Testowanie własności w czasie stałym - zadania kwalifikacyjne

Marcin Kotowski

11 czerwca 2015

Jeśli doślesz zadania wcześniej, jest możliwość ich poprawiania, można też się dopytywać, przesyłać częściowe pomysły itd. Rozwiązania, pytania i znalezione błędy należy nadsyłać na adres: marcin.kotowski1@gmail.com (najlepiej z czymś typu [WWW11] w temacie). Akceptuję jedynie rozwiązania napisane w LaTeXu albo, w ostateczności, odręcznie [1].

Nie ma ściśle określonej punktacji za zadania, próg kwalifikacji będzie zależał od ilości zgłoszeń i ogólnej jakości nadesłanych rozwiązań.

1 Słowo wstępne

1.1 Wymagania

Zapoznaj się samodzielnie z notacją asymptotyczną (O(1), O(n) itd.) oraz podstawowymi pojęciami z teorii grafów - stopień wierzchołka, spójność, dwudzielność, spójne składowe, planarność. Wystarczy znajomość poję na poziomie Wikipedii.

1.2 Reprezentacje grafów

Niech $G = (V, E)$ będzie grafem o n wierzchołkach. Rozważać będziemy dwie możliwe reprezentacje grafu:

1. **Macierz sąsiedztwa**. Graf reprezentujemy jako tablicę $n \times n$ liczb A_{ij}, gdzie $A_{ij} = 1$, jeśli istnieje krawędź między wierzchołkami i oraz j, i $A_{ij} = 0$ w przeciwnym przypadku. Zapytanie o krawędź polega na sprawdzeniu wartości A_{ij} dla ustalonej pary wierzchołków (i, j).

2. **Lista sąsiedztwa**. Każdy wierzchołek przechowuje listę swoich sąsiadów. Zapytanie o krawędzie następuje przez podanie pary (i, k), gdzie i jest numerem wierzchołka grafu, a k liczbą naturalną - w odpowiedzi dostajemy numer k-tego sąsiada wierzchołka o numerze i.

[1] Jeśli nie znasz jeszcze LaTeXa i rozważasz pisanie odręczne, zastanów się dwa razy. LaTeXa i tak musisz się kiedyś nauczyć - a jeśli nie teraz, to kiedy?

Wybór reprezentacji będzie zależał od ilości krawędzi w grafie. Jeśli grafem jest **gęsty**, czyli zawiera rzędu n^2 krawędzi, będziemy reprezentować go przez macierz sąsiedztwa. Jeśli zaś graf jest **rzadki**, czyli zawiera jedynie rzędu n krawędzi, będziemy używać list sąsiedztwa. Dodatkowo często będziemy zakładać, że istnieje pewna globalna stała d (np. $d = 100$) taka, że rozważane grafy rzadkie mają maksymalny stopień d - wtedy oczywiście taki graf może mieć co najwyżej $\frac{d}{2} \cdot n$ krawędzi, a więc jest rzadki.

Czas działania algorytmu będziemy zazwyczaj mierzyć liczbą wykonanych zapytań. To, co rozumiemy przez zapytanie, zależy od tego, której z dwóch reprezentacji używamy.

1.3 Rachunek prawdopodobieństwa

W zadaniach i na zajęciach dopuszczamy czasem użycie losowości, np. wybranie losowo wierzchołka grafu albo pozycji w ciągu znaków. Zazwyczaj będziemy wymagali, żeby algorytm dawał poprawną odpowiedź z prawdopodobieństwem $\geqslant \frac{2}{3}$. Dokładna wartość $\frac{2}{3}$ nie jest tu istotna - jeśli w zadaniach poniżej oczekiwane jest prawdopodobieństwo sukcesu $\geqslant \frac{2}{3}$, to wystarczające będzie, jeśli otrzymasz dowolne prawdopodobieństwo $p > \frac{1}{2}$ (np. $p = 0.55$ - byle było większe od $\frac{1}{2}$ i niezależne od rozmiaru wejścia n). Uzasadnienie, dlaczego nie jest to istotna różnica, chętni znajdą w sekcji 3.

2 Zadania

Zadanie 1. Skonstruuj graf G o n wierzchołkach, który nie zawiera żadnych trójkątów (to znaczy trzech wierzchołków a, b, c połączonych wzajemnie krawędzimai) i który ma jak najwięcej krawędzi. Im więcej krawędzi, tym wyżej punktowane rozwiązanie, ale uwaga - można udowodnić, że n-wierzchołkowy graf bez trójkątów może mieć co najwyżej $\frac{n^2}{4}$ krawędzi.

Zadanie 2. Niech $x, y \in \{0,1\}^n$ będą dwoma ciągami długości n złożonymi z zer i jedynek. Mamy obiecane, że albo $x = y$, albo też x i y różnią się na co najmniej $\frac{n}{4}$ pozycjach. Zapytanie o pozycję i zwraca wartość x_i oraz y_i. Podaj algorytm używający $O(1)$ zapytań i dający poprawną odpowiedź z prawdopodobieństwem $\geqslant \frac{2}{3}$.

Zadanie 3. Przypuśćmy, że mamy dany graf G o n wierzchołkach i ustalmy $\varepsilon > 0$. Rozpatrujemy dwa przypadki:

1. G jest zadany przez macierz sąsiedztwa. Mamy obiecane, że G albo jest pusty (nie zawiera krawędzi), albo zawiera co najmniej $\varepsilon \cdot n^2$ krawędzi.

2. G jest zadany przez listę sąsiedztwa i ma maksymalny stopień d. Mamy obiecane, że G albo jest pusty (nie zawiera krawędzi), albo zawiera co najmniej $\varepsilon \cdot d \cdot n$ krawędzi.

Dla każdego z przypadków podaj algorytm rozstrzygający, czy graf jest pusty, czy nie, poprawny z prawdopodobieństwem $\geqslant \frac{2}{3}$ i wykorzystujący $O(1)$ zapytań (tj. liczba zapytań może zależeć od ε i d, ale nie od n).

Zadanie 4. Przypuśćmy, że mamy graf G o n wierzchołkach i maksymalnym stopniu d. Ustalmy $\varepsilon > 0$.

1. Przypuśćmy, że G jest ε-daleki od bycia spójnym - aby uzyskać z niego jakikolwiek spójny graf G', należy dodać lub usunąć co najmniej εdn krawędzi. Uzasadnij, że G ma co najmniej εdn spójnych składowych.

2. (dodatkowe, trudniejsze) Przypuśćmy, że G jest ε-daleki od bycia spójnym w mocniejszym sensie - aby uzyskać z niego jakikolwiek spójny graf G' *o maksymalnym stopniu* d, należy dodać lub usunąć co najmniej εdn krawędzi. Zauważ, że ta definicja jest bardziej restrykcyjna od tej z punktu 1 - być może G można uspójnić dodając $< \varepsilon dn$ krawędzi, ale tak powstały graf G' może mieć wierzchołki o stopniu większym od d. Uzasadnij, że G ma co najmniej $\frac{1}{4}\varepsilon dn$ spójnych składowych.

Zadanie 5. Przypuśćmy, że wykonujemy zapytania, których wynikiem może być 0 lub 1 (np. zapytania o elementy macierzy sąsiedztwa grafu lub o pozycje w ciągu zerojedynkowym). Przypuśćmy, że mamy algorytm wykonujący q zapytań, przy czym każde kolejne zapytanie może zależeć od wyniku poprzednich zapytań - na przykład, w pierwszym zapytaniu pytamy o istnienie krawędzi $(1, 2)$, jeśli istnieje, to w drugim pytamy o krawędź $(3, 4)$, a jeśli nie istnieje, to pytamy o krawędź $(5, 6)$. Pokaż, jak skonstruować algorytm dający taki sam wynik, używający co najwyżej 2^q zapytań i którego kolejne zapytania nie zależą od poprzednich - to znaczy, zapytania nowego algorytmu są z góry ustalone (można myśleć, że wykonujemy je jednocześnie). Można założyć, że pierwotny algorytm jest deterministyczny (nie korzysta z losowości).

3 Rachunek prawdopodobieństwa

Przypuśćmy, że mamy algorytm A zwracający wartość 0 lub 1, działający w czasie k i dający poprawną odpowiedź z prawdopodobieństwem $p > \frac{1}{2}$. Rozpatrzmy algorytm A' polegający na uruchomieniu A pewną liczbę N razy, zliczeniu, którą odpowiedź, 0 lub 1, występuje częściej w ciągu wyników, i zwróceniu tej odpowiedzi (dla uproszczenia pomijamy przypadek remisu).

Jakie jest prawdopodobieństwo, że odpowiedź zwrócona przez algorytm A' będzie poprawna? Poprawna odpowiedź musi wystąpić w ciągu wyników co najmniej $> \frac{N}{2}$ razy, więc prawdopodobieństwo to wynosi:

$$p' = \sum_{i=\lfloor \frac{N}{2} \rfloor + 1}^{N} \binom{N}{i} p^i (1-p)^{N-i}$$

Korzystając z nietrudnego rachunku prawdopodobieństwa (chętni mogą zerknąć np. tutaj: `http://en.wikipedia.org/wiki/Chernoff_bound`) można pokazać, że aby otrzymać $p' \geqslant \frac{2}{3}$, wystarczy dobrać N będące pewną stałą zależną jedynie od p, natomiast niezależną od rozmiaru wejścia algorytmu (np. jeśli $p = 0.55$, to wystarczy $N = 10000$).

Jeśli pierwotny algorytm działał w czasie k, nowy algorytm działa w czasie $N \cdot k$, a więc czas działania pomnożył się jedynie przez stałą. Istotne jest, że pierwotne prawdopodobieństwo p nie zależało od rozmiaru wejścia n. Możesz przekonać się, że jeśli np. byłoby $p = \frac{1}{2} + \frac{1}{2^n}$ dla wejścia rozmiaru n, to powyższa metoda nie będzie działać.