Warsaw University of Technology

Kombinatoryka na słowach

Dokumentacja teoretyczna

Głuszczak Weronika
Kołakowska Aleksandra
Litkowski Andrzej

8 kwietnia 2022

Spis treści

Wstęp

Celem tego projektu jest stworzenie gry w niepowtarzanie [1] rozgrywanej pomiędzy człowiekiem a komputerem. W grze w niepowtarzanie sprawdzamy czy w budowanym słowie nie występują repetycje, czyli dwa identyczne podsłowa, występujące jedno za drugim.

Niech alfabet A będzie zbiorem składającym się z symboli nazywanych literami. Zakładamy, że zbiór wszystkich liter składa się z 26 liter znajdujących się w angielskim alfabecie.

Niech $S = s_1 s_2 \dots s_n$ będzie ciągiem długości n, składającym się z liter z alfabetu A. Segmentem w S nazywamy podciąg składający się z kolejnych wyrazów ciągu S. Ciąg S nazywamy repetycją jeżeli składa się dwóch identycznych segmentów, czyli $n = 2k$ oraz $s_i = s_i + k$ dla każdego $i = 1, 2, \dots, k$. Mówimy, że ciąg S nie zawiera repetycji jeżeli żaden jego segment nie jest repetycją. Zakładamy, że każdy z identycznych segmentów musi być składać się z conajmniej 2 liter.

Dane wejściowe to ustalony alfabet A oraz liczba naturalna n, określająca maksymalną długość słowa podczas rozgrywki. Podczas gry bierzemy pod uwagę tylko repetycje nietrywialne - repetycje dłuższe niż powtórzenie jednej litery.

Człowiek rozpoczyna grę poprzez dopisanie jednej litery ze zbioru A, następnie komputer dopisuje kolejną literę. Pierwszy z Graczy stara się uniknąć nietrywialnej repetycji w budowanym słowie, podczas gdy drugi z Graczy stara się ją stworzyć. Gra może się zakończyć w dwóch przypadkach - musi wystąpić repetycja lub długość słowa musi być równa n.

1 Treść zadania - Gra w niepowtarzanie

W tej grze bierzemy pod uwagę tylko nietrywialne (dłuższe niż powtórzenie jednej litery typu 'aa') repetycje. Dane wejściowe:

- ustalony alfabet A
- liczba naturalna n

Gracze budują słowo składające się z liter z ustalonego alfabetu A. Po kolei dopisują po jednej literze na końcu słowa. Pierwszy gracz stara się uniknąć repetycji, drugi stara się ją stworzyć. Pierwszy gracz wygrywa jeżeli uda mu się uniknąć repetycji i słowo osiągnie długość n, w przeciwnym przypadku wygrywa gracz drugi.

2 Teoria

Poniżej znajdują się definicje wszystkich pojęć użytych w twierdzeniu 2.7 oraz jego dowodzie.

Definicja 2.1 (Graf) *Grafem nazywamy parę $G = (V, E)$, gdzie V jest zbiorem wierzchołków, a E jest zbiorem krawędzi $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$*

Definicja 2.2 (Drzewo) *Drzewem określamy graf nieskierowany dla którego zachodzi własność, że z każdego wierzchołka można dotrzeć do innego wierzchołka i można to zrobić tylko jednym sposobem. Każde drzewo jest acykliczne i spójne.*

Definicja 2.3 (Ścieżka) *Ścieżką nazywamy ciąg wierzchołków $(v_0, v_1, ..., v_n)$ łączący wierzchołek v_0 z wierzchołkiem v_n taki, że dla każdego $k \in 0, 1, ..., n$ istnieje krawędź z v_k do v_{k-1}.*

Definicja 2.4 (Ścieżka prosta) *Ścieżka w której w ciągu wierzchołków $(v_0, v_1, ..., v_n)$ każdy z wierzchołków występuje maksymalnie raz.*

Definicja 2.5 (Alfabet) *Alfabetem nazywamy zbiór liter alfabetu łacińskiego.*

Definicja 2.6 (Repetycja) *Repetycją nazywany sekwencję składającą się z dwóch identycznych bloków liter $x_1 \dots x_h x_1 \dots x_h$.*

Lemma 2.1 (Lemat Königa) *Jeżeli drzewo jest nieskończone, a każdy wierzchołek ma skończoną liczbę dzieci, to musi w tym drzewie istnieć nieskończona ścieżka prosta.*

Łatwo zauważyć, że każda binarna sekwencja składająca się z co najmniej 4 liter posiada repetycję. W 1906 Thue wykazał, że 3 symbole są wystarczające, aby utworzyć dowolnie długą sekwencję liter nie zawierającą repetycji [2]. Jego metoda jest konstruktywna i wykorzystuje podstawienia nad podanym zestawem symboli. Oznacza to, że zastąpienie wszystkich symboli w sekwencji bez repetycji przypisanymi blokami powoduje powstanie sekwencji, która nadal nie zawiera repetycji. Poniższe twierdzenie (oraz jego dowód) wykorzystuje zależność, jaką zauważył Thue, oraz pozwala opracować strategię do naszej gry.

Twierdzenie 2.7 *W grze w niepowtarzanie dla alfabetu składającego się z 6 liter, istnieje strategia dla Gracza starającego się uniknąć repetycji według której ciąg liter o dowolnej długości nie będzie zawierał repetycji o długości większej niż 1.*

Dowód. Załóżmy, że w grze w niepowtarzanie mamy dwóch graczy - Gracza N i Gracza C. Każdy z Graczy poczas swojej tury dodaje jedną literę do tworzonego ciągu. Gracz N stara się uniknąć repetycji, podczas gdy Gracz R stara się doprowadzić do repetycji w tworzonym ciągu. Gra kończy się zwycięstwem Gracza N gdy

ciąg osiągnie długość n lub zwycięstem Gracza R jeżeli w ciągu wystąpi repetycja. W tym dowodzie przez repetycję rozumiemy repetycję o rozmiarze większym niż 1.

Wybieramy dowolne n i zamierzamy udowodnić, że gracz starający się uniknąć repetycji ma strategię pozwalającą zbudować ciąg o rozmiarze n bez powtórzeń o rozmiarze większym niż 1. W dowodzie bierzemy pod uwagę losowość strategii Gracza N i pokazujemy, że dla każdej strategii Gracza R istnieje ewaluacja eksperymentów losowych prowadząca do wygenerowania ciągu bez rekurencji o długości n. Oznacza to, że Gracz R nie może mieć zwycięskiej strategii. Zatem, istnieje zwycięska strategia dla Gracza N. Wtedy, ponownie, przez zastosowanie Lematu Königa otrzymujemy treść twierdzenia.

Niech $s_1, ..., s_{m-1}$ będzie ciągiem liter stworzonym podczas $m-1$ tur gry. Załóżmy, że w następnej turze o numerze m Gracz N wybiera literę, którą chce dołączyć do tworzonego ciągu. Wiemy, że Gracz N zaczyna grę, więc na podstawie założeń wiemy, że m jest liczbą nieparzystą. Strategia wyboru dodawanej litery używana przez Gracza N składa się z następujących kroków.

1. Rozważ wszystkie litery z alfabetu A.
2. Wyeliminuj literę która została wybrana w turze $m-2$, a więc literę wybraną poprzednio przez Gracza N.
3. Jeżeli litera s_{m-1} wybrana przez Gracza R w poprzedniej turze jest taka sama jak litera s_{m-4} (wybrana dwie tury wcześniej przez Gracza N), wyeliminuj literę s_{m-3}.
4. Jeżeli w poprzednich krokach Gracz N wyeliminował tylko jedną literę, litera s_{m-4} powinna zostać wyeliminowana.
5. Losowo wybierz literę, która znajduje się w alfabecie A i nie została wyeliminowana w poprzednich krokach.

Ta strategia sprawia, że w grze nie występują repetycje o długości 2 i 3 liter. Możemy także za pomocą dowodu nie wprost pokazać, że używając tej strategii nie wystąpią repetycje o długości 4 liter.

Załóżmy, że w pewnym momencie gry pojawi się ciąg z przyrostkiem w postaci $x_1x_2x_3x_4x_1x_2x_3x_4$, gdzie $x_1x_2x_3x_4$ jest powtarzającym się ciągiem liter o długości 4. Załóżmy też, że Gracz N wybrał ostatnią literę w ciągu - x_4. Przywołując opisaną strategię, ostatnia litera dodana przez Gracza N to $s_m = x_4$ i jest to taka sama litera jak litera wybrana 4 tury wcześniej - $s_{m-4} = x_4$. Oznacza to, że punkt 4 strategii nie został wykonany, a zatem użyto zarówno punktu 2 i 3 opisanej strategii. W szczególności, x_3 musi się równać $x4$.

Oznacza to jednak, że w poprzednim ruchu Gracza N (gdy $x2$ zostało dodane do repetycje) symbole wykluczone przez w kroku 2 i kroku 3 były takie same. Oznacza to, że powinna była zostać wykonany krok 4. Użycie tej reguły wykluczyłoby dodanie x_2 do ciągu, co jest *sprzecznością*. Analogiczne rozumowanie działa w przypadku, gdy Gracz R dodaje ostatnią literę do repetycji o długości 4.

Ustalamy strategię dla Gracza R. Symulujemy grę między Graczem N, grającym losowo, a Graczem R grającym z tą ustalaną strategią. Jeżeli w m-tym ruchu (w prawdziwej, niesymulowanej grze) pojawi się repetycja o długości h, cofamy się do ruchu $m - h + 1$. Oznacza to, że usuwamy cały powtarzający się segment i ponownie rozpoczynamy symulację od ruchu $m - h + 1$ (z niezależnymi eksperymentami losowymi).

Ciąg ruchów to ciąg kolejnych symboli wybieranych przez graczy w symulacji. Zauważ, że nie jest możliwe, aby Gracz R wprowadził trzy symbole z rzędu w ciągu ruchów. W istocie, jeśli wprowadza dwa symbole z rzędu, to po pierwszym symbolu musiała nastąpić repetycja (o równej wielkości). W związku z tym drugi symbol jest taki sam jak symbol wymazany w tej pozycji (ponieważ strategia Gracza R jest stała w symulacji). Oznacza to, że drugi symbol nie mógł wygenerować repetycji, a zatem Gracz N kontynnuje rozgrywkę w symulacji.

Waga ciągu ruchów to liczba symboli wybranych przez Gracza N w tym ciągu. Ustalmy, że M jest wystarczająco duże. Pokażemy, że istnieje scenariusz, w którym pierwsze M eksperymentów losowych (pierwsze M ruchów Gracza N) powoduje, że symulacja zwraca ciąg o długości n. To udowodni, że Gracz N ma strategię budowania ciągu o długości n przeciwko ustalonej strategii Gracza R. Dla *sprzeczności* zakładamy, że wszystkie ciągi ruchów wygenerowane po M ruchach Gracza N w symulacji mają długość mniejszą niż n dla wszystkich możliwych ewaluacji eksperymentów losowych.

Wiemy, że ciąg ruchów o wadze M jest jednoznacznie określony przez ciąg M-wyborów Gracza N. Niech $r_1, ..., r_M$ będą symbolami wybranymi przez Gracza N. Gracza N zawsze wybiera jeden symbol spośród co najmniej $C-2$ symboli więc ciąg $r_1, ..., r_M$ ma co najmniej $(C-2)^M$ możliwych ewaluacji. Ciąg ruchów wywołany przez ewaluację $r_1, ..., r_M$ nazywany jest realizacją tej ewaluacji.

Niech h_j oznacza długość obecnego ciągu tuż przed krokiem (ruchem) symulacji który jest na pozycji j w tym ciągu. Ciąg h_j nazywamy *ciągiem wysokości*. Jeśli Gracz N doda literę w $j-$tym kroku, to jej następny ruch jest w kroku $k \in j+1, j+2, j+3$.

Istnieje tylko kilka możliwych ciągów wysokości od h_j do h_k:

(a) Gracz N nie powoduje repetycji w kroku j, a Gracz R nie powoduje reptycji w kroku $(j+1)$. W tym przypadku $k = j+2$ oraz $h_{j+1} = h_j + 1$, $h_{j+2} = h_j + 2$.

(b) W $j-$tym kroku Gracz N powoduje repetycje o nieparzystej długości, która jest większa bądź równa 5. Powoduje to, że Gracz N gra ponownie w kroku $(j+1)$. W tym przypadku $k = j+1$ oraz $h_k \leq h_j - 4$.

(c) W $j-$tym kroku Gracz N powoduje repetycje o parzystej długości, która jest większa bądź równa 6. Gracz R nie wykonuje żadnej repetycji w kroku $(j+1)$. W tym przypadku $k = j+2$ oraz $h_k \leq h_j - 4$.

(d) Gracz N nie powoduje żadnej repetycji w kroku j. Gracz R w kroku $(j+1)$ powoduje repetycję o parzystej długości, która jest większa bądź równa 6. W tym przypadku $k = j+2$ oraz $h_k \leq h_j - 4$.

(e) Gracz N nie powoduje repetycji w $j-$tym kroku. Gracz R w kroku $(j+1)$ powoduje repetycję o nieparzystej długości większą bądź równą 5. Następnie Gracz R nie powoduje żadnej repetycji w kroku $(j+2)$. W tym przypadku $k = j+3$ oraz $h_k \leq h_j - 2$.

Chcemy pozbyć się pewnej nadmiarowościw ciągu wysokości. Dokładniej, kodujemy ciąg wysokości na jego podciąg składający się z elementów h_j odpowiadających ruchom Gracza N oraz informacji o typie ciągu. Niech h_j, h_k będą ponownie wysokościami obecnego ciągu tuż przed dwoma kolejnymi ruchami Gracza N. Należy zauważyć, że

- Jeśli $h_k > h_j$, to ciąg wysokości między h_j i h_k jest typu (a).
- Jeśli $h_k = h_j - 2$, to ciąg wysokości między h_j i h_k jest typu (e).
- Jeśli $h_k \leq h_j - 4$, to ciąg wysokości między h_j i h_k jest typu (b), (c), (d), lub (e).

Zatem, aby zapisać cały ciąg wysokości, wystarczy zapamiętać podciąg wysokości $h_1', ..., h_M'$ odpowiadających ruchom Gracza N. Dodatkowo, jeśli $h'_{j+1} \leq h'_j - 4$, trzeba zapamiętać $typ(h'_j, h'_{j+1}) \in b, c, d, e$, czyli typ oryginalnego ciągu wysokości między symbolami odpowiadającymi h'_j i h'_{j+1}.

Na koniec zauważmy, że wszystkie elementy h_j są parzyste. Jest to spowodowane tym, że obecny ciąg przed ruchem Gracza N zawiera parzystą liczbę symboli. *Funkcję typu* nazywamy $typ(d_{j+1}) = typ(h_j, h_{j+1})$. Jeżeli funkcja typu jest zdefiniowana, określamy *zredukowany ciąg różnic* jako $d_1 = 1$, $d_{j+1} = (h'_{j+1} - h'_j)/2$ dla $1 \leq j \leq M$.

Zauważmy, że

(i) $d_j \leq 1$,

(ii) $\sum_{j=1}^{k} d_j \geq 1$, for all $1 \leq k \leq M$,

(iii) $typ(d_j)$ jest zdefiniowany wtedy i tylko wtedy gdy $d_j \leq -2$

Para $((D, typ), S)$ jest *rejestrem wyszukiwań*, jeśli istnieje ewaluacja $r_1, ..., r_M$ taka, że D jest zredukowanym ciągiem różnic w realizacji $r_1, ..., r_M$, typ jest funkcją typu dla D, a S jest ciągiem końcowym powstałym po M krokach Gracza N w tej realizacji procedury wyszukiwania ruchów.

Założenie 1. *Każdy rejestr wyszukiwań odpowiada niepowtarzalnej ewaluacji ciągu* $r_1, ..., r_M$.

Dowód. Zakładamy, że mamy rejestr wyszukiwań $((D, typD), S)$, gdzie $S = (s_1, ..., s_l)$, dekodujemy ewaluację ciągu $r_1, ..., r_M$ w kilku krokach. Najpierw wyodrębniamy ciąg wysokości $h_1, ..., h_m$ z (D, typ_D) i uzupełniamy go o element $h_m + 1 = |S|$. Teraz, możemy opisać jak zrekonstruować ciąg $x_1, ..., x_m$ wszystkich symboli wprowadzonych w symulacji. Wykonujemy to w odwrotnej kolejności, tzn. najpierw dekodujemy x_m, a następnie ciąg S_{m-1} utworzony po $m-1$ krokach symulacji. Następnie za pomocą prostej iteracji wyodrębniamy wszystkie pozostałe symbole $x_{m-1}, ..., x_1$.

- Jeżeli $h_{m+1} - h_m = 1$, to wprowadzenie x_m do ciągu nie spowodowało repetycji. Zatem x_m jest ostatnim symbolem w ciągu końcowym S, tzn. $x_m = s_l$ oraz $S_{m-1} = (s_1, ..., s_{l-1})$.
- Jeżeli $h_{m+1} - h_m \leq 0$, to niektóre symbole zostały usunięte po wprowadzeniu x_m. Znamy jednak rozmiar repetycji, czyli $h = |h_{m+1} - h_m| + 1$, oraz wiemy, że tylko połowa repetycji została skasowana. To oznacza, że możemy skopiować odpowiedni fragment, aby przywrócić $s_{m-1} = (s_1, ..., s_l, s_{l-h+1}, ..., s_{l-1})$ oraz $x_m = s_l$.

Po zrekonstruowaniu wszystkich elementów ciągu, $x_1, ..., x_m$, odczytujemy ciąg od początku i obserwujemy obecny ciąg w symulacji. Za każdym razem, gdy obecny ciąg ma parzystą długość, następny symbol jest wprowadzany przez Gracza N.

Możemy zdefiniować *typowaną ścieżkę wyszukiwania* jako parę $((d_1, ..., d_M), typ)$ spełniającą reguły (i), (ii), (iii) oraz dodatkowo warunek $\sum_{j=1}^{M} = 1$. Niech T_M będzie liczbą ścieżek wyszukiwania o długości M (tzn. D ma długość M). Przy naszym założeniu, że Gracz N nigdy nie wygrywa, każdy wykonalny ciąg różnic w typowanej ścieżce wyszukiwania sumuje się do mniej niż n. Liczba typowanych ścieżek wyszukiwania o długości M spełniających podpunkty (i), (ii), (iii) z sumą całkowitą k (stałe $k > 1$) wynosi $O(T_M)$.

Z tego wszystkiego wynika, że liczba możliwych do typowanych wykonania ścieżek wyszukiwania wynosi $n \cdot O(T_M)$. Dla danej wykonalnej typowanej ścieżki wyszukiwania (D, typ) liczba sekwencji końcowych, które mogą wystąpić z (D, typ) w rejestrze wyszukiwań, jest ograniczona przez C^n. Zatem liczba zredukowanych rejestrów jest ograniczona przez

$$n \cdot O(T_M) \cdot C^n.$$

Następnym krokiem jest wyznaczenie przybliżonej wartości T_m. Każda ścieżka wyszukiwania $((d1, ..., dm), typ)$ jest albo pojedynczym krokiem w górę (tzn. $m = 1$, $d_1 = 1$), albo można go rozłożyć w niepowtarzalny sposób na $|d_m| + 1$ kolejnych ścieżek wyszukiwania o łącznej długości $m - 1$. Jeżeli typ d_m jest określony (tzn. $d_m \leq -2$), możemy go także rozłożyć na typ d_m. Z tego rozkładu wynika następujące równanie funkcyjne dla funkcji generującej $t(z)$:

$$t(z) = z + zt^2(z) + 4z(t^3(z) + t^4(z) + t^5(z) + ...),$$

gdzie z oznacza trywialną ścieżką składającego się z jednego kroku w górę, $zt^2(z)$ oznacza przypadek $d_m = -1$, w którym d_m nie ma typu, a ostatnia część równania oznacza przypadek $d_m \leq -2$. Prawa strona tego równania jest w rzeczywistości równa $z + zt^2(z) + 4z\frac{t^3(z)}{1-t(z)}$. Z tej postaci wyprowadzamy wielomian definiujący dla $t(z)$:

$$P(z, t) = -t + t^2 + z - tz + t^2z + 3t^3z.$$

W standardowy sposób obliczamy wielomian wyróżniający, uzyskując:

$$-1 - 12z + 24z^2 + 80z^3 + 288z^4.$$

Promień zbieżności $t(z)$ jest jednym z pierwiastków powyższego wielomianu. Ten wielomian ma tylko jeden pierwiastek rzeczywisty dodatni, który wynosi $0,2537...$ Pierwiastek $0,2537...$ jest większy od $1/4$, zatem $T_M = o(4^M)$. Zgodnie z **Założeniem 1**, liczba realizacji jest dokładnie równa liczbie rejestrów wyszukiwania. Zatem, otrzymujemy

$$(C - 2)^M \leq n \cdot O(T_M) \cdot C^n = o(4^M).$$

W ten sposób dla $C \geq 6$ i dostatecznie dużego M, otrzymujemy *sprzeczność*. Oznacza to, że udowodniliśmy, że istnieje taka strategia dla gracza chcącego uniknąć repetycji, przy alfabecie składającym się z 6 liter, wedle której można stworzyć ciąg dowolnej długości bez repetycji dłuższych niż 1.

3 Implementowana strategia dla komputera

Zdecydowaliśmy się zaimplementować strategię podobną do prostego podejścia w szachach. Będziemy patrzeć na wszystkie możliwe ciągi liter o długości x (jeden z parametrów) które mogą zostać dopisane a następnie przypisywać im punkty. Jeżeli w danym ciągu powstanie repetycja, ciąg będzie bardziej promowany od innych. Podobnie, jeżeli repetycja powstanie w mniejszej ilości znaków, będzie to opcja preferowana bardziej niż dłuższa repetycja. W szczególności, jeżeli repetycja powstanie po jednym znaku, przyćmi to wszystkie inne możliwości. Jeżeli repetycja nie powstanie, przypiszemy takiemu ciągowi w aktualnie bliżej nieokreślony sposób wartość. Następnie dla każdej litery zsumujemy punkty ze wszystkich ciągów, które zaczynają się tą literą. To ona zostanie wybrana jako następny ruch komputera.

4 Implementacja

Gra zostanie zaimplementowana w języku C# wraz z GUI stworzonym za pomocą WPFa.

Gracz po uruchomieniu aplikacji zobaczy miejsce na słowo razem z ekranową klawiaturą i docelową długością. Obie te rzeczy (klawiaturę zawierającą alfabet oraz docelową długość słowa) będzie można dostosować w okienku ustawień. Grę w każdym momencie będzie można zrestartować.

Gracz będzie musiał kliknąć odpowiedni znak alfabetu w aplikacji, co doda go na końcu słowa. Następnie komputer wykona swój ruch, dopisując automatycznie swój znak. W przypadku długiego oczekiwania na ruch komputera, gracz będzie w stanie wizualnie rozpoznać fakt, że komputer "myśli".

Jeżeli gra zakończy się zwycięstwem gracza, to znaczy słowo osiągnie wymaganą długość, gracz zobaczy stosowną informację. W przypadku zwycięstwa komputera, poza informacją podkreślona zostanie repetycja.

5 Przykładowa rozgrywka

Grę w niepowtarzanie zawsze rozpoczyna człowiek poprzez wpisanie pierwszej litery.

5.1 Przykład 1

Rozważmy następujące dane wejściowe:

- A = {a, b, c, d, e, f}
- n = 5

Rozgrywka pomiędzy pierwszym i drugim graczem przebiega następująco.

Gracz	Wybrana litera	Uzyskane słowo
1	'a'	'a'
2	'b'	'ab'
1	'c'	'abc'
2	'b'	'abcb'
1	'a'	'abcba'

Słowo ma długość równą 5 czyli n, zatem wygrywa pierwszy gracz.

5.2 Przykład 2

Rozważmy następujące dane wejściowe:

- A = {a, b, c, d, e, f}
- n = 5

Rozgrywka pomiędzy pierwszym i drugim graczem przebiega następująco.

Gracz	Wybrana litera	Uzyskane słowo
1	'd'	'd'
2	'f'	'df'
1	'd'	'dfd'
2	'f'	'dfdf'

Słowo ma długość równą 4 i zawiera repetycję podsłowa 'df' - zatem wygrywa drugi z graczy.

6 Podsumowanie

W dokumentacji teoretycznej udowodniliśmy, **Twierdzenie** 2.7. Podczas testów zamierzamy eksperymentalnie ten dowodu używając naszej aplikacji.

Literatura

[1] Micek Piotr Grytczuk Jarosław, Kozik Jakub. Nonrepetitive games. Accessed: 2022-03-12.

[2] A. THUE. Uber unendliche zeichenreihen. *Norske Vid Selsk. Skr. I Mat-Nat Kl. (Christiana)*, 7:1–22, 1906.