Algorytmiczne Aspekty Teorii Gier

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.