O matematycznej złożoności i praktycznej implementacji gry Sprouts

Michał Dobranowski
I LO im. Bartłomieja Nowodworskiego w Krakowie

27 lutego 2021

Streszczenie

Sprouts to gra wymyślona w latach sześćdziesiątych XX wieku przez J. Conwaya oraz M. Patersona na University of Cambridge. Chcieli oni stworzyć grę, która ma jak najprostsze zasady i jednocześnie jest bardzo trudna w analizie. Celem mojej pracy jest udowodnienie podstawowych faktów, pokazanie, na czym polega złożoność tej gry oraz przedstawienie niezbędnych podstaw matematycznych i algorytmicznych, które mają zachęcić do próby stworzenia własnej implementacji gry na systemy komputerowe.

Słownik

Słownik zawiera użyte w dalszej części pojęcia wykraczające poza zakres matematyki szkolnej, których wprowadzenie jest kluczowe dla zrozumienia całości.

Graf – zbiór wierzchołków połączonych krawędziami w dany sposób.

Graf planarny – graf, który można przedstawić na płaszczyźnie w taki sposób, że jego krawędzie się nie przecinają.

Graf spójny – taki graf, że wszystkie wierzchołki łączy ciąg krawędzi (żaden nie jest odseparowany).

Stopień wierzchołka – liczba krawędzi, które wychodzą z danego wierzchołka grafu.

Cykl – taki ciąg krawędzi, że jego koniec jest identyczny z początkiem.

Algorytm przeszukiwania wszerz – takie przechodzenie przez graf od zadanego wierzchołka, że wierzchołki leżące bliżej (pod względem liczby krawędzi do przejścia) zostaną odwiedzone wcześniej.

Funkcja Catmulla-Roma – specjalny przypadek funkcji sklejanej (splajnu), co znaczy, w uproszczeniu, funkcji określonej przedziałami w taki sposób, że każdy przedział jest funkcją wielomianową względnie niskiego stopnia. Jest często wykorzystywana w grafice komputerowej, dlatego też dla większości nowoczesnych języków programowania istnieją biblioteki ją implementujące.

1 Zasady

Grę zaczyna się od n kropek. Każdy ruch polega na narysowaniu **linii łączącej dwie kropki** (możliwa jest też pętelka na jednej kropce) oraz **kolejnej kropki** na tej linii. Linia nie może przecinać innej linii oraz z każdej kropki mogą wychodzić co najwyżej trzy linie (pętelka liczy się jako dwie). Gracz, który nie może wykonać ruchu, przegrywa.

2 Liczba ruchów

Określmy, ile maksymalnie ruchów można wykonać w trakcie jednej gry, zaczynając od n kropek. Liczbę ruchów w grze oznaczmy jako m.

Każda kropka na początku ma trzy *życia*, to znaczy, że można do niej podłączyć trzy linie. W każdym ruchu zabieramy dwa życia, ale powstaje jedno nowe, czyli każdy ruch kosztuje dokładnie jedno życie. Po ostatnim ruchu gry zostaje nam $3n - m$ żyć. Wiemy, że

$$3n - m \geq 1$$

ponieważ w ostatnim ruchu tworzymy jedno życie, dodając kropkę. Z tej nierówności możemy już wyprowadzić

$$m \leq 3n - 1 \tag{1}$$

Czy istnieje w takim razie dolny limit liczby ruchów? Okazuje się, że tak, jednak zanim go znajdziemy, musimy zdefiniować jeszcze trzy pojęcia (nazwy zaczerpnąłem od R. Focardiego [1]).

sąsiad - kropka, którą można połączyć z daną kropką bez przecinania innych linii (to znaczy zachowując planarność),

ocaleńcy – kropki, które po ostatnim ruchu mają jeszcze jedno życie,

faryzeusze – kropki, które po ostatnim ruchu nie sąsiadują z żadnymi ocaleńcami.

Oznaczmy teraz liczbę ocaleńców jako o, liczbę martwych sąsiadów ocaleńców jako s oraz liczbę faryzeuszy jako f. Wiemy, że

$$o = 3n - m$$

Każdy ocaleniec ma dokładnie dwóch martwych sąsiadów, przy czym żadna martwa kropka nie może mieć dwóch ocalałych sąsiadów, inaczej można by było wykonać ruch i gra nie byłaby skończona. Z tego powodu mamy

$$s = 2o = 2(3n - m)$$

Ponadto oczywistym jest, że

$$f \geq 0$$

Całkowita liczba kropek na końcu gry jest równa

$$\begin{aligned} m + n &= o + s + f \\ m + n &\geq 3n - m + 2(3n - m) \\ m + n &\geq 8n - 4m \\ m &\geq 2n \end{aligned} \tag{2}$$

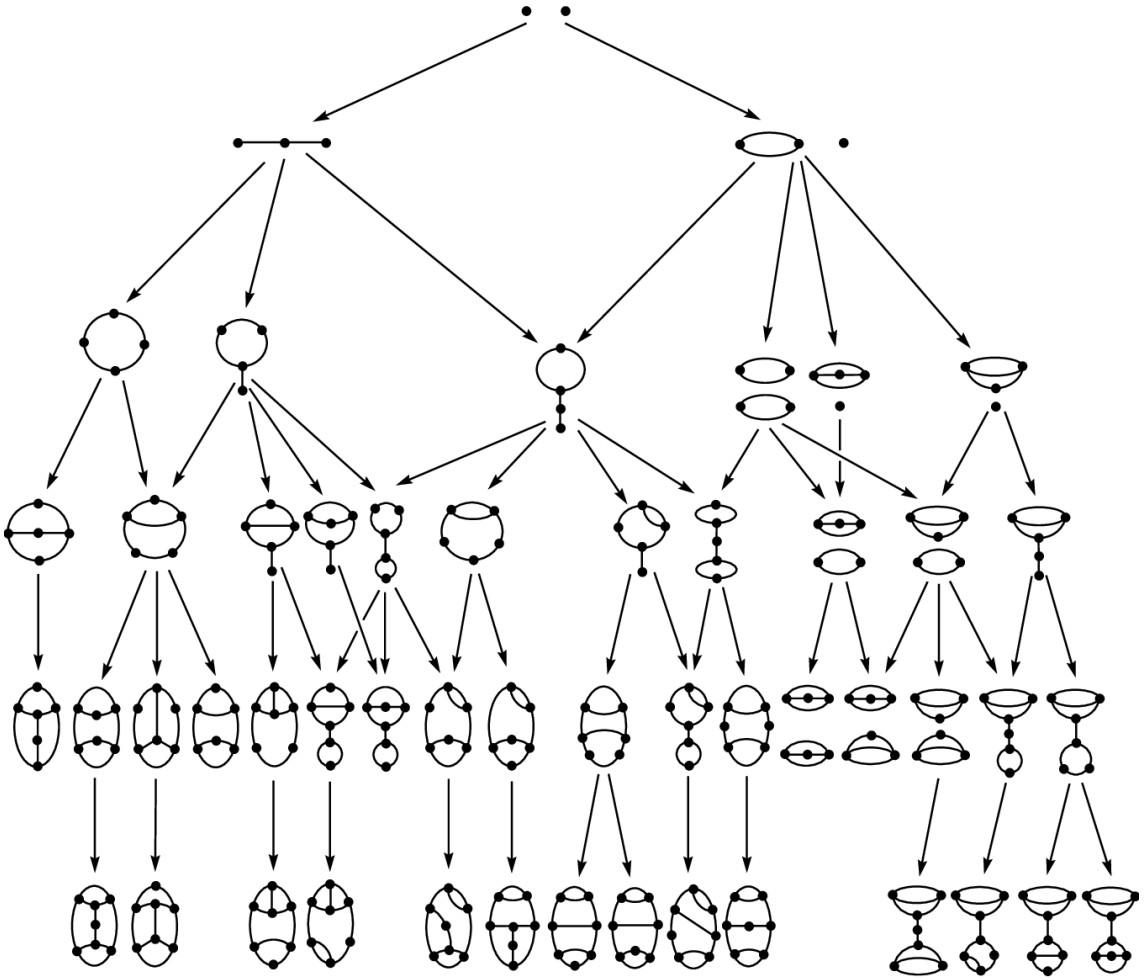
Na podstawie (1) oraz (2) możemy sformułować następujące twierdzenie:

Twierdzenie 1. *Dla n początkowych kropek, liczba ruchów m w każdej grze Sprouts spełnia nierówność* $2n \leq m \leq 3n - 1$.

3 Analiza gry dla dwóch kropek

Mogłoby się wydawać, że analiza gry dla dwóch kropek jest raczej trywialna, zwłaszcza, że gra musi się skończyć w 4 lub 5 ruchach, ale okazuje się, że rozpatrzenie wszystkich przypadków jest raczej czasochłonne.

Po narysowaniu całego drzewka możliwości (patrz: Rysunek 1) dochodzimy do dosyć ciekawych wniosków. Gracz, który zaczyna pierwszy, ma 14 możliwości wygranych, a gracz drugi tylko 5. Jednak to drugi gracz jest na zwycięskiej pozycji i, jeśli tylko zapamięta całe drzewko możliwości, ma pewność wygranej niezależnie od ruchów pierwszego gracza.



Rysunek 1: Analiza gry dla $n = 2$. Źródło: [2]

4 Analiza gry dla większej ilości kropek

Analiza gry dla $n = 2$ nie jest czymś trywialnie prostym, ale zdecydowanie możliwym do wykonania. Gdzie w takim razie leży obecnie granica w możliwościach analitycznych *Sprouts*?

W wyniku zakładu między J. Conwayem a D. Mollisonem, ten drugi, w ciągu miesiąca, tworzył 47-stronicową analizę gry sześciokropkowej. Powstały też oczywiście analizy dla gier $n = 3, 4, 5$. Jak pokazuje Tabela 1, za każdym razem dla któregoś z graczy istniała strategia wygrywająca.

n	1	2	3	4	5	6
gracz A			✓	✓	✓	
gracz B	✓	✓				✓

Tabela 1: Istnienie strategii wygrywającej dla gracza w n-kropkowej grze.

Hipoteza 1 (Sprouts Conjecture)**.** *Istnieje strategia wygrywająca dla pierwszego gracza wtedy i tylko wtedy, gdy liczba początkowych kropek n przystaje do 3, 4 lub 5 modulo 6. W przeciwnym razie istnieje strategia wygrywająca dla drugiego gracza.*

W latach dziewięćdziesiątych potwierdzono komputerowo [2] tę hipotezę dla $n \leq 11$, a następnie coraz większe n były uzupełniane przez otwarte oprogramowanie GLOP, aktualnie aż do $n = 44$.

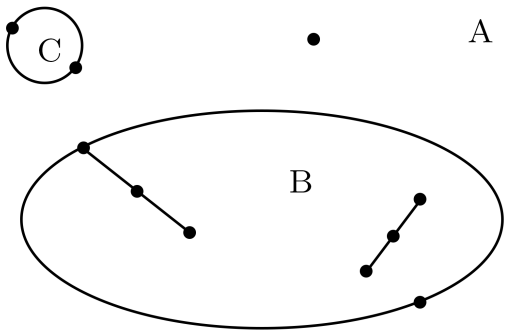
Można zauważyć, że mimo pewnej popularności tej gry w formie „papierowej", jedyne istniejące aplikacje umożliwiające granie w *Sprouts* są bardzo niedopracowanymi projektami, którym brakuje podstaw matematycznych do wizualizacji rozgrywki. Nie istnieje również żaden silnik umożliwiający zagranie idealnej gry dla chociażby niezbyt wielkiej liczby kropek. Jak pokażemy w kolejnej sekcji, ten drugi problem jest skutkiem wykładniczo wzrastającej złożoności obliczeniowej oraz trudności interpretowania wyników.

5 Reprezentacja rozgrywki

Każda rozgrywka może być reprezentowana na trzy powiązane ze sobą sposoby: jako graf, jako ciąg znaków lub w postaci kanonicznej. Sposób reprezentacji zaadaptuję z wcześniejszych prac: [2] oraz [3].

5.1 Postać grafowa

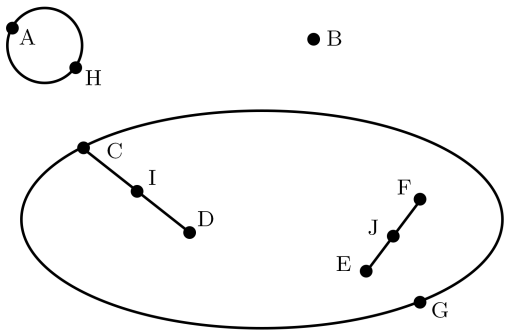
Każda pozycja w grze jest **grafem planarnym**, którego wierzchołki nie mogą mieć **stopnia** większego od 3. Cykle w grafie dzielą go na składowe, które nazwiemy **regionami**. Każdy region zawiera co najmniej jedną **granicę**, czyli po prostu spójny podgraf. Na przykładzie rysunku 2 mamy trzy regiony oznaczone dużymi literami. W regionie A mamy trzy granice, w B mamy dwie granice, a w C tylko jedną.



Rysunek 2: Przykład gry sześciokropkowej po 4 ruchach

5.2 Postać tekstowa

Niech każda kropka (wierzchołek) będzie oznaczona literami od A do Z. Każda granica jest ciągiem takich liter (w odpowiedniej kolejności oraz w taki sposób, aby po ścieżce wrócić do pierwszego wierzchołka nie powtarzając go). Każdy region jest zbiorem granic oddzielonych przecinkiem, a pozycja gry jest zbiorem regionów oddzielonych znakiem średnika.



Rysunek 3: Przykład gry sześciokropkowej z widoczną kolejnością ruchów

Stan gry zilustrowany na rysunku 3 można opisać w następujący sposób:

`AH,B,DICGCI;DICGCI,EJFJ;AH;`

Przetworzenie postaci tekstowej na postać kanoniczną polega na zastosowaniu ciągu optymalizacji, takich jak uogólnienie nieużytych jeszcze kropek do jednej litery, usunięcie martwych regionów i inne, które są niezwykle istotne dla tworzenia dobrych strategii gry i komputerowej analizy, jednak nie będziemy się tym zajmować. Przykłady takich analiz można znaleźć w sekcji **Literatura** pod [1, 4, 5].

5.3 Liczba możliwych otwarć

Rozróżnijmy najpierw dwa typy ruchów: wiążące i niewiążące[1]. Ruchy **wiążące** to te, które łączą dwie kropki różnych granic w zakresie jednego regionu, a **niewiążące** to takie, które łączą kropki w obrębie jednej granicy.

Mając grę o początkowych n kropkach jako pierwszy ruch możemy wykonać $\binom{n}{2} = \frac{n(n-1)}{2}$ ruchów wiążących dwie kropki oraz n ruchów niewiążących, z których każdy może okalać dowolny podzbiór $n-1$ kropek. Tak więc liczba ruchów otwierających dla gry n-kropkowej jest dana wzorem

$$O_n = \frac{n(n-1)}{2} + n \cdot 2^{n-1} \tag{3}$$

Już wzrost samych ruchów otwierających jest wykładniczy (dla $n = 15$ przekracza 200 milionów) nie mówiąc o skończeniu gry (znów dla $n = 15$ od 30 do 44 ruchów). Wiele z tych ruchów jest topologicznie jednakowych, co znaczy, że można je rozpatrywać jako jedną i tę samą rozgrywkę. Znacznie zmniejsza to liczbę ruchów otwierających, lecz rozpatrzenie *wszystkich możliwych scenariuszy* gry nadal ma wykładniczą złożoność obliczeniową.

6 Wizualizacja

Odpowiedź na pytanie, jak można stworzyć grywalną aplikację *Sprouts* niestety nie jest prosta z matematycznego punktu widzenia. Jednak, jak pokazał C. Browne

[1] Cameron Browne nazwał je w swojej pracy [3] odpowiednio *Double-Boundary* oraz *Single-Boundary Moves*.

[3], techniką, która może w tym pomóc, jest *triangulacja Delone*, czyli, w tym przypadku, podział płaszczyzny na takie trójkąty, że każde dwa z nich mają wspólny bok lub w ogóle nie mają żadnego punktu wspólnego oraz we wnętrzu okręgu opisanego na dowolnym z tych trójkątów nie ma wierzchołków.

Powiązany z triangulacją Delone jest *diagram Woronoja*, to znaczy taki podział płaszczyzny zawierającej n zaznaczonych punktów na komórki (obszary), że każda komórka zawiera jeden zaznaczony punkt oraz takie punkty płaszczyzny, które są położone bliżej tego zaznaczonego punktu, niż któregokolwiek innego z pozostałych $n-1$ zaznaczonych punktów. Środki okręgów opisanych na trójkątach w triangulacji Delone są wierzchołkami diagramu Woronoja.

6.1 Ruchy wiążące

Wizualizacja ruchu wiążącego przebiega następująco.

1. Znajdujemy najmniejszy zbiór trójkątów, które łączą dane dwie kropki. Można tego łatwo dokonać poprzez algorytm wyszukiwania wszerz.

2. Rysujemy krzywą łączącą te kropki. Jest to zdecydowanie najtrudniejsza część procesu, dlatego posłużę się tylko sugestią wcześniej wymienionego twórcy tego algorytmu, aby użyć funkcji sklejanej Catmulla-Roma.

3. Dodajemy nowy wierzchołek na środku krzywej (jego położenie nie ma znaczenia dla dalszej rozgrywki).

6.2 Ruchy niewiążące

Wizualizacja ruchu niewiążącego, który tworzy krzywą zamkniętą od punktu A wokół k punktów $B_1, \ldots, B_k$ przebiega następująco.

1. Zaznaczamy wszystkie krawędzie komórek, w których znajdują się punkty $B_1, \ldots, B_k$ w diagramie Woronoja oraz takie krawędzie, aby stworzyć spójną łamaną.

2. Bierzemy punkt A oraz środki wszystkich boków trójkątów, z którymi to bokami powyżej opisana łamana ma punkty wspólne. Te punkty będą punktami, do których dopasowana będzie wspomniana w poprzedniej sekcji funkcja sklejana Catmulla-Roma.

Po każdym ruchu należy oczywiście od nowa obliczyć diagram Woronoja i triangulację Delone dla wszystkich punktów.

7 Podsumowanie

Gra Sprouts jest bardzo trudną w analizie grą, do której, z pokazanych wcześniej powodów, stworzenie silnika, nawet opartego na technologiach sztucznej inteligencji, może okazać się zadaniem niemożliwym w najbliższych latach. Jednakże nawet

w ostatnim czasie pojawiło się kilka nowych metod i optymalizacji, które dają nadzieję na sukces w tej dziedzinie.

Liczę, że moja praca chociaż trochę przyczyni się do popularyzacji tak ciekawej z matematycznego punktu widzenia gry, a może nawet zachęci kogoś do stworzenia własnej implementacji, która będzie umożliwiać swobodne granie i opracowywanie nowych strategii.

Literatura

[1] R. Focardi and F. Luccio, "A new analysis technique for the sprouts game," 2001.

[2] D. Applegate, G. Jacobson, and D. Sleator, "Computer analysis of sprouts," 1991.

[3] C. Browne, "Algorithms for interactive sprouts," *Theoretical Computer Science*, vol. 644, pp. 29–42, 2016. Recent Advances in Computer Games.

[4] R. Focardi and F. Luccio, "A modular approach to sprouts," *Discrete Applied Mathematics*, vol. 144, no. 3, pp. 303–319, 2004.

[5] J. Lemoine and S. Viennot, "Computer analysis of sprouts with nimbers," 2010.