Zadanie 4 lista 2

26 czerwca 2020

Spis treści

1 Wstępne spostrzeżenia

Chcemy tak pokolorować wierzchołki, by na każdej ścieżce było co najwyżej k pokolorowanych wierzchołków. Zauważmy, że dla dowolnej ścieżki S_1 istnieje ścieżka S_2 o końcach w pewnych liściach tego drzewa, zatem możemy ograniczyć się do rozważania ścieżek o końcach w liściach.
Zastanówmy się jakie wierzchołki chcemy wybierać do pokolorowania. Rozsądne wydaje się kolorowanie wierzchołków, które występują w najmniejszej liczbie ścieżek. Takimi wierzchołkami są oczywiście liście, bo jeśli ścieżka ma zawierać dany liść, to któryś z jej końców musi być w tym liściu.

2 Idea rozwiązania

Dla danego drzewa G_1 i dopuszczalnej liczby k_1 pokolorowanych wierzchołków na ścieżce, o ile $k_1 > 1$ to kolorujemy wszystkie liście tego drzewa. Po takim kolorowaniu wszystkie ścieżki w drzewie G mają co najwyżej 2 pokolorowane wierzchołki - pokolorowane są tylko liście, więc ścieżka musi się kończyć w liściu by zawierać dany pokolorowany wierzchołek, a skoro ma 2 końce, to stąd maksymalnie 2 pokolorowane wierzchołki. Następnie tworzymy nowe drzewo G_2 usuwając z drzewa G_1 wszystkie liście (oczywiście jest to drzewo, bo G_1 było drzewem, a usunięcie liści nie rozspójnia drzewa). Oznaczamy $k_2 \leftarrow k_1 - 2$. Następnie postępujemy analogicznie jak z G_1 i k_1. Zauważmy, że w G_1 każda ścieżka zawiera maksymalnie 2 pokolorowane wierzchołki, więc po ponownym rozszerzeniu G_2 do G_1 otrzymamy ścieżki o maksymalnie 4 pokolorowanych wierzchołkach. Analogicznie tworzymy G_i oraz k_i do momentu, aż $k_i \leqslant 1$. W przypadku $k_i = 1$ możemy pokolorować jeszcze tylko jeden wierzchołek, bo pokolorowanie dwóch lub więcej utworzy nam ścieżkę o $k + 1$ pokolorowanych wierzchołkach. Jeśli $k_i = 0$ to kończymy kolorowanie. Widzimy, że nasze kolorowanie daje pewne poprawne rozwiązanie. Później udowodnimy, że jest ono optymalne.

3 Algorytm

while $k > 1$ oraz G jest niepuste **do**
  Pokoloruj wszystkie liście drzewa G
  Usuń wszystkie liście z drzewa G
  $k \leftarrow k - 2$
end while
if $k = 1$ i G jest niepuste **then**
  Pokoloruj dowolny wierzchołek
end if

4 Dowód poprawności algorytmu

Pokażemy, że dowolne optymalne kolorowanie można sprowadzić do kolorowania zwracanego przez nasz algorytm.

Lemat 1. *Dla danego drzewa G oraz $k = 1$ można pokolorować co najwyżej 1 wierzchołek. Dla wygody zapisu kolorowanie spełniające warunki zadania dla danego k będę nazywał k kolorowaniem.*

Dowód. Załóżmy nie wprost, że istnieje drzewo, dla którego można pokolorować dwa wierzchołki. Weźmy takie drzewo i kolorowanie. Skoro graf jest drzewem, to jest spójny, równoważnie drogowo-spójny, czyli istnieje droga między pokolorowanymi wierzchołkami, czyli istnieje ścieżka o 2 pokolorowanych wierzchołkach. Sprzeczność. □

Lemat 2. *Załóżmy, że C_1 dla danego k_1 jest optymalnym kolorowaniem drzewa G_1 oraz C_1 ma wszystkie liście pokolorowane. Wtedy dla drzewa G_2 powstałego w wyniku usunięcia wszystkich liści z G_1 oraz $k_2 = k_1 - 2$ kolorowanie C_2 powstałe w wyniku zawężenia kolorowania C_1 do zbioru wierzchołków G_2 jest optymalnym kolorowaniem. Ponadto dowolne optymalne kolorowanie drzewa G_2 można rozszerzyć do optymalnego kolorowania grafu G_1.*

Dowód. Załóżmy nie wprost, że C_2 nie jest optymalnym kolorowaniem grafu G_2. Niech $\overline{C_2}$ będzie jego dowolnym optymalnym kolorowaniem. Skoro C_2 nie jest optymalnym, to liczba pokolorowanych wierzchołków w C_2 jest mniejsza niż w $\overline{C_2}$. Rozważmy kolorowanie $\overline{C_1}$ powstałe w wyniku rozszerzenia G_2 do G_1 i pokolorowania jego liści. Oczywiście skoro $\overline{C_2}$ było optymalnym $k_2 = k_1 - 2$ kolorowaniem grafu G_2, to $\overline{C_1}$ jest optymalnym k_1 kolorowaniem grafu G_1 (dodanie pokolorowanych liści może zwiększyć liczbę pokolorowanych wierzchołków w dowolnej ścieżce o co najwyżej 2). Zatem $\overline{C_1}$ oraz C_1 są optymalnymi kolorowaniami G_1. Ale skoro $\overline{C_2}$ ma więcej pokolorowanych wierzchołków niż C_2 to $\overline{C_1}$ ma więcej niż C_1. Sprzeczność z optymalnością C_1.
Oczywiście każde optymalne kolorowanie grafu G_2 można sprowadzić do optymalnego kolorowania grafu G_1 dodając pokolorowane liście G_2. □

Lemat 3. *Załóżmy, że C jest optymalnym $k > 1$ kolorowaniem drzewa G. Wtedy można sprowadzić C do optymalnego kolorowania, które koloruje wszystkie liście G.*

Dowód. Pierwszym spostrzeżeniem jest, że liczba pokolorowanych wierzchołków jest większa równa liczbie liści, bo skoro $k > 1$ to pokolorowanie wszystkich liści jest poprawnym kolorowaniem, więc optymalne nie może zawierać mniej wierzchołków. Jeśli C zawiera wszystkie liście to teza jest spełniona. Załóżmy więc, że istnieje liść, który nie jest pokolorowany w C. Weźmy dowolny taki liść. Oznaczmy go jako v. Weźmy wierzchołek w leżący najbliżej v, który nie jest liściem i jest pokolorowany (istnieje, bo liczba pokolorowanych $\geqslant$ liczba liści, a jeden liść jest niepokolorowany). Pokolorujmy v i odkolorujmy w. Pokażemy, że nowe kolorowanie jest poprawne. Załóżmy nie wprost, że nie jest poprawne. Wtedy, skoro pokolorowaliśmy tylko jeden nowy wierzchołek, to ścieżka, która zawiera więcej niż k pokolorowanych wierzchołków musi zawierać v. Nazwijmy tę ścieżkę S. Wiemy również, że kolorowanie przed zmianą było poprawne, zatem S nie może zawierać w

(gdyby zawierało, to S przed zmianą kolorowania też zawierałaby więcej niż k pokolorowanych wierzchołków, co jest sprzeczne z poprawnością C). Ale skoro w jest najbliższym v pokolorowanym w C wierzchołkiem, to istnieje ścieżka z w zawierająca wszystkie pokolorowane z S wierzchołki i ma ona więcej niż k pokolorowanych wierzchołków. Sprzeczność. Zatem nowe kolorowanie jest poprawne, a skoro zawiera tyle samo wierzchołków co optymalne C, to również jest optymalne. Postępując analogicznie dla wszystkich niepomalowanych liści otrzymujemy optymalne malowanie, które maluje wszystkie liście. □

Twierdzenie 1. *Każde optymalne kolorowanie można przekształcić do postaci zwracanej przez nasz algorytm.*

Dowód. Indukcja względem k:

1. $k \in \{0, 1\}$
   (a) $k = 0$ Trywialne
   (b) $k = 1$ Z lematu 1.
2. Załóżmy, że dla k można. Pokażemy dla $k + 2$.

   Weźmy dowolne optymalne $k + 2$ kolorowanie C drzewa G. Z lematu 3. możemy przekształcić je do optymalnego kolorowania, które ma wszystkie liście pokolorowane. Z kolei z lematu 2. wiemy, że możemy sprowadzić ten problem do k kolorowania grafu $\overline{G}$ powstałego z usunięcia z G wszystkich liści. Wiemy z lematu 2. że kolorowanie C obcięte do wierzchołków z $\overline{G}$ jest optymalne dla $\overline{G}$. Z założenia indukcyjnego można je sprowadzić do postaci zwracanej przez nasz algorytm. Znowu z lematu 2. możemy je rozszerzyć do optymalnego kolorowania G dodając z powrotem pokolorowane liście. Zatem dla $k+2$ nasze założenie również jest spełnione.

Z 1. i 2. na mocy zasady indukcji matematycznej każde optymalne kolorowanie można przekształcić do postaci zwracanej przez nasz algorytm. □

5 Złożoność pamięciowa i obliczeniowa

5.1 Złożoność pamięciowa

Drzewo G możemy trzymać w odwrotnej liście sąsiedztwa (lista ojcostwa). Odwrotna lista sąsiedztwa drzewa zajmuje $\mathcal{O}(n)$ pamięci (w ogólności $\mathcal{O}(n + m)$, ale dla drzewa $m = n - 1$). Do sprawdzenie czy wierzchołek jest liściem możemy utworzyć tablicę pomocniczą t, która dla każdego wierzchołka będzie trzymała liczbę wychodzących z niego krawędzi - $\mathcal{O}(n)$ pamięci. Dodatkowo do trzymania zbioru liści możemy użyć kolejki, której rozmiar może wynosić co najwyżej tyle ile jest wierzchołków w drzewie, czyli $\mathcal{O}(n)$. Zatem łącznie potrzebujemy $\mathcal{O}(n)$ pamięci.

5.2 Złożoność obliczeniowa

Liście możemy trzymać w kolejce. W każdym obrocie pętli pobieramy liście z kolejki i kolorujemy je w czasie stałym. Następnie zmniejszamy liczbę wychodzących krawędzi dla ojca każdego liścia o 1 i jeśli wynosi 0 to dodajemy go do nowej kolejki (dzięki odwrotnej liście sąsiedztwa i tablicy t wykonujemy to w czasie stałym). Zmniejszamy k o 2 w czasie stałym. Wszystkie operacje wykonują się w czasie stałym. Należy zastanowić się zatem ile łącznie we wszystkich obrotach pętli będziemy pobierać wierzchołków z kolejek. To w dużej mierze zależy od struktury drzewa oraz wartości k, ale w pesymistycznym przypadku będziemy musieli pobrać wszystkie wierzchołki drzewa, czyli łącznie we wszystkich obrotach pętli wykonamy $\mathcal{O}(n)$ operacji. Wykonanie ostatniego warunku if wymaga stałego czasu, więc nie wpływa na złożoność asymptotyczną algorytmu.