Na koniec wróćmy do twierdzenia Van der Waerden'a, czyli twierdzenia głoszącego, że dla dowolnej liczby graczy istnieje plansza, dla której dla każdej możliwej rozgrywki ktoryś gracz wygra. Niestety jest to twierdzenie czysto egzystencjalne, tzn. mówi, że tak zachodzi, ale nie dostajemy konretnej mininalnej długości planszy n, a jedynie pewne - dosyć grube - ograniczenie górne. Przykładowo dla dwóch graczy i długości ciągu 3, minimalną planszą jest plansza długości 9, a twierdzenie mówi nam, że $W(2, 3) < 325$. O ile dla $n = 9$ możemy "na palcach" zbadać wszystkie rozgrywki i stwierdzić, że to właśnie jest szukana wartość $W(2, 3)$, tak dla większych możemy napotkać problemy, nawet przy użyciu komputera. Widzimy zatem, że musimy poszukać innych rozwiązań, by znaleźć dokładną wartość. Poniżej przedstawiamy parę znanych wartości liczby $W(2, k)$, gdzie k to długość ciągu wygrywającego:

$W(2, 3)$	9
$W(2, 4)$	35
$W(2, 5)$	178
$W(2, 6)$	1132

Dla większych wartości k nie są znane dokładne wyniki liczby W, a jedynie ograniczenia górne.

Postawiliśmy jeszcze sobie na koniec pytanie, w jaki sposób możnaby rozbudować naszą grę. Sądzimy, że ciekawym trybem gry byłaby rozgrywka dla 3 użytkowników (dwoje graczy i jeden komputer lub jeden gracz i dwa komputery) oraz dla większej liczby użytkowników. Dodatkowo każdy z komputerów mógłby grać inną strategią, np. pierwszy grałby defensywnie, a drugi ofensywnie. Wprowadzilibyśmy różne poziomy trudności, rozgrywkę na czas i wiele urozmaiceń. Ze zmian estetycznych, to z pewnością spróbowalibyśmy stworzyć aplikację z naszego kodu z pełnym interfejsem i wyborem kolorów. Projekt ten pozwolił nam przypomnieć i poszerzyć wiedzę z zakresu programu Python.