Algorytmiczne Aspekty Teorii Gier

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).