Algorytmiczne Aspekty Teorii Gier

1 Ćwiczenia 1

16 lutego 2009

1. Hex. Plansza gry jest rombem o polach będących sześciokątami, bok rombu wynosi w tradycyjnej wersji $n = 11$, ale można rozważać grę dla dowolnego n. Gra dwóch graczy. Stawiają swoje piony na polach naprzemiennie, jeden powiedzmy białe, drugi czarne. Cel gry każdego z nich polega na połączeniu pary naprzeciwległych boków ścieżką swoich pionów (jeden gracz ma parę boków poziomych, drugi ukośnych).
Strona o grze: `http://pl.wikipedia.org/wiki/Hex_(gra)`
Który gracz ma strategię wygrywającą i czy któryś ma?

Uwagi ogólne

- Ogólna uwaga o tym, kiedy gracz posiada strategię wygrywającą, że pozycja jest wygrywająca, gdy z niej mogę pójść do przegrywającej, a pozycja jest przegrywająca, gdy z niej gdziekolwiek nie pójdę, to są wygrywające (oczywiście to działa tylko dla naprzemiennych ruchów).
- Komentarz do gry, wymyślona w 1942 przez jakiegoś gościa, 5 lat później J. Nash, ten on Pięknego Umysłu ponownie ja wymyślił i spopularyzował.
- Warto powiedzieć, że da się w tę grę grać na kratkowanej kartce, robimy wtedy kwadrat $n \times n$ i mówimy, że z konkretnym polem sąsiadują pola: na lewo, na prawo, do góry, do dołu, na skos lewo-góra oraz na skos prawo-dół. Gracz musi połączyć naprzeciwległe boki kwadratu.

Wskazówki

- Pokazać, że w tej grze nie ma remisów. Robi się to następująco. Bierzemy spójną składową dolnego boku (w wersji z sześciokątami). Jeśli dochodzi ona do górnego boku, to gracz łączący te boki wygrał. W przeciwnym przypadku, idąc granicą tej spójnej składowej, otrzymujemy ścieżkę łączącą drugą parę naprzeciwległych boków.
- Zauważmy, że skoro gra jest skończona (każda rozgrywka skończona i o długości ograniczonej przez pewne M), to na pewno któryś gracz ma strategię wygrywającą, można to łatwo pokazać idąc od dołu po drzewie gry (przy okazji pokażemy drzewo gry).
- Spróbować niekonstruktywnie.
- Nie wprost. Robi się tak. Przypuśćmy, że jestem drugi i mam strategię wygrywającą. Wtedy postawię gdzieś i będę grał tak, jak bym był pierwszy. Jeśli muszę, zgodnie ze strategią pierwszego tam, gdzie stoi mój zapomniany pion, to przypominam sobie o nim, stawiam gdzieś indziej pion i zapominam o nim. W ten sposób mam zawsze jeden zapomniany pion, ale ponieważ żaden mój pion mi nie przeszkadza, więc to przechodzi. Ogólnie taka metoda jest często stosowana i nazywa się *strategy-stealing*.
- Największe n, dla którego znana jest strategia, to $n = 9$.

2. NIM Mamy kilka rzędów, w każdym rzędzie po kilka pionów. Jest dwóch graczy, ruszają się na przemian. W jednym ruchu można wybrać konkretny rząd i zdjąć z niego kilka pionów. Zdejmując ostatni pion wygrywam. Kto ma strategię wygrywającą i jaka ona jest?

Uwagi ogólne

- Gramy sobie w to na tablicy.

Wskazówki

- Spróbować rozwiązać małe szczególne przypadki.
- Jak jest dla 2 rzędów. Wychodzi łatwo, że gdy są równe, to jest to pozycja przegrywająca, w przeciwnym przypadku wygrywająca.
- Jak jest dla 3 rzędów?
- Pokazać, że dla $a, b \in \mathbb{N}$ istnieje takie $f(a, b)$, że przy trzech rzędach, długości a, b oraz $f(a, b)$ przegrywamy, a dla dowolnej innej wielkości tego trzeciego słupka wygrywamy. Znaleźć konkretnie to f.
- Sprawdzić dla małych liczb i próbować zgadnąć co to jest. Sprawdzić dla 0, 1, 2, 3, zrobić sobie tabelkę.
- Można zauważyć, że to jest xor bitowy, tzn. $f(a, b) = a \oplus b$, gdzie xor rozumiemy po współrzędnych. (Zauważmy, że wtedy $f(a, b) \oplus a \oplus b = \mathbf{0}$.) Teraz już chcemy dowodzić, że to działa ogólnie.
- Trzeba wykazać, że z xora wszystkich n rzędów równego 0, czyli pozycji przegrywającej zawsze dochodzimy do pozycji wygrywającej, czyli xora różnego od 0 oraz, że z xora różnego od 0, czyli pozycji wygrywającej zawsze możemy dojść do pozycji przegrywającej, czyli xora równego 0.
- To, że z 0 nie dojdziemy do 0 jest jasne, bo zmieniamy liczbę w pewnej kolumnie i xor nie będzie równy ponownie 0.
- Gorzej, że z nie 0 możemy dojść do 0. Robi to się następująco. Niech $x_1, \ldots, x_n$ to liczby pionów w poszczególnych rzędach. Niech $x = x_1 \oplus x_2 \oplus \ldots \oplus x_n$. Spójrzmy na indeks wiodącej jedynki w x, niech będzie to k-ta pozycja. Z własności xora wynika, że dla pewnego i liczba x_i ma na k-tej pozycji również 1-kę. Wówczas zmniejszamy tę liczbę x_i zmieniając tę jedynkę na zero, a mniejsze liczby odpowiednio, tak, żeby xor wszystkiego wyszedł same zero (możemy te mniejsze cyfry dowolnie dopasować, bo już wiodącą 1-kę zmieniliśmy na 0). Czyli innymi słowy x_i zmieniamy na $x_i \oplus x$ i korzystamy z tego, że $x_i > x_i \oplus x$.

3. NIM misère. NIM, tylko z drobną modyfikacją, ten, kto zbierze ostatni pion przegrywa.

Wskazówki

- Podobnie jak poprzednio.

- Identycznie jak poprzednio, tylko dla małych przypadków inaczej.
- Tylko dla rzędów długości jeden inaczej.
- Gramy tak, jak dawniej, tylko w momencie, gdy mamy zostawić same rzędy długości jeden zostawiamy nie tak, jak w poprzednim przypadku parzystą ilość rzędów, ale nieparzystą. Prosto pokazać, że to właśnie gracz utrzymujący xor równy 0 po swoim ruchu może dokonać tej decyzji i że faktycznie zawsze może zmienić tę parzystość.

4. Grundy numbers (nimbers). Najpierw wstęp. Czasem mamy do czynienia z (można tak powiedzieć) produktem gier skończonych. Co więcej, gry te muszą mieć taką własność, że dla każdej pozycji każdy z graczy może zrobić te same ruchy (przykładowo NIM jest taką grą, ale szachy już nie, tam jest gracz biały, który rusza białymi i czarny, który rusza czarnymi figurami). Konkretnie, gramy w na przykład dwóch grach i nasz ruch polega na wykonaniu jednego ruchu w dokładnie jednej z tych dwóch gier. Przegrywamy, gdy nie możemy zrobić ruchu w żadnej z gier. Tak na przykład NIM z trzema rzędami długości 3, 4 oraz 6 to produkt trzech gier: $||| \otimes |||| \otimes ||||||$, jeśli przez grę z n znakami $|$ oznaczmy grę w jednym rzędzie o n zapałkach, czy też pionach (skądinąd niespecjalnie ciekawą).

Chcemy umieć rozwiązywać takie gry, to znaczy mówić kto ma strategię i jaka ona jest. Produkt gier A i B oznaczmy przez $A \otimes B$. Chcemy znaleźć taką informację inf o grze, żeby z tej informacji dało się odczytać, czy dana pozycja jest wygrywająca, czy przegrywająca, żeby $inf(A \otimes B)$ dało się nietrudno obliczyć mając $inf(A)$ oraz $inf(B)$. A przy tym chcemy dość prosto umieć obliczyć tę informacje dla gier składowych.

Uwagi ogólne

- To się przydaje w topcoderze, często zadanie za 1000 punktów w DIV 1 jest właśnie na jakieś gry, nie trzeba dużo kodować, tylko takie rzeczy umieć.
- Grundy to nie nazwa, tylko nazwisko (chyba).

Wskazówki

- Opierajmy się na NIM-ie, zobaczmy jak tam to działa.
- Wystarczy jedna liczba.
- To jest xor i podobnie jak w NIM-ie jeśli xor wynosi 0, to jest to pozycja przegrywająca, a jeśli jest różny od 0, to jest to pozycja wygrywająca.
- Pokazujemy jak w dowolnej grze skończonej obliczyć inf dla niej. Rysujemy drzewo gry. W liściach piszemy 0, bo są to pozycje przegrywające. W dowolnym węźle wewnętrznym piszemy najmniejszą liczbę nie występującą wśród synów.
- To jest dobra definicja, udowodnimy, że granie tak, żeby xor na wszystkich grach był 0 po ruchu jest ok. Gramy jak wcześniej. Jeśli ktoś zwiększył liczbę idąc do syna, to my możemy ją zmniejszyć w tej samej grze. W przeciwnym wypadku gramy jak w NIM-ie. Tak że jest ok.

Obserwacje

- Zauważmy, że produkt dwóch identycznych gier jest jak gra pusta $H \otimes H = \emptyset$
- Produkt przegrywającej gry H i dowolnej gry G jest równoważny grze G, $H \otimes G = G$

5. Gra czekolada. Mamy prostokątną czekoladę o rozmiarach $n \times m$. W lewym dolnym rogu na kostce jest trucizna. Jest dwóch graczy, ruszają się na zmianę. Ruch polega na wybraniu punktu kratowego i zjedzeniu wszystkiego na prawo i w górę od tego punktu (zjedzenie musi być niepuste). Przegrywa ten, kto zje truciznę. Pytanie brzmi - który gracza ma strategię wygrywającą.

Za to zadanie jest konkurs, kto pierwszy po zajęciach prześle rozwiązanie wygrywa konkurs i na następnych zajęciach otrzymuje czekoladę. Rozwiązanie też na następnych zajęciach. Umawiamy się, że nie przeszukujemy sieci, jest to do znalezienia na sieci.

2 Ćwiczenia 2

23 lutego 2009

1. Gra czekolada, inaczej Chomp. Definicja problemu jak na poprzednich ćwiczeniach. Mamy prostokątną czekoladę o rozmiarach $n \times m$. W lewym dolnym rogu na kostce jest trucizna. Jest dwóch graczy, ruszają się na zmianę. Ruch polega na wybraniu punktu kratowego i zjedzeniu wszystkiego na prawo w górę od tego punktu (zjedzenie musi być niepuste). Przegrywa ten, kto zje truciznę. Pytanie brzmi - który gracz ma strategię wygrywającą.

Rozwiązanie
Rozwiązanie pokazuje laureat konkursu. Metoda jest niekonstruktywna, trochę jak strategy-stealing. Gra jest skończona, więc dokładnie jeden gracz ma strategię wygrywającą z początkowej pozycji. Przypuśćmy, że strategię wygrywającą ma gracz drugi. Wówczas wszystkie pozycje osiągalne po pierwszym ruchu są wygrywające. Przypuśćmy, że pierwszy gracz zje jedną kostkę, tę narożną. Z założenia sytuacja ta jest wygrywająca. Jednak po wykonaniu dowolnego ruchu z tej pozycji otrzymujemy pozycję, która była osiągalna na samym początku, czyli pozycję wygrywającą. Sprzeczność, bo żeby pozycja była wygrywająca, to musimy z niej móc osiągnąć pozycję przegrywającą.

2. NIM na liczbach porządkowych. Zasady jak z NIM, tylko wysokość słupka to pewna liczba porządkowa mniejsza niż ω^ω. Ruch gracza polega na zmniejszeniu liczby porządkowej na słupku. Rozstrzygnąć który gracz ma strategię wygrywającą i jak powinien grać.

Uwagi ogólne

- Przypomnieć co to jest liczba porządkowa. Dobry porządek to porządek liniowy taki, że każdy podzbiór niepusty ma element najmniejszy. Liczba porządkowa to reprezentant klasy izomorficzności dobrych porządków.
- W dobrym porządku nie występują nieskończone ciągi zstępujące, gdyby istniały, to zbiór złożony z takiego ciągu nie miałby elementu najmniejszego. Zatem w liczbach porządkowych również nie ma nieskończonego zstępującego ciągu.

Wskazówki

- Trzeba zobaczyć jak wyglądają liczby porządkowe mniejsze od ω^ω. W ogóle najpierw zobaczmy, że po kolei idą 1, 2, 3, ..., ω, $\omega+1$, ..., 2ω, ... Te liczby porządkowe to jakby wielomiany od ω, czyli są postaci $a_n\omega^n + a_{n-1}\omega^{n-1} + \ldots + a_1\omega + a_0$. Warto zobaczyć kilka przykładów konkretnych liczb.
- Wzorować się na NIM-ie.
- Xor po pozycjach ma być 0, wtedy pozycja jest przegrywająca.
- Dowodzimy to analogicznie jak dla zwykłego NIM-a, jedyny problem jest, czy z xora nie 0 możemy dojść do 0, ale to robimy tak, że znajdujemy maksymalną pozycję, na której xor współczynników jest różny od 0, analogicznie jak w NIM-ie zmniejszamy go do 0, a mniejsze współczynniki możemy ustawić dowolnie (bo liczbę już zmniejszyliśmy).
- Warto też zauważyć, że korzystamy implicite z tego, że gra jest skończona, czyli, że nie ma łańcuchów nieskończonych w liczbach porządkowych.

3. Silver dollar game, gra w monety. Plansza gry to pasek wysokości jeden, nieskończony w prawo (możemy myśleć o tym jako o liczbach naturalnych na przykład). Jest na nim ustawiona skończona ilość pionów. Gra dwóch graczy, ruszają się na zmianę. Ruch polega na przesunięciu piona w lewo, ale tak, żeby nie przeskakiwać ani nie najeżdżać na żaden inny pion. Przegrywa gracz nie mający ruchu, czyli zastający na planszy sytuację, w której wszystkie piony są zepchnięte maksymalnie w lewo. Rozstrzygnąć kto posiada strategię wygrywającą i jaka ona jest.

Wskazówki

- Czy ta gra nam coś przypomina?
- NIM!
- Starać się sprowadzić jakoś do NIMa.
- Jak byśmy chcieli każdą przerwę pomiędzy pionami traktować jak słupek z NIMie, to powstaje problem, że pojedynczy ruch może zmodyfikować dwa słupki. Spróbujmy zaradzić jakoś temu, żeby ruch modyfikował tylko jeden słupek. Jak zdefiniować zatem słupki?
- Rozwiązaniem jest powiedzenie, że odpowiednikami słupków w NIMie będzie co druga przerwa między pionami. Wówczas jeden ruch modyfikuje dokładnie jeden słupek. Występuje problem, że słupki można nie tylko zmniejszać, ale też zwiększać. Jest to jednak pozorny problem, gdy ktoś zwiększy słupek, to zmniejszamy go do poprzedniej wartości. Ponieważ gra jest skończona, to nie może to trwać zbyt długo. Chcąc wygrać utrzymujemy jak w NIMie niezmiennik, że po naszym ruchu xor wszystkich słupków (czyli u nas długości co drugiej przerwy) ma być równy 0.

4. Wartość drzewa gry. Najpierw zdefiniujmy wartość drzewa gry. Rysujemy drzewo gry, w każdym liściu jeśli wygrywa w nim Ewa, to piszemy 1, a jeśli Adam, to 0. W każdym wierzchołku, jeśli tam rusza się Ewa, to oczywiście chce wybrać większą z wartości poddrzew, jeśli Adam, to mniejszą, czyli w wierzchołkach zależnie od właściciela piszemy odpowiednio max lub min z wartości poddrzew. W ten sposób w każdym wierzchołku napiszemy liczbę, która będzie równa 1 wtedy i tylko wtedy, gdy Ewa posiada z niego strategię wygrywającą. Wartość drzewa gry to liczba w korzeniu.
Warto przy okazji przypomnieć algorytm minimax.

Uwaga: Podobnie możemy zdefiniować wartość drzewa gry, gdyby gra byłaby bardziej skomplikowana, tzn. nie kończyła się po prostu wygraną któregoś z graczy, ale powiedzmy Adam płaciłby Ewie x złotych. Wówczas również Ewa chciałaby zmaksymalizować, a Adam zminimalizować wypłatę, a wartość obliczona jak wcześniej, czyli wartość drzewa gry oznaczałaby wartość, co do której Adam może sobie zapewnić, że co najwyżej tyle zapłaci, a Ewa, że co najmniej tyle dostanie.

Zadanie Pokazać, że dla każdego deterministycznego algorytmu obliczającego wartość drzewa gry dla pełnego drzewa binarnego istnieje takie etykietowanie liści liczbami 0 i 1, dla którego algorytm będzie musiał zajrzeć do wszystkich liści.

Wskazówki

- Można na tym zadaniu pokazać fajny trik, zastosowanie gier do udowadniania czegoś (tu o drzewie gry akurat). To jest dość popularny schemat. Ogólnie, jeśli chcemy pokazać, że czegoś nie da się zrobić (tu - obliczyć wartość drzewa przed obejrzeniem wszystkich liści), to definiujemy grę pomiędzy dwoma graczami, jednym, który stara się to jednak zrobić (u nas nazwany Algorytmem), drugim, który mu przeszkadza, nazywany standardowo Spoilerem.

- Zrobimy następująco. Ruch Algorytmu to wybranie konkretnego liścia i spytanie o niego. Ruch Spoilera to wybranie wartości 0 lub 1 i powiedzenie, że ten liść ma tę właśnie wartość. Spoiler stara się utrzymać sytuację, w której przy znanych już odpowiedziach wartość drzewa gry nadal jeszcze nie jest ustalona.

- Dość standardowe jest przy pisaniu programów na drzewa, że używamy rekurencji. Użyjmy jej dowodowego odpowiednika - indukcji.

- Robimy indukcyjnie po wysokości poddrzewa. Zakładamy, że dla wysokości h Spoiler może grać tak, że wartość ustala się dopiero na koniec. Dla drzewa pełnego wysokości $h+1$ gramy Spoilerem w każdym drzewie odpowiednio, a gdy w którymś zostaje zadane pytanie o ostatni liść w tym drzewie, to odpowiadamy tak, by wartość tego poddrzewa wyniosła 0, jeśli korzeń to wierzchołek max, a 1, jeśli korzeń to min. W drugim poddrzewie grami do końca, więc faktycznie trzeba sprawdzić wszystkie liście.

- Istnieją inne rozwiązania, analizujące co tak naprawdę dzieje się w tym drzewie i wskazujące explicite strategię Spoilera. Generalnie Spoiler stara się w każdym wierzchołku pod którym jest niespytany liść zachowywać niepewność, a jeśli już pytamy o ostatni liść pod danym wierzchołkiem, to odpowiadamy tak, by otrzymana wartość nic nie wniosła (czyli wynosiła 0 w synach wierzchołków max i 1 w synach wierzchołków min).

3 Ćwiczenia 3

2 marca 2009

1. Wartość drzewa gry, algorytm zrandomizowany. Znaleźć algorytm zrandomizowany, który dla pełnego drzewa binarnego o wysokości $h = 2k$ (wysokość definiujemy jako ilość krawędzi na najdłuższej ścieżce od korzenia) z wartościami zero i jeden w liściach oblicza wartość drzewa gry (definicja wartości drzewa gry pod koniec poprzednich ćwiczeń) odwiedzając średnio nie więcej niż 3^k liści.

Wskazówki

- Najpierw zaproponujmy jakiś sensowny algorytm, a później będziemy się martwić obliczeniem czasu, w którym działa. Najprawdopodobniej to będzie ten algorytm o który chodzi.
- Zastanówmy się, czy jeśli wiemy coś o pewnych poddrzewach, to być może nie musimy obliczać wartości innych.
- Tak. Przykładowo jeśli w węźle max jedno z poddrzew wiemy, że ma wartość 1, to nie musimy obliczać wartości drugiego, bo wiadomo, że w ojcu też będzie wartość 1. Analogicznie dla min i wartości 0.
- Warto przypomnieć, omówić główną ideę alfa-beta obcięć w algorytmie minimax. To ma podobną ideę jak u nas, tylko, że w naszym przypadku, gdy są wartości tylko 0 i 1, to możemy wykonywać nawet prostsze obcięcia.
- Algorytm jest zatem następujący:
  Będąc w danym węźle losujemy które z jego poddrzew obliczamy najpierw. Jak obliczymy wartość tego poddrzewa i wówczas wartość ojca będzie już zdeterminowana, to nie obliczamy wartości drugiego poddrzewa. W ten sposób postępujemy w każdym węźle.
- Zastanówmy się teraz średnią ilością liści, które musimy odwiedzić. Jak to obliczyć?
- Niech *drzewo dobre* to takie drzewo, które jeśli korzeń to max, to ma wartość 1, a jeśli korzeń do min, to ma wartość 0. Niech *drzewo złe* to takie drzewo, które nie jest dobre. Niech D_k i Z_k to średnia ilość odwiedzeń liści odpowiednio w drzewie (pełnym, binarnym) dobrym i złym o głębokości k. Chcemy zatem pokazać, że $D_{2k} \leqslant 3^k$ i $Z_{2k} \leqslant 3^k$.
- Zastosujmy rekurencję. Mamy oczywiście $D_0 = 1$, $Z_0 = 1$. Poza tym $Z_{k+1} = D_k + D_k = 2D_k$, gdyż jeśli drzewo głębokości $k+1$ jest złe, to jego poddrzewa muszą być dobre, czyli nie ominiemy żadnego obliczenia. Mamy też $D_{k+1} \leqslant \frac{1}{2}(Z_k + (D_k + Z_k)) = Z_k + \frac{1}{2}D_k$, gdyż jeśli drzewo głębokości $k+1$ jest dobre, to któreś jego poddrzewo jest złe i mamy co najmniej $\frac{1}{2}$ prawdopodobieństwa, że najpierw policzymy jego wartość i nie będzie trzeba liczyć wartości tego drugiego.
- Łatwo teraz indukcyjnie wykazać, że $D_{2k} \leqslant 3^k$ i $Z_{2k} \leqslant 3^k$ (przez indukcję dowodzimy oba fakty razem).
- Gdy przyjrzymy się dokładniej sytuacji, to widać, że jest tu jeszcze trochę luzu i asymptotycznie ilość odwiedzonych liści powinna być mniejsza niż 3^k. Można to rozwiązać rekurencyjnie. Korzystając z napisanych wyżej rekurencji można otrzymać, że $D_{k+2} \leqslant$

$Z_{k+1} + \frac{1}{2}D_{k+1} = \frac{1}{2}D_{k+1} + 2D_k$. Rozwiązując rekurencję otrzymujemy asymptotyczne oszacowanie górne na D_{2k} wynoszące $\left(\frac{\sqrt{33}+1}{4}\right)^2 k = \left(\frac{34+2\sqrt{33}}{16}\right)^k \approx 2,84^k$.

2. Pebbling game. Gra jest jednoosobowa. Gramy na DAG-u (Directed Acyclic Graph), czyli grafie skierowanym acyklicznym o stopniu wejściowym 2 (czyli do każdego wierzchołka wchodzą co najwyżej 2 strzałki). Celem gry jest położenie kamyka na pewnym wyróżnionym wierzchołku v. Dostępne ruchy to:

- zdjęcie kamyka, który leży na pewnym wierzchołku
- położenie kamyka na wierzchołku w, jest to jednak dozwolone o ile na wszystkich wierzchołkach, z których wchodzą strzałki do w leżą już kamyki (czyli w szczególności, jeśli do w nie wchodzi żadna strzałka, to od razu można kłaść na w kamyk).

Naszym celem jest zminimalizować ilość użytych w grze kamyków. Ile kamyków potrzeba do położenia kamyka w korzeniu drzewa binarnego o wysokości h (niekoniecznie pełnego)? A ile potrzeba do położenia kamyka w korzeniu drzewa binarnego o n wierzchołkach?

Uwaga: Dalszą częścią tego zadania jest zadanie domowe nr 1.

Wskazówki

- Sprawdzić ile wychodzi dla małych przypadków, na przykład małego pełnego drzewa binarnego.
- Udowodnić, że $h + 2$ działa.
- Indukcyjnie po wysokości (łatwo, dla jednego syna mam co najwyżej $h + 1$, zostawiam kamyk w synie i mam jeszcze $h + 1$ na drugiego syna (o wysokości być może mniejszej nawet niż $h - 1$). Zatem faktycznie $h + 2$ wystarcza.
- Dla drzewa o n wierzchołkach podobnie.
- Można pokazać indukcyjnie, że $\lceil \lg n \rceil$ wystarczy, najpierw do drzewa o syna o większej (ściśle rzecz biorąc nie mniejszej) ilości wierzchołków dajemy kamyk, a potem mając ten jeden kamyk w większym synu robimy mniejszego syna (który ma przynajmniej 2 razy mniej wierzchołków niż korzeń). W ten sposób łatwo wychodzi.

3. Piony w rzędzie. Gra dwóch graczy, ruszają się naprzemiennie. Na początku w rzędzie jest ustawiona pewna liczba pionów, pomiędzy niektórymi są przerwy. W pojedynczym ruchu można zdjąć z planszy bądź 1 pion, bądź 2 sąsiadujące piony. Przegrywa ten, kto nie może zrobić ruchu. Napisać program, który odpowie na pytanie kto z danej pozycji początkowej (czyli na przykład ciągu spójnych grup pionów wielkości odpowiednio $2, 5, 3, 1, 3$) posiada strategię wygrywającą (gra ta jest w oczywisty sposób zdeterminowana, gdyż jest skończona, czyli dla każdej pozycji któryś z graczy posiada strategię wygrywającą z tej pozycji).

Wskazówki

- Skorzystać z czegoś, co już było.

- Spojrzeć jak wyglądają możliwe ruchy. Tak naprawdę albo pojedynczy słupek zmniejszamy, albo rozdzielamy na dwa mniejsze słupki, na których gry już toczą się rozdzielnie.
- Zauważmy, że możemy tu zastosować Grundy numbers, o których była mowa na pierwszych ćwiczeniach. Zauważmy, że oprócz obserwacji powyżej - nasza gra ma taką własność, że gracze są tu symetryczni, czyli każdy z nich ruszając się z konkretnej planszy może zrobić te same ruchy. Ta własność również jest konieczna do skorzystania z Grundy numbers.
- Dlaczego tak naprawdę Grundy numbers. Zobaczmy, że jeśli ze słupkiem długości n związżemy liczbę $G(n)$, to jest ona równa najmniejszej liczbie całkowitej większej lub równej zero, która nie występuje wśród $G(i) \otimes G(n-1-i)$ oraz wśród $G(i) \otimes G(n-2-i)$ dla i od 0 do $n-1$ lub $n-2$ odpowiednio.
- Schemat działania jest następujący. Zwracamy xor wszystkich Grundy numberów dla odpowiednich spójnych grup pionów. Wcześniej liczymy Grundy numbery dla wszystkich liczb od 0 do $M-1$ (załóżmy że jest to maksymalna wielkość słupka), metodą liczenia najmniejszej liczby nie należącej do zbioru określonego wyżej.
- Przejdźmy do napisanie kodu. Przykładowy może wyglądać następująco:

```
// Uwaga, nie kompilowałem tego programu, więc może zawierać błędy
// Tu oczywiście jakieś includy
#define M 1000
#define pb push_back
#define REP(i,n) for(i=0;i<n;i++) // moje ulubione makro, bardzo skraca napisy

int gnumbers[M];

int mex(vector<int> v) {
// minimum excludant
// napisałem go chamsko kwadratowo, choć pewnie da się lepiej
  int i, j;
  REP(i,v.size()+1) {
    REP(j,v.size()) {
      if (v[j]==i) break;
      return i;
    }
  }
}

int gnumber(int n) {
  int i;
  vector<int> v;
  REP(i,n/2) {
  // To n/2 jest tu z górką i tak zwykle jedna, czy dwie pary
  // pośrodku zostaną wrzucone dwa razy
    v.pb(gnumbers[i]^gnumbers[n-1-i]);
    v.pb(gnumbers[i]^gnumbers[n-2-i]);
```

```
  }
  return mex(v);
}

int mainResult(vector<int> v) {
  int i, ret;
  REP(i,M) gnumbers[i] = 0;
  REP(i,M) gnumbers[i] = gnumber(i);
  ret = 0;
  REP(i,v.size()) ret ^= gnumbers[v[i]];
  return ret;
}
```

4 Ćwiczenia 4

9 marca 2009

1. Gra Ehrenfeuchta-Fraïssè. Uwaga: Nie jest pewne, czy warto to robić na ćwiczeniach, wychodzi z tego bardziej wykład niż ćwiczenie, a w dodatku niektórzy ludzie już to znają, jako, że pojawia się czasem na logice na I roku.

W tym ćwiczeniu nie ma jednego konkretnego zadania, będziemy szli po kolei przez zagadnienia dotyczące gier Ehrenfeuchta-Fraïssè. Zobaczymy na tym przykładzie, że gry mogą mieć też istotne zastosowanie w logice.

Ogólny problem jaki rozważamy jest następujący: Czy pewne dwie struktury $\mathcal{A}$ i $\mathcal{B}$ są rozróżnialne przy pomocy pewnej logiki, czyli, czy istnieje formuła tej logiki, która w jednej z tych struktur jest prawdziwa, a w drugiej fałszywa.
Ponieważ tu chcemy pokazać tylko ideę, to będziemy rozważać logikę pierwszego rzędu z relacją $\leqslant$, czyli $FO(\leqslant)$, a rozważane struktury będą miały postać jedynie $(\langle 1, 2, \ldots, n\rangle, \leqslant)$ dla różnych $n \in \mathbb{N}$, czyli innymi słowy liczby od 1 do n z naturalnym porządkiem liniowym.

Rozważmy przykład, weźmy dwie struktury, $\mathcal{A} = \langle 1, 2, 3\rangle$ oraz $\mathcal{B} = \langle 1, 2, 3, 4\rangle$, implicite zakładamy, że są one z liniowym porządkiem $\leqslant$. Zadajmy pytanie, czy istnieje pewna formuła $FO(\leqslant)$ o randze kwantyfikatorowej (czyli intuicyjnie głębokości najgłębszego kwantyfikatora) równej 3, która rozróżnia podane struktury. Po chwili namysłu dochodzimy do wniosku, że faktycznie, istnieje

$$\exists x \Big(\big(\exists y \exists z \ (x \leqslant y) \wedge (x \leqslant z) \wedge (y \leqslant z) \big) \wedge \big(\exists y \ (y \leqslant x) \big) \Big)$$

A czy istnieje formuła $FO(\leqslant)$ o randze kwantyfikatorowej równej 2 rozróżniająca te struktury? Wydaje się, że nie. Ale jak to pokazać?

Zdefiniujmy grę Ehrenfeuchta-Fraïssè. Zdefiniujemy ją tylko na przykładzie logiki pierwszego rzędu z liniowym porządkiem, ale w bardzo podobny sposób są definiowane analogiczne gry dla innych logik na innych strukturach. Gra dwóch graczy, Duplikator i Spoiler. Jak zwykle, Duplikator ma na celu pokazanie, że coś jest podobne, czyli w naszym przypadku będzie chciał pokazać, że struktury $\mathcal{A}$ i $\mathcal{B}$ są nierozróżnialne, Spoiler będzie mu przeszkadzał i pokazywał,

że jednak da się rozróżnić. Najpierw rusza się Spoiler i kładzie kamień na pewnym elemencie którejś ze struktur (intuicyjnie - mówiąc - a takiego elementu to w tej drugiej strukturze nie znajdziesz Duplikatorze). Potem rusza się Duplikator. Musi położyć kamień w tej strukturze, w której Spoiler nie położył swojego, na pewnym elemencie (intuicyjnie - mówiąc - a właśnie, że nie Spoilerze, ten element jest bardzo podobny do Twojego w tamtej strukturze). Tak Spoiler i Duplikator grają przez n rund. Spoiler co turę może zmieniać strony, nie jest na stałe przywiązany do kładzenie kamieni w konkretnej strukturze. Duplikator kładąc swój kamień musi respektować porządek, czyli jeśli kiedyś Spoiler położył swój kamień na elemencie S_i i Duplikator odpowiedział elementem D_i, a teraz Spoiler położył kamień na S_j, to odpowiedź Duplikatora D_j musi spełniać warunek

$$(S_i \leqslant S_j) \iff (D_i \leqslant D_j),$$

jeśli S_i i S_j są w tej samej strukturze albo

$$(S_i \leqslant D_j) \iff (D_i \leqslant S_j),$$

jeśli S_i i D_j są w tej samej strukturze. Duplikator wygrywa grę, jeśli przetrwa n rund i ciągle będzie miał gdzie położyć kamień w odpowiedzi na ruch Spoilera. Spoiler wygrywa grę, gdy w pewnym momencie Duplikator nie będzie miał odpowiedzi.

Sformułujmy teraz twierdzenie o zdefiniowanej grze.

Twierdzenie Duplikator wygrywa grę Ehrenfeuchta-Fraïssè o n rundach na strukturach $\mathcal{A}$ i $\mathcal{B}$ wtedy i tylko wtedy, gdy struktury te są nierozróżnialne za pomocą formuł głębokości n w rozważanej logice.

Przeprowadzimy teraz szkic dowodu tego twierdzenia, dla przypadku $FO(\leqslant)$ i porządków liniowych. Dlatego robimy to w tak uproszczonym przypadku, żeby zobaczyć ideę, a nie zamotać się w zbytnich szczegółach technicznych.

Pokażemy równoważność:
istnieje formuła rozróżniająca głębokości n $\iff$ Spoiler wygrywa w n rundach.

Najpierw pokażemy implikację $\Rightarrow$. Pokażemy ją przez indukcję fakt, że jeśli mamy grę z być może już jakimiś kamykami położonymi, to powyższy fakt jest prawdziwy.

Dla głębokości równej 1 łatwo, formuła musi być czymś postaci $\exists x$ i ewidentnie jeśli jest prawdziwa w jednej strukturze, a w drugiej nie, to Spoiler wskaże ten element, który istnieje, a w drugiej strukturze nie istnieje żaden.

Przypuśćmy, że dla n mamy dowiedzioną tezę indukcji. Wiemy teraz, że istnieje formuła rozróżniająca struktury głębokości $n+1$. Można ją rozbić na boolowską kombinację formuł głębokości $n+1$, które zaczynają się od kwantyfikatora, jeśli boolowska kombinacja takich formuł rozróżnia struktury $\mathcal{A}$ i $\mathcal{B}$, to któraś z formuł zaczynających się od kwantyfikatora też musi je rozróżniać. Jeśli formuła ta zaczyna się od $\exists x$, to Spoiler stawia kamyk w ten strukturze, w której istnieje taki x na tym właśnie x, Duplikator siłą rzeczy postawi w drugiej strukturze kamyk na elemencie, który nie spełnia tej własności (bo taki tam nie istnieje). Analogicznie, gdy mamy formułę zaczynającą się od $\forall x$, to Spoiler stawia kamyk tam, gdzie nie każdy x to spełnia, na takim, który nie spełnia. Korzystając z założenia indukcyjnego dostajemy tezę (bo warunek na x był głębokości n).

Teraz pokażmy przeciwną implikację $\Leftarrow$. Baza indukcji analogicznie jest prosta. Przypuśćmy, że z n rundowej strategii Spoilera umiemy stworzyć formułę głębokości n. Jeśli Spoiler stawia kamyk na jakimś elemencie, to zaczynamy formułę - $\exists x$. Zauważmy, że teraz obie struktury podzielone są na 2 kawałki, po lewej stronie od położonego kamyka i po prawej stronie.

Spoilerowi nie opłaca się grać w obu, gdyż są one niezależne, możemy więc założyć, że dalsza gra będzie się odbywać tylko w jednej z nich. Ponieważ Spoiler jest w stanie wygrać w n rundach w którejś z części, to istnieje formuła φ głębokości n rozróżniająca te części. Zatem nasza ogólna formuła ma (w zależności od tego gdzie φ jest prawdziwa, a gdzie fałszywa) postać

$$\exists x\ \varphi \wedge (x \leqslant y_1) \wedge \ldots \wedge (x \leqslant y_n),$$

lub

$$\exists x\ \neg(\varphi \wedge (x \leqslant y_1) \wedge \ldots \wedge (x \leqslant y_n)),$$

gdzie $y_1, \ldots, y_n$ to zmienne występujące w φ (zakładamy dla ułatwienia, że gra toczyła się w lewej połówce). W ten sposób dowiedliśmy twierdzenia (choć nie do końca formalnie może).
Uwaga: Blisko związane z tym ćwiczeniem jest zadanie domowe nr 2.

2. Gra Banacha-Mazura. Najpierw historia. Banach i Mazur należeli do bardzo aktywnego środowiska przedwojennych matematyków działających we Lwowie. Mieli oni zwyczaj spotykać się wieczorami w kawiarni i w atmosferze imprezy rozwiązywać zadania. Zapisywali je sobie w tak zwanej Księdze Szkockiej, nierzadko z zakładami typu - na tego, kto rozwiąże podane zadanie czeka nagroda w postaci żywej gęsi - bądź inne podobne. Wiele problemów podanych w Księdze Szkockiej i rozwiązanych przez tamtych ludzi okazało się bardzo ważne dla matematyki. W księdze tej między innymi pojawiła się (podobno pierwsza rozważana) gra nieskończona, tzn. gra Banacha-Mazura, sformułował ją Mazur, rozwiązał Banach.

Gra toczy się pomiędzy dwoma graczami, ruszają się oni naprzemiennie, nazwijmy ich, dla ułatwienia, Banach i Mazur (choć w oryginale oczywiście tak nie było). Gra odbywa się na odcinku $[0,1]$. Dany jest pewien wyróżniony zbiór $F \subset [0,1]$. Pierwszy rusza się Mazur i wybiera pewien odcinek otwarty $I_1 \subset [0,1]$. Następnie rusza się Banach i wybiera pewien odcinek otwarty I_2 zawarty w I_1. Dalej gra toczy się analogicznie, gracze wybierają ciąg odcinków $I_1, I_2, \ldots$, taki, że dla każdego k zachodzi $I_{k+1} \subset I_k$ oraz żaden odcinek nie może być pusty. Banach wygrywa, gdy przecięcie wszystkich odcinków i zbioru F jest niepuste, czyli $\left(\bigcap_{n=1}^{\infty} I_n\right) \cap F \neq \emptyset$. Pytanie brzmi, jaki warunek musi spełniać zbiór F, by Banach posiadał strategię wygrywającą?

Wskazówki

- Czy Banachowi opłaca się, by zbiór F był duży, czy mały? Oczywiście, by był duży.
- Rozważmy najpierw proste przypadki. Czy gdy $F = [0,1]$, to Banach posiada strategię wygrywającą?
- Tak, posiada. Wystarczy, by w każdy ruchu wybierał odcinek, który istotnie obcina oba końce. Wówczas lewe końce I_n będą dążyć do pewnej granicy L, prawe końce do pewnej granicy P, oczywiście $L \leqslant P$ oraz dla każdego n prawdą będzie, że $[L,P] \subset I_n$, czyli innymi słowy $[L,P] \subset \bigcap_{n=1}^{\infty} I_n$.
- Zauważmy przy okazji, że Mazur może zapewnić, że $\bigcap_{n=1}^{\infty} I_n$ będzie co najwyżej jednopunktowe, zwyczajnie na przykład skracając za każdym swoim ruchem odcinek przynajmniej 2 razy.

- Czy Banach może wygrać, gdy $F = [0,1] \setminus \{q\}$, gdzie q jest pewnym punktem z $[0,1]$? Tak, wystarczy, że stosuje poprzednią zasadę obcinania końców i za pierwszym ruchem ruszy się tak, by dalej rozważany kawałek nie zawierał punktu q.
- A gdy F to $[0,1]$ bez skończonej liczby punktów? Analogicznie skończoną liczbę punktów można wyrzucić za jednym ruchem.
- A gdy F to liczby niewymierne?
- Też Banach może wygrać, ustawiamy liczby wymierne, czyli te, które Banach chce ominąć, w ciąg. Za pierwszym ruchem Banach rusza się tak, by wyrzucić pierwszą liczbę wymierną, za drugim drugą, za trzecim trzecią itd. W przecięciu nie pojawi się żadna liczba wymierna, bo każda liczba wymierna stała na pewnym n-tym miejscu w ciągu, więc od ruchu n-tego w górę już na pewno nie należała od odcinków I_k, czyli nie należy też do nieskończonego przecięcia.
- W ten sposób możemy wyrzucać wszystkie zbiory przeliczane. Ale czy możemy wyrzucać pewne zbiory nieprzeliczalne? Okazuje się, że tak.
- Rozważmy F równy $[0,1]$ bez zbioru Cantora. Wówczas po pierwszym ruchu Mazura Banach może wybrać sobie taki przedział, który już w całości siedzi w F i wygrał (o ile stosuje swoją stałą zasadę, czyli obcinanie końców). Czyli w jednym ruchu można wyrzucić nawet zbiór nieprzeliczalny!
- **Dalsza część odbywała się już na piątych ćwiczeniach**
- Przyjrzyjmy się jaka własność zbioru Cantora pozwoliła na to, by Banach ominął go w jednym ruchu.
- Tę właściwością jest to, że w każdym przedziale jest taki podprzedział, że w tym podprzedziale w ogóle zbioru Cantora nie ma. Tu warto wprowadzić definicję zbioru gęstego, zbiór nazywamy *gęstym* jeśli w każdym przedziale jest element tego zbioru. Zbiór Cantora ma własność w pewnym sensie przeciwną, nie jest on gęsty, a co więcej w żadnym przedziale nie jest on gęsty. Zbiory o tej własności, czyli takie, że na żadnym przedziale nie są one gęste nazywamy zbiorami *nigdziegęstymi*.
- A zatem Banach w jednym ruchu jest w stanie ominąć zbiór nigdziegęsty. Czyli w ogólności może on ominąć zbiory będące przeliczalną sumą zbiorów nigdziegęstych. Takie zbiory nazywane są zbiorami *I kategorii*.
- Przyjrzyjmy się dla jakich zbiorów F wygrywa Mazur. Co będzie, gdy $F = \mathbb{Q}$?
- Wówczas Mazur będzie mógł zastosować strategię Banacha dla liczb niewymiernych i ominąć po prostu cały zbiór liczby wymiernych - czyli dla $F = \mathbb{Q}$ wygrywa Mazur.
- Czy Mazur wygrywa jeszcze dla jakichś większych zbiorów?
- Tak, gdy F jest zbiorem I kategorii to Mazur może zastosować strategię wyżej opisaną i ominąć cały zbiór F.
- Mamy więc teraz pokazane, że dla $F = [0,1] \setminus$ zbiór I kategorii wygrywa Banach, dla $F =$ zbiór I kategorii wygrywa Mazur. Jednak nie wszystkie podzbiory $[0,1]$ są I kategorii lub ich dopełnienie jest I kategorii, co wówczas?

- Okazuje się, że wówczas wygrywa Mazur, czyli Banach wygrywa grę wtedy i tylko wtedy, gdy zbiór F jest całym odcinkiem $[0,1]$ z wyrzuconym zbiorem I kategorii. Ale tego ja nie umiem dowieść, może komuś się uda. Choć pytanie to było w księdze Szkockiej, może jednak nie jest to tak trudne do dokończenia.

5 Ćwiczenia 5

16 marca 2009

1. Pebbling game. Ktoś pokazuje rozwiązanie zadania domowego **Pebbling game**. W tym roku nikt nie znalazł jak na razie lepszego wyniku niż n kamieni potrzebnych do grafu o $\Theta(\frac{n^2}{2})$ wierzchołkach.

2. Determinacja gier. Najpierw przypomnienie faktu z wykładu o tym, kiedy w ogólności mówimy, że gra jest zdeterminowana. Niech S_A zbiór strategii Adama, S_E zbiór strategii Ewy. Zakładamy, że Adam chce zmaksymalizować wynik gry, Ewa chce zminimalizować. Zawsze prawdziwa jest nierówność

$$\max_{a\in S_A}\min_{e\in S_E}\Phi(e,a) \leqslant \min_{e\in S_E}\max_{a\in S_A}\Phi(e,a).$$

Jeśli odpowiednio spojrzymy na tę nierówność, to wyda się ona oczywista. Zauważmy, że prawa strona jest wartością gry, jaką może sobie zapewnić Adam, wybiera strategię $a \in S_A$ taką, że przy tym wyborze najgorsza (dla Adama) strategia Ewy $e \in S_E$ daje w efekcie największą wartość $\Phi(e,a)$. Czyli Adam może zapewnić, że wynik gry będzie nie mniejszy niż lewa strona nierówności. Analogicznie możemy pokazać, że Ewa może zapewnić, że wynik gry będzie nie większy niż prawa strona nierówności. Zatem mamy $L \leqslant$ wynik gry $\leqslant P$ co dowodzi nierówności. Usprawiedliwia to również rozszerzoną definicję determinacji gry, gdy

$$\max_{a\in S_A}\min_{e\in S_E}\Phi(e,a) = \min_{e\in S_E}\max_{a\in S_A}\Phi(e,a),$$

to mówimy, że gra jest zdeterminowana. Jasne - wtedy po prostu wynik gry ma tylko jedną opcję (gdy gracze grają optymalnie), być równy lewej i prawej stronie.

Rozpatrywaliśmy do tej pory cały czas gry z pełną informacją, takie, gdzie

- wynik gry należał do zbioru $\{0,1\}$ (gracz wygrał, bądź przegrał)
- ruchy wykonywane były naprzemiennie przez graczy, nigdy naraz

Wówczas przykład gry niezdeterminowanej był bardzo skomplikowany, a gry skończone w oczywisty sposób były zdeterminowane. Czy ustąpienie z któregoś z powyższych założeń pozwoli nam na skonstruowanie prostej (w szczególności może skończonej) gry niezdeterminowanej (w nowym sensie, który jest spójny ze starym zresztą)?

Wskazówki

- Najpierw zrezygnujmy z $\Phi(e,a) \in \{0,1\}$. Co to da?
- Niestety nic, gdy zbudujemy drzewo gry, to wartość drzewa gry jest równa lewej i prawej stronie nierówności, czyli gra jest zdeterminowana.

- Pozwólmy więc na równoległość. Czy to coś da?
- Tak, można skonstruować proste gry niezdeterminowane korzystając z założenia, że gracze mogą się ruszać równolegle. Jedną z takich gier jest gra papier-nożyce-kamień. Można skonstruować zresztą jeszcze prostszą grę, w której Adam ma dwie strategie $a = 0$ oraz $a = 1$, Ewa ma dwie strategie $e = 0$ i $e = 1$, a wynikiem gry jest $e \oplus a$. Wówczas lewa strona to 0, a prawa 1 i gra jest niezdeterminowana. Widać, że równoległość daje bardzo dużo.

Wracamy jednak do gier z naprzemiennymi ruchami.

3. Przykład gry niezdeterminowanej. W tym zadaniu użyjemy istnienia funkcji będącej „nieskończonym xorem", której to istnienie jest treścią zadania domowego nr 3. Zakładamy więc, że istnieje funkcja $f : \{0,1\}^\omega \to \{0,1\}$ taka, że samym zerom przyporządkowuje wynik równy 0 oraz, że przy zmianie bitu argumentu zmienia się wynik. Używając tej funkcji skonstruuj grę niezdeterminowaną dla dwóch graczy ruszających się na przemian.

Uwaga
Zazwyczaj naturalną propozycją tutaj jest następująca gra. Adam i Ewa ruszają się na zmianę i ich ruch polega na dopisaniu kolejnego bitu, wygrywa Adam, jeśli wynik funkcji f na stworzonym ciągu bitów jest zero, w przeciwnym przypadku Ewa. Nie jest jednak jasne, że zaproponowana gra jest niezdeterminowana, ja przynajmniej nie umiem dowieść, że niezależnie od funkcji f (o ile ma ona własność bycia nieskończonym xorem) jest to prawdą. Błędem, który łatwo tu popełnić jest próba dowodzenia metodą strategy-stealing i skorzystanie implicite z faktu, że $f(x) = f(0x)$, który wcale niekoniecznie jest prawdą (jest on prawdą dla xora skończonego liczącego ilość jedynek).

Wskazówki

- Gracze muszą jakoś ustalać bity.
- Jeśli pada propozycja z uwagi powyżej to usiłujemy skonstruować dowód, pokazuję jaki jest haczyk.
- Ogólnie warto zastosować metodę strategy-stealing, podobnie jak na wykładzie w przykładzie dla ultrafiltrów.
- Problem jaki napotykamy jest taki, że jeśli powiedzmy Adam zrobił ruch, a po nim Ewa zrobiła ruch i doszła do pozycji p, to Adam nie mógł od razu zrobić ruchu takiego, żeby dojść do pozycji p
- Sposobem w jaki możemy to ominąć jest pozwolenie na wybranie dowolnego skończonego ciągu bitów przez gracza. Taka gra jest niezdeterminowana co łatwo pokazać metodą strategy-stealing.

4. Topologia Na tych ćwiczeniach rozpocząłem wprowadzanie topologii w grach, jednak większa część tego działu odbyła się na szóstych ćwiczeniach, w związku z czym opiszę na w odcinku dotyczącym ćwiczeń szóstych.

6 Ćwiczenia 6

23 marca 2009

1. Rozstrzygalność problemu dla $FO(\leqslant)$. Ktoś pokazuje rozwiązanie zadania domowego **Rozstrzygalność problemu dla** $FO(\leqslant)$. W skrócie chodzi o to, że dla konkretnego k aby sprawdzić czy formuła φ jest spełniona w $(\langle 1, 2, \ldots, k\rangle, \leqslant)$ wystarczy brutalnie przeszukać całą przestrzeń możliwości. Drugie spostrzeżenie to fakt, że dla formuły φ o randze kwantyfikatorowej m wystarczy sprawdzić modele dla k od 1 do około 2^m, dalej już jest tak samo.

2. Topologia, teoria. Najpierw wstęp. Gry mają dużo wspólnego z topologią, w szczególności używając topologii możemy pokazywać, że pewne gry są z pewnością zdeterminowane. Z doświadczenia wiemy, że gry (dla ruchów naprzemiennych) niezdeterminowane są zwykle jakieś skomplikowane, a proste gry są zwykle zdeterminowane. By sformalizować tę intuicję sięgniemy do topologii.

Najpierw trzeba zdefiniować przestrzeń. Rozważmy grę nieskończoną na arenie X (arena to zbiór możliwych pozycji). Zauważmy, że każdą grę skończoną możemy przeformułować do równoważnej gry nieskończonej po prostu dodając dwa zapętlone stany: przegrana Ewy i przegrana Adama i ze stanów, z których nie ma już wyjścia dodać przejścia do odpowiedniego z tych stanów. Możemy więc zakładać, że wszystkie rozgrywki są nieskończone, czyli są nieskończonymi ciągami pozycji z X, czyli innymi słowy należą do zbioru X^ω. Tak, jak to w grach nieskończonych, aby powiedzieć kiedy który gracz wygrywa musimy określić zbiór W rozgrywek wygrywających dla (powiedzmy) Ewy. Określamy zatem $W \subseteq X^\omega$. Zauważmy, że w tej sytuacji zbiór W mówi wszystko o warunku wygrywającym. Będziemy mówić, że gra jest skomplikowana, jeśli zbiór W jest skomplikowany i przeciwnie. W tym celu jednak, aby powiedzieć kiedy zbiór W jest skomplikowany musimy zdefiniować topologię (czyli powiedzieć które zbiory uważamy za otwarte w X^ω), a wcześniej w tym celu zdefiniować odległość na rozgrywkach, czyli elementach X^ω. Zauważmy, że to, co my teraz robimy to tak naprawdę są rozważania dotyczące języków słów nieskończonych nad alfabetem X, właściwie lepiej byłoby nawet tymi pojęciami operować.

Niech $s, t \in X^\omega$. Wówczas definiujemy odległość między s i t jako $\delta(s,t) = \frac{1}{2^k}$, gdzie k to indeks pierwszej różnicy w ciągach s i t. Czyli na przykład $\delta(0100\ldots, 0101\ldots) = \frac{1}{2^3} = \frac{1}{8}$. Gdy już mamy odległość (metrykę), to możemy zdefiniować co uważamy za zbiory otwarte i domknięte. Zbiór G jest *otwarty*, gdy dla dowolnego $x \in G$ istnieje takie $r > 0$, że kula o środku w x i promieniu r jest zawarta w G (innymi słowy jeśli jakiś punkt leży w zbiorze otwartym, to leży tam również pewne jego otoczenie, jest to zgodne z naszą intuicją z prostej rzeczywistej). Zbiór F jest *domknięty*, gdy dla dowolnego ciągu zbieżnego $x_1, x_2, \ldots$ jeśli dla każdego i punkt x_i leży w F, to również granica tego ciągu leży w F. Zbiory domknięte są dopełnieniami zbiorów otwartych. Zbiory otwarte i domknięte to w pewnym sensie przyzwoite zbiory.

Pytanie pomocnicze: czym w rozważanej metryce będzie kula o środku w $x_0x_1x_2\ldots$ i promieniu równym $\frac{1}{2^n}$. Odpowiedź: będzie ona zawierała wszystkie ciągi o prefiksie $x_0x_1\ldots x_n$.

Przyjrzyjmy się trochę dokładniej strukturze przyzwoitych zbiorów. Suma (dowolna) zbiorów otwartych jest zbiorem otwartym, podobnie przecięcie zbiorów domkniętych. Przecięcie skończone zbiorów otwartych jest otwarte, jednak już przecięcie przeliczalne zbiorów otwartych może otwarte nie być (przykładem niech będzie rodzina odcinków $(-\frac{1}{n}, \frac{1}{n})$, których przecięciem jest

$\{0\}$). Przecięcie przeliczalnej rodziny zbiorów otwartych nazwiemy zbiorem typu G_δ, inaczej Π^0_2, sumę przeliczalnej rodziny zbiorów domkniętych zbiorem typu F_σ, inaczej Σ^0_2. Można iść dalej, przeliczalne przecięcia zbiorów typu Π^0_2 są Π^0_2, ale sumy już nie, są to zbiory Σ^0_3. Idąc dalej po pewnej liczbie kroków (równej pewnej liczbie porządkowej) otrzymamy wszystkie zbiory borelowskie. Powiemy, że klasa zbiorów *borelowskich* to najmniejsza klasa zbiorów zawierająca zbiory otwarte oraz zamknięta na przeliczalne sumy i dopełnienie (z tego już wynika zamkniętość na przeliczalne przecięcia). Właśnie zbiory borelowskie będą naszymi zbiorami przyzwoitymi. Całą tę teorię wprowadzaliśmy, by móc sformułować twierdzenie Martina (udowodnione w roku 1975), mówiące, że

Twierdzenie Martina (1975)
Każda gra borelowska (czyli z borelowskim zbiorem W definiującym warunek wygrywający) jest zdeterminowana.

Dowodzić tego twierdzenia nie będziemy, dowód jest zapewne porządnie skomplikowany, jednak korzystając z niego możemy łatwo pokazywać determinację (będącą bardzo przydatną cechą gier) dla naprawdę wielu gier.

Przejdźmy do zadań.

3. Czy zbiory domknięte są G_δ Rozstrzygnij, czy w rozważanej przestrzeni z rozważaną metryką każdy zbiór domknięty jest typu G_δ.

Wskazówki

- Przypomnijmy sobie przykład z odcinkami, których przecięcie było punktem $\{0\}$
- Czy można podobną konstrukcję zrobić z dowolnym zbiorem domkniętym?
- Tak.
- Oznaczymy $\delta(x, F) = \inf_{y \in F} \delta(x, y)$. Niech $F_\varepsilon = \{x : \delta(x, F) < \varepsilon\}$. Zbiór F_ε jest otwarty dla dowolnego ε. Zauważmy, że $F = \bigcap_{n \in \mathbb{N}} F_{\frac{1}{n}}$. Zatem dowolny zbiór domknięty jest przecięciem przeliczalnej rodziny zbiorów otwartych, czyli zbiorem typu G_δ.
- A gdzie tu była wykorzystana domkniętość zbioru F?
- W tym, że $F = \bigcap_{n \in \mathbb{N}} F_{\frac{1}{n}}$, bo jeśli pewien punkt jest dowolnie blisko zbioru domkniętego, to jest też w nim.

4. Charakteryzacja zbiorów otwartych. Udowodnić, że następujące warunki są równoważne dla języka $L \subseteq X^\omega$

- L jest otwarty
- istnieje zbiór $M \subseteq X^*$ (gdzie przez X^* rozumiemy zbiór wszystkich słów skończonych nad X) taki, że $L = \bigcup_{v \in M} vX^\omega$, przy czym możemy M wybrać tak, by był antyłańcuchem (antyłańcuchem nazywamy zbiór, w którym żadne dwa elementy nie są porównywalne, tu względem relacji bycia prefiksem). Przez vX^ω rozumiemy wszystkie słowa rozpoczynające się prefiksem v.

Wskazówki

- Najpierw zróbmy implikację w lewo.
- Zauważmy, że każdy zbiór vX^ω jest otwarty (łatwo pokazać wprost, lub argumentując, że jest kulą). Zatem zbiór L jest otwarty jako suma kul.
- Teraz implikacja w prawo, nieco trudniejsza, trzeba jakoś zdefiniować ten zbiór M.
- Zauważmy, że ponieważ zbiór L jest otwarty, to dla każdego słowa u istnieje pewne otoczenie, z którym on razem siedzi w L. Czyli istnieje taki indeks n_u, że $(u|n_u)X^\omega \subseteq L$, gdzie przez $u|n$ oznaczymy obcięcie słowa u do pierwszych n liter.
- Gdy wysumujemy te słowa po wszystkich u, to otrzymamy żądany zbiór M, czyli $M = \{u|n_u : u \in L\}$. Gdy jeszcze weźmiemy minimalne takie n_u dla każdego u, to otrzymamy M będące antyłańcuchem.

5. Gra n-ty stan ustalony. Rozważmy grę między Adamem, a Ewą na grafie gry, rozmiaru co najwyżej przeliczalnego, w której Ewa wygrywa rozgrywkę, gdy rozgrywka w n-tym ruchu będzie w stanie q, gdzie n oraz $q \in X$ są ustalone. Czy rozważana gra jest zdeterminowana? Rozstrzygnąć to opisywanymi wyżej metodami.

Wskazówki

- Spróbujmy rozstrzygnąć, czy rozważany warunek wygrywający jest borelowski.
- Pokażmy, że jest on otwarty.
- Jest on tak naprawdę sumą kul - dowolny prefiks $n-1$ literowy, na n-tej współrzędnej litera q, a potem wszystko jedno. Jako suma kul jest to zbiór otwarty, czyli gra z twierdzenia Martina jest zdeterminowana.

6. Gra przechodząca przez ustalony stan. Rozważmy grę między Adamem, a Ewą na grafie gry podobnie jak poprzednio. Ewa wygrywa, gdy rozgrywka kiedykolwiek przejdzie przez ustalony stan q. Rozstrzygnąć, czy rozważana gra jest zdeterminowana.

Wskazówki

- Skorzystać z poprzedniego zadania.
- Niech Q_n - zbiór określający, że n-ty stan to q. Wiemy, że Q_n otwarty. My rozważamy zbiór $Q = \bigcup_{n\in\mathbb{N}} Q_n$, mówiący, że nasza rozgrywka w pewnym stanie jest w q, czyli należy do pewnego Q_n. Zatem Q jako suma zbiorów otwartych jest otwarty, czyli gra jest zdeterminowana.

7. Gra przechodząca nieskończenie wiele razy przez ustalony stan. Rozważmy grę między Adamem, a Ewą na grafie gry podobnie jak poprzednio. Ewa wygrywa, gdy rozgrywka przechodzi nieskończenie wiele razy przez pewien ustalony stan q. Rozstrzygnąć, czy rozważana gra jest zdeterminowana.

Wskazówki

- Skorzystać z poprzedniego zadania.
- Zapisać warunek wygrywający w postaci kwantyfikatorów.
- Formuła będzie miała postać $\forall_{k\in\mathbb{N}}\exists_{n>k}$ w n-tym ruchu jesteśmy w q.
- Przerobić formułę na przecięcia i sumy zbiorów.
- Dla każdego przerabiamy na przecięcie, istnieje na sumę, więc otrzymujemy $\bigcap_{k\in\mathbb{N}}\bigcup_{n>k} Q_n$, czyli przeliczalne przecięcie sum zbiorów otwartych, czyli zbiór typu G_δ, w każdym razie borelowski, czyli gra jest zdeterminowana.
- W oczywisty sposób jeśli powiemy, że gra ma przechodzić nie przez ustalony stan q nieskończenie wiele razy, ale przez któryś ze zbioru dobrych stanów F nieskończenie wiele razy, to dostajemy alternatywę rozważanych warunków, czyli sumę zbiorów G_δ, czyli zbiór również borelowski. Tak jest zdefiniowana znana gra Büchiego - jeden z graczy wygrywa, gdy rozgrywka przechodzi nieskończenie wiele razy przez któryś z „dobrych" stanów, mówi się na ten warunek warunek Büchiego (zauważmy, że tu trzeba wykorzystać przeliczalność zbioru stanów, bo inaczej suma mogłaby być nieprzeliczalna).

8. Gra parzystości. Zdefiniujmy grę parzystości. Klasa tych gier jest bardzo znana i bardzo szeroko rozważana, w szczególności wielkim problemem otwartym jest, czy istnieje wielomianowy algorytm rozstrzygający dla danej pozycji w grze parzystości, który gracz posiada strategię wygrywającą.

Gra odbywa się na grafie skierowanym $G = (V, E)$, załóżmy, że $|G| < \infty$, choć nie trzeba aż tak mocnego założenia zawsze. Grają gracze Odd i Even, ruszają się na przemian zgodnie z dozwolonymi przejściami w tym grafie. Dodatkowo dana jest funkcja $rank : V \to \mathbb{N}$ przyporządkowująca każdemu wierzchołkowi pewną liczbę naturalną. Gracz Even wygrywa rozgrywkę, jeśli największy rank powtarzający się nieskończenie wiele razy na tej rozgrywce jest parzysty, w przeciwnym wypadku wygrywa gracz Odd. Czyli innymi słowy patrzymy, czy $\limsup rank(v)$ dla v występujących na rozgrywce jest parzysty, czy nieparzysty. Rozstrzygnąć, czy gra ta jest zdeterminowana.

Wskazówki

- Działamy metodą podobną jak poprzednio. Określmy najpierw zbiór, w którym największym występującym rankiem będzie ustalone j.
- Dzielimy zbiór V na zbiory $V_1, V_2, \ldots, V_n$ dla odpowiednich ranków. Niech formuła M_j mówi, że przechodziliśmy nieskończenie wiele razy przez pewien wierzchołek z V_j, a dla wierzchołków z V_k dla $k > j$ nie przechodziliśmy przez nie nieskończenie wiele razy. Taka formuła przekłada się na zbiór borelowski.
- Następnie wystarczy powiedzieć, że poszukiwany zbiór to suma $\bigcup_{1\leqslant k\leqslant n, 2|k} M_k$.

7 Ćwiczenia 7

30 marca 2009

1. Nieskończony xor. Ktoś pokazuje rozwiązanie zadania domowego **Nieskończony xor**. Istnieją dwie metody rozwiązywania tego zadania, przez lemat Kuratowskiego-Zorna oraz przez zdefiniowanie pewnej relacji. Niech $X = \{0,1\}^\omega$. Jest to relacja $\sim \subseteq X \times X$, $x \sim y$, gdy różnią o skończoną ilość elementów. Następnie na każdej klasie abstrakcji relacji $\sim$ określamy f oddzielnie.

Warto zauważyć, że metoda przez relację $\sim$ jest prostsza, ale metoda przez lemat K.-Z. jest w pewnym sensie ogólniejsza. Da się tą metodą pokazać, że istnieją funkcje f spełniające dodatkowo warunek $f(x \oplus y) = f(x) \oplus f(y)$ (nakładamy ten warunek na definiowaną relację porządku).

Ciekawym ćwiczeniem jest policzenie mocy zbioru funkcji typu xor nieskończony, wychodzi $2^{\mathcal{C}}$ (na każdej klasie abstrakcji $\sim$ możemy określić na dwa sposoby) oraz policzenie mocy zbioru funkcji typu xor nieskończony z dodatkowym warunkiem (z powyższego akapitu), co zostało wpisane jako zadanie domowe nr 4.

2. Atraktory. Ten paragraf to wprowadzenie pojęcia. Rozważmy grę na arenie V. Niech $X \subseteq V$. Zdefiniujmy $attr_E(X)$ (odp. $attr_A(X)$) jako zbiór wierzchołków $x \in V$ takich, że będąc w x Ewa (odp. Adam) może zmusić rozgrywkę do wkroczenia (przynajmniej raz) do X.

Warto zrobić ćwiczenie, które pokazuje, że $attr_E(X \cup Y)$ może być istotnie większy od $attr_E(X) \cup attr_E(Y)$. Po prostu może być wierzchołek, w którym Ewa może zmusić rozgrywkę do wejścia do $X \cup Y$, ale do żadnego z nich osobno nie może.

3. Obliczanie atraktora. Warto zrobić takie ćwiczenie, choć nie było ono explicite na ćwiczeniach. Znaleźć algorytm wielomianowy obliczający atraktor.

Wskazówki

- Najpierw myślimy.
- Zastanówmy się na początek co na pewno należy do atraktora.
- Trzymajmy na pewnej zmiennej to, co należy do atraktora i uaktualniajmy ją. Powiedzmy, że liczymy $attr_E(X)$. Zróbmy na początek $A := X$, to, co leży w A to już na pewno będzie w atraktorze. Do atraktora na pewno należą wszystkie wierzchołki x takie, że jeśli należą do Ewy, to istnieje z nich krawędź do A, a jeśli należą do Adama, to wszystkie krawędzie idą do A. Taki zbiór zsumowany z A nazwijmy $exp(A)$. Wiemy więc, że do atraktora $attr_E(X)$ na pewno należy $exp(A)$, możemy więc napisać $A := exp(A)$.
- Możemy tak robić aż do osiągnięcia punktu stałego, czyli piszemy tak naprawdę

```
A:=X;
while (A sie zmienia) {
  A:=exp(A);
}
return A;
```

- Należy się zastanowić dlaczego ten algorytm jest dobry. Na pewno wszystkie wierzchołki w zwróconym A będą należeć do atraktora Ewy od X. Zastanówmy się dlaczego żaden inny wierzchołek nie będzie należeć do atraktora Ewy.

- Pokażemy jak Adam może z dopełnienia A w nieskończoność unikać wejścia do A, czyli w efekcie nigdy tam nie wejść. W każdym wierzchołku Ewy z $\bar{A}$ wszystkie krawędzie prowadzą do $\bar{A}$, czyli tam rozgrywka nie wejdzie do A. W każdym wierzchołku Adama z $\bar{A}$ istnieje krawędź nie wchodząca do A, tam właśnie będzie ruszał się Adam i w ten sposób uniknie wejścia do A. Czyli faktycznie algorytm jest poprawny.

- Policzmy w jakim działa on czasie. Faz jest maksymalnie $n = |V|$, bo w każdej dochodzi do A nowy wierzchołek. Każda z nich wymaga sprawdzenia, czy nowy wierzchołek wchodzi do atraktora, czyli de facto przejrzenia wszystkich krawędzi wszystkich wierzchołków, czyli $O(m = |E|)$ czasu. Czyli algorytm działa w czasie $O(nm)$.

- Znajdźmy szybszy algorytm. Będzie on działał na podobnej zasadzie, tylko implementacja będzie inna.

- Można zrealizować to samo odwracając strzałki w grafie i przejść potrzebne wierzchołki jednym BFS-em, czyli czas wyniesie $O(m)$.

4. Gra Büchiego. Zdefiniujmy grę Büchiego, jest to tak naprawdę definiowana już gra parzystości, tylko dla dwóch ranków, 0 oraz 1. Można to powiedzieć nieco inaczej. Na skończonym grafie grają Adam i Ewa, jest pewien zbiór $F \subseteq V$ wierzchołków dobrych. Ewa wygrywa rozgrywkę, jeśli ta rozgrywka przechodzi nieskończenie wiele razy przez pewien wierzchołek dobry (co dla skończonej ilości tych wierzchołków jest równoważne temu, że przechodzi nieskończenie wiele razy przez F), w przeciwnym wypadku wygrywa Adam. Zadanie polega na znalezieniu wielomianowego algorytmu rozwiązującego gry Büchiego, czyli dla każdego wierzchołka odpowiadającego na pytanie, który z graczy posiada strategię wygrywającą z tego wierzchołka (któryś ma, bo gra jest zdeterminowana, co zostało udowodnione na jednych z poprzednich ćwiczeń).

Wskazówki

- Najpierw się zastanawiamy, testujemy różne opcje. Jeśli ktoś proponuje algorytm, to patrzymy, czy jest dobry, a jeśli nie, to próbujemy go obalić.

- Spróbujmy wziąć pewien zbiór, który na pewno będzie zawierał wszystkie punkty, z których wygrywa Ewa, a potem go zmniejszać, do odpowiedniego.

- Na pewno, jeśli z któregoś wierzchołka wygrywa Ewa, to musi z niego móc dojść do zbioru F, czyli zbiór wierzchołków wygrywających dla Ewy zawiera się w zbiorze $attr_E(F)$. $attr_E(F)$ to zbiór wierzchołków, z których mogę sobie zapewnić przynajmniej jedną jedynkę na ścieżce.

- Jak go teraz zmniejszać?

- Spróbujmy jakoś znaleźć te wierzchołki, z których mogę (jako Ewa) zapewnić sobie przynajmniej dwie jedynki na ścieżce. To będą te wierzchołki, z których będę mógł przechodząc przez F dojść w niezerowej liczbie ruchów do wierzchołków, z których mogę uzyskać przynajmniej jedną jedynkę.

- Potrzebuję zatem czegoś, co zachowuje się jak atraktor, ale wymusza zrobienie przynajmniej jednego ruchu (do bycia w $attr_E(X)$ wystarcza bycie w X, u nas nie chcemy, żeby tak było). Zdefiniujmy więc $attr_E^+(X)$ jako zbiór tych wierzchołków, z których Ewa może zmusić rozgrywkę, żeby robiąc przynajmniej jeden ruch doszła do X. Liczymy to podobnie, jak poprzednio, tylko nie wrzucamy za darmo X do dobrych wierzchołków (można się zastanowić jak dokładnie).

- Zatem zbiór tych wierzchołków, z których jestem w stanie osiągnąć minimum dwie jedynki to $attr_E^+(F \cap attr_E^+(F))$ (nie wystarczy $attr_E^+(F \cap attr_E(F))$, który różni się od poprzedniego brakiem wewnętrznego plusa, gdyż tu możemy tę samą jedynkę policzyć dwukrotnie, bycie w $attr_E^+(F)$ zapewnia bycie w tym).

- Gdy chcemy uzyskać kolejną jedynkę po prostu jeszcze raz aplikujemy funkcję $X \mapsto attr_E^+(X \cap F)$.

- Algorytm zatem będzie wyglądać następująco

```
R:=attr_E^+(F);
while (R sie zmienia) {
  R:=attr_E^+(R \cap F);
}
return R;
```

- Czyli tak naprawdę nasz algorytm znalazł największy punkt stały przekształcenia $X \mapsto attr_E^+(X \cap F)$.

- Dlaczego ten algorytm jest poprawny?

- Z pewnością z tych wierzchołków da się osiągnąć dowolnie wiele jedynek, czyli są wygrane dla Ewy.

- Dlaczego jednak z pozostałych wygrywa Adam?

- Dlatego, że dla pewnego k nie da się z nich zapewnić sobie przynajmniej k jedynek (w którymś kroku ten wierzchołek odpadł), czyli Adam może grać tak, że będzie jedynie $k-1$ jedynek, czyli wygra.

- W jakim czasie działa algorytm?

- Kroków jest co najwyżej $n = |V|$. W każdym z nich obliczamy atraktor (z plusem, ale to niewiele zmienia), który może być liczony w czasie $O(m = |E|)$, zatem algorytm działa w czasie $O(nm)$. Być może nie jest najszybszy, tego nie twierdzę.

8 Ćwiczenia 8

6 kwietnia 2009

1. Barman. Można opowiedzieć historię zadania. Marcin Jurdziński bodajże opowiedział ją kiedyś na Gamesach przy okazji zakładając się, obiecał temu, kto zgadnie rozwiązanie 2 drinki. Michał Strojnowski zgadł. Ja zrobiłem podobne zakłady, każdy mógł obstawić jeden wynik, z tych jeszcze nie zajętych.

Barman proponuje klientowi zakład. Barman obstawia jedną z dwóch liczb 1 lub 2. Podobnie klient. Jeśli klient trafi tę samą liczbę k, którą obstawia barman, to barman stawia klientowi k drinków (czyli jeden albo dwa). W przeciwnym razie klient nie dostaje żadnego drinka, ale płaci za x drinków. Pytanie brzmi: dla jakiego x ta gra jest sprawiedliwa.

Wskazówki

- Można zrobić tak, że teraz rozpocząć obstawianie, a w międzyczasie zrobić zadania 2 oraz 3. Dopiero potem rozwiązać to zadanie. Ja tak zrobiłem.
- Trzeba się zastanowić co to znaczy, że gra jest sprawiedliwa.
- Powinno być tak, że średnia wypłata wynosi 0. Pytanie - w jakiej sytuacji. Sensownie jest założyć, że ta sytuacja to równowaga Nasha. Raczej nie ma co wdawać się w dyskusje, czy to jest naprawdę naturalna sytuacja do której zbiegamy i w jakim sensie.
- Każdy z graczy ma dwie strategie, pierwszą - obstawianie 1 oraz drugą - ostawianie 2. Niech Barman obstawia 1 z prawdopodobieństwem p, a klient z prawdopodobieństwem q.
- Zauważmy, że jeśli $((p, 1-p), (q, 1-q))$ to równowaga Nasha (niekoniecznie w strategiach czystych), to każdy z graczy może zmienić swoją strategię na którąś ze strategii czystych występujących z niezerowym prawdopodobieństwem w jego aktualnej strategii mieszanej i wówczas wynik gry będzie taki sam. Oczywiście może nie być to równowaga Nasha i potem może gra pójść w jakimś dziwnym kierunku. Jest to fakcik z wykładu.
- Chcemy więc, żeby wartość gry, gdy klient obstawi pierwszą strategię była równa wartości, gdy klient obstawi drugą. Prowadzi to do równania $1 \cdot p + (-x) \cdot (1-p) = (-x) \cdot p + 2 \cdot (1-p)$.
- Czy mamy jeszcze jakieś równanie?
- Tak, wartość gry ma być równa 0, bo gra jest sprawiedliwa, czyli do powyższych równości dopisujemy $= 0$.
- Wychodzi z tego, że $x = \sqrt{2}$.
- A co byłoby w ogólności, gdy zamiast 1 i 2 wstawimy a i b?
- Wyjdzie podobnie $\sqrt{ab}$, można to nietrudno pokazać, bo musi być

$$-xp + b(1-p) = ap - x(1-p) = 0 \qquad (1)$$

z czego wynika $0 = ap - x(1-p) \overset{(1)}{=} (ap - x(1-p)) \cdot \frac{b}{-x} + p(\frac{ab}{x} - x) = 0 + p(\frac{ab}{x} - x)$, czyli $ab = x^2$, co daje $x = \sqrt{ab}$.

2. Gra trzech graczy. Niech $Pos = Pos_0 \cup Pos_1 \cup Pos_2$, gdzie Pos_i są rozłączne. Rozważamy grę trzech graczy, którzy nazywają się 0, 1 i 2. Pos_i to pozycje gracza i dla $i = 0, 1, 2$. Z każdego wierzchołka istnieje przynajmniej jeden ruch, czyli każda rozgrywka jest nieskończona. Gracz i wygrywa rozgrywkę $\pi = (p_0, p_1, p_2, \ldots)$, gdy $\limsup_{n\to\infty} rank(p_n) = i \pmod 3$, czyli gdy największy rank występujący nieskończenie często daje resztę i modulo 3. Rozstrzygnąć, czy jest prawdą, że dla każdej pozycji jeden z graczy posiada z tej pozycji strategię wygrywającą (czyli, czy gra jest zdeterminowana).

Wskazówki

- Trzeba sprawdzić, czy gracz może wygrać jeśli pozostali dwaj sprzymierzą się przeciwko niemu.
- To nie jest gra zdeterminowana, spróbować znaleźć kontrprzykład.
- Może być na przykład taki: 3 wierzchołki, z każdego da się przejść do każdego innego, ale do siebie nie, pierwszy jest gracza nr 0 i ma rank 0, drugi jest gracza nr 1 i ma rank 1, a trzeci jest gracza nr 2 i ma rank 2. Wówczas żaden z graczy nie jest w stanie zatrzymać gry w swoim wierzchołku, a wtedy przegrywa.

3. Algorytm dla gry trzech graczy. Zaprojektować algorytm, który rozstrzyga dla każdego wierzchołka, czy któryś gracz posiada z niego strategię wygrywającą, a jeśli tak, to który. Załóżmy, że możemy używać jako podalgorytmu algorytmu rozwiązywania gry parzystości.

Wskazówki

- Trzeba się zastanowić kiedy gracz ma strategię przeciwko dwóm pozostałym graczom grającym wspólnie.
- Można połączyć pozostałych graczy w jednego gracza.
- Zredukować do gry parzystości.
- Dla każdego wierzchołka sprawdzamy, czy gracz i może z niego wygrać, gdzie i przebiega zbiór $\{0, 1, 2\}$. Dla gracza i łączymy pozostałych dwóch i modyfikujemy ranki. Konkretnie - wierzchołki gracza i dajemy graczowi Odd, wierzchołki pozostałych graczy dajemy graczowi Even. Ranki zmieniamy następująco $3n+i \mapsto 2n+1$ oraz $3n+j \mapsto 2n+2\cdot\mathbb{1}_{\{j>i\}}$ dla $j \in \{0, 1, 2\} \setminus \{i\}$, czyli dla pozostałych graczy (gdzie $\mathbb{1}_X$ to funkcja zwracająca 1 na zbiorze X, a 0 wpp.). Wówczas w pierwotnej grze wygrywał gracz i wtedy i tylko wtedy, gdy w drugiej grze, grze parzystości wygrywa gracz Odd.

4. Czyste równowagi Nasha. Rozważmy grę dwóch graczy daną w formie macierzowej, gdzie każdy z graczy posiada n strategii czystych. Załóżmy, że wypłaty dla każdego gracza w każdej z n^2 pól macierzy są wybrane niezależnie z rozkładem jednostajnym na $[0, 1]$. Oszacować od dołu prawdopodobieństwo, że ta losowa gra będzie miała równowagę Nasha w strategiach czystych.

Kontynuacją tego zadania może być zadanie, które zostało zapisane jako zadanie domowe numer 6.

Wskazówki

- Najpierw można postrzelać ile to mniej więcej będzie dla $n = 1000$. Ciekawie będzie później porównać wynik ze strzałami.
- Zastanówmy się co wystarczy, żeby było spełnione, by istniała czysta równowaga Nasha.
- Oznaczmy strategie gracza A przez $a_1, \ldots, a_n$, a gracza B przez $b_1, \ldots, b_n$. Wystarczy żeby istniały dwie strategie a_i oraz b_j, które są wzajemnie dla siebie najlepsze. Czyli $f_A(a_i, b_j) \geqslant f_A(a_k, b_j)$ dla dowolnego k oraz $f_B(a_i, b_j) \geqslant f_B(a_i, b_k)$ dla dowolnego k, gdzie przez f_A i f_B oznaczamy zyski odpowiednio graczy A i B przy ustalonych strategiach.
- Spójrzmy na sytuację z punktu widzenia gracza A (załóżmy, że wybiera on wiersze). Spójrzmy na pierwszy wiersz od góry, czyli strategię a_1. Przy ustalonej strategii a_1 pewna strategia b_{i_1} jest najlepsza dla gracza B. Jeśli jeszcze strategia a_1 jest najlepsza dla gracza A przy ustalonej strategii b_{i_1} gracza B, to mamy czystą równowagę Nasha. Prawdopodobieństwo tego wynosi $\frac{1}{n}$. Zatem prawdopodobieństwo, że nie znaleźliśmy jeszcze równowagi wynosi $1 - \frac{1}{n}$.
- Spójrzmy teraz na drugi wiersz. Przy ustalonej strategii a_2 pewna strategia b_{i_2} jest najlepsza dla gracza B, pytanie, czy a_2 jest najlepsza dla A przy ustalonej b_{i_2}. Policzmy prawdopodobieństwo, że tak jest. Może być tak, że $b_{i_1} \neq b_{i_2}$ i wówczas wynosi ono $\frac{1}{n}$, w przeciwnym wypadku, gdy $b_{i_1} = b_{i_2}$, to wiemy, że a_1 nie jest najlepsza dla A przy ustalonej strategii gracza B $b_{i_1} = b_{i_2}$, czyli prawdopodobieństwo, że b_2 jest najlepsza wynosi $\frac{1}{n-1}$. My jednak nie chcemy dokładnie liczyć tego prawdopodobieństwa, wystarczy, że jest ono nie mniejsze niż $\frac{1}{n}$, czyli po drugim rzędzie zostajemy bez znalezionej równowagi Nasha w strategiach czystych z prawdopodobieństwem nie większym niż $(1 - \frac{1}{n})^2$.
- Podobnie jest dla każdego następnego wiersza, jeśli strategie gracza B powtarzają się, to wiemy, że te rozważane już wcześniej strategie gracza A nie były najlepsze, więc zwiększa się prawdopodobieństwo, że aktualna jest najlepsza. Zatem po rozważeniu wszystkich n wierszy prawdopodobieństwo, że nie mamy równowagi Nasha wynosi nie więcej niż $(1 - \frac{1}{n})^n$. Liczba ta dąży przy n dążącym do nieskończoności do $\frac{1}{e}$. Zatem dla dużych n dolne oszacowanie na znalezienie równowagi w strategiach czystych dąży do $1 - \frac{1}{e}$ (choć być może nie jest to najlepsza stała).

9 Ćwiczenia 9

20 kwietnia 2009

1. Moc zbioru funkcji. Ktoś pokazuje rozwiązanie zadania domowego **Moc zbioru funkcji**. Wynikiem jest $2^{\mathfrak{C}}$. Najpierw dzielimy zbiór $\{0,1\}^\infty$ na klasy abstrakcji $\sim$, gdzie $\sim$ utożsamia elementy różniące się na skończenie wielu bitach. Później patrzymy na ilu klasach abstrakcji możemy niezależnie zadać wartość (0 lub 1). Wychodzi, że na $\mathfrak{C}$ klasach, czyli ilość dobrych funkcji xor jest równa $2^{\mathfrak{C}}$.

Postaram się tutaj wrzucić link do dobrego rozwiązania.

2. Gry parzystości o skończonej liczbie ranków. Ktoś pokazuje rozwiązania zadania domowego **Gry parzystości o skończonej liczbie ranków**. Znalezienie szukanego algorytmu jest dalece niebanalne. Rozwiązania bazują zwykle na wykorzystywaniu atraktorów, eliminowaniu kolejno największych ranków i rozważaniu mniejszych gier.

Postaram się również tutaj wrzucić link do dobrego rozwiązania.

3. Gra o sumie zerowej. Przy okazji zadania z ćwiczeń nr 8 wyszedł następujący problem.

Rozstrzygnąć, czy dla gier dwóch gracz o sumie zerowej istnieje dokładnie jedna równowaga Nasha.

Dla każdej pary strategii naturalnie definiuje się wartość gry przy tej parze strategii (oczekiwany zysk dla powiedzmy gracza pierwszego). Gdy pierwsze zostanie rozstrzygnięte dodajemy pytanie: czy może się zdarzyć, że istnieją dwie równowagi Nasha, które dają różne wartości gry.

Pewnym nawiązaniem do tego zadania jest zadanie domowe nr 7, który analizuje trochę podobną sytuację dla ogólnych gier dwóch graczy (nie tylko o sumie zerowej).

Wskazówki

- Nie, to nieprawda, chociażby w grze, w której gracz pierwszy zawsze wygrywa stałą wartość $C \in \mathbb{R}$ dowolny wybór strategii jest równowagą Nasha.
- Dodajemy pytanie drugie - czy mogą być dwie równowagi, które dają różne wartości gry.
- Spróbujmy najpierw rozwiązać to pytanie w wersji uproszczonej, czy może się zdarzyć, że istnieją dwie równowagi Nasha w strategiach czystych o tej własności.
- Nie, nie może się zdarzyć. Przypuśćmy, że są to równowagi (a_1, b_1) oraz (a_2, b_2), gdzie a_1, a_2 to strategie gracza A, analogicznie b_1, b_2 strategie gracza B. Przez $f(a, b)$ oznaczmy wartość gry dla strategii a oraz b. Niech A chce zmaksymalizować wypłatę, B zminimalizować (tak jakby B płacił A tę kwotę). Aby punkty (a_1, b_1) i (a_2, b_2) były równowagami Nasha muszą być spełnione odpowiednio nierówności: $f(a_1, b_2) \geqslant f(a_1, b_1) \geqslant f(a_2, b_1)$ oraz $f(a_2, b_1) \geqslant f(a_2, b_2) \geqslant f(a_1, b_2)$. Zauważmy, że po połączeniu te cztery nierówności układają się w kółko, więc muszą być równościami, z czego otrzymujemy $f(a_1, b_1) = f(a_2, b_2)$.
- Teraz pytanie jak to uogólnić na strategie mieszane.
- Dokładnie to samo piszemy dla strategii mieszanych, zamieńmy tylko powiedzmy strategie gracza A na σ_1, σ_2, a strategie gracza B na τ_1, τ_2, by wyglądało bardziej na strategie mieszane.
- Pokazanie, że wszystkie równowagi Nasha w grze o sumie zerowej dają tę samą wartość gry usprawiedliwia trochę to, co zrobiliśmy w zadaniu z Barmanem - wzięliśmy pewną równowagę Nasha i na niej prowadziliśmy obliczenia.