Algorytmy i Struktury Danych

7 czerwca 2021

Ćwiczenia 11: Maksymalny przepływ oraz drzewa BST

Zadania obowiązkowe

Zadanie 1. (wiele źródeł i ujść) Mamy dany graf skierowany $G = (V, E)$ oraz funkcję $c: E \to \mathbb{N}$ opisującą przepustowość każdej krawędzi (liczbę jednostek towaru na godzinę, które mogą się przemieszczać krawędzią). Poza tym mamy dany zbiór wierzchołków-fabryk $S = \{s_1, \ldots, s_n\}$ oraz zbiór wierzchołków-sklepów $T = \{t_1, \ldots, t_m\}$. Dla każdej fabryki s_i znamy liczbę p_i określającą ile jednostek towaru na godzinę fabryka może maksymalnie produkować. Jednocześnie dla każdego sklepu t_j mamy liczbę q_j, która mówi ile jednostek towaru na godzinę musi do tego sklepu docierać. Proszę podać algorytm, który sprawdza, czy da się zapewnić, żeby do każdego sklepu docierało z fabryk dokładnie tyle jednostek towaru ile sklep wymaga jednocześnie nie zmuszając żadnej fabryki do przekroczenia swoich możliwości produkcyjnych i nie przekraczając przepustowości żadnej z krawędzi.

Komentarz. Wystarczy zbudować superźródło z odpowiednimi przepustowościami do fabryk i superujście od sklepów, a potem użyć standardowego algorymu.

Zadanie 2. (następnik) Proszę zaimplementować funkcję znajdującą element o następnej wartości klucza niż podany w drzewie BST

Maksymalny przepływ

Zadanie 1. (maksymalny przepływ w grafie nieskierowanym) Proszę wskazać algorytm, który znajduje maksymalny przepływ między źródłem i ujściem w grafie nieskierowanym. Proszę użyć algorytmu z wykładu—dla grafów skierowanych, gdzie między każdą parą wierzchołków jest najwyżej jedna krawędź—jako czarnej skrzynki. Alternatywnie można opisać implementację bezpośrednio pracującą na grafie nieskierowanym.

Zadanie 2. (spójność krawędziowa) Dany jest graf nieskierowany $G = (V, E)$. Mówimy, że spójność krawędziowa G wynosi k jeśli usunięcie pewnych k krawędzi powoduje, że G jest niespójny, ale usunięcie dowolnych $k - 1$ krawędzi nie rozspójnia go. Proszę podać algorytm, który oblicza spójność krawędziową danego grafu.

Zadanie 3. (Formuły logiczne z dwoma wystąpieniami zmiennej) Dana jest formuła logiczna postaci: $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, gdzie każda C_i to klauzula będąca alternatywą zmiennych i/lub ich zaprzeczeń. Wiadomo, że każda zmienna występuje w formule dokładnie dwa razy, raz zanegowana i raz niezanegowana. Na przykład poniższa formuła stanowi dopuszczalne wejście:

$$(x \vee y \vee z) \wedge (\overline{y} \vee w) \wedge (\overline{z} \vee v) \wedge (\overline{x} \vee \overline{w}) \wedge (\overline{v}).$$

Proszę podać algorytm, który oblicza takie wartości zmiennych, że formuła jest prawdziwa.

Zadanie 4. (skojarzenie na drzewie) Proszę podać algorytm, który mając na wejściu drzewo oblicza skojarzenie o maksymalnej liczności. Czy algorytm dalej będzie działać jeśli każda krawędź będzie mieć dodatnią wagę i szukamy skojarzenia o maksymalnej sumie wag?

Zadanie 5. (rozłączne ścieżki) Dany jest graf skierowany $G = (V, E)$ oraz wierzchołki s i t. Proszę zaproponować algorytm znajdujący maksymalną liczbę rozłącznych (wierzchołkowo) ścieżek między s i t.

Drzewa BST

Zadanie 1. (Indeksowane drzewa BST) Rozważmy drzewa BST, które dodatkowo w każdym węźle zawierają pole z liczbą węzłów w danym poddrzewie. Proszę opisać jak w takim drzewie wykonywać następujące operacje:

1. znalezienie i-go co do wielkości elementu,
2. wyznaczenie, którym co do wielkości w drzewie jest zadany węzeł

Proszę zaimplementować obie operacje.

Zadanie 2. Proszę zapropnować algorytm, który oblicza sumę wszystkich wartości w drzewie binarnym zdefiniowanym na węzłach typu:

```
class BNode:
  def __init__( self, value ):
    self.left   = None
    self.right  = None
    self.parent = None
    self.value  = val
```

Program może korzystać wyłącznie ze stałej liczby zmiennych (ale wolno mu zmieniać strukturę drzewa, pod warunkiem, że po zakończonych obliczeniach drzewo zostanie przywrócone do stanu początkowego.)

Zadanie 3. (geny) W pewnym laboratorium genetycznym powstał ciąg sekwencji DNA. Każda sekwencja to pewien napis składający się z symboli G, A, T, i C. Przed dalszymi badaniami konieczne jest upewnić się, że wszystkie sekwencje DNA są parami różne. Proszę opisać algorytm, który sprawdza czy tak faktycznie jest.

Zadanie 4. (klocki) Dany jest ciąg klocków $(K_1, \ldots, K_n)$. Klocek K_i zaczyna sie na pozycji a_i i ciągnie się do pozycji b_i (wszystkie pozycje to nieujemne liczby naturalne) oraz ma wysokość 1. Klocki układane są po kolei–jeśli klocek nachodzi na któryś z poprzednich, to jest przymocowywany na szczycie poprzedzającego klocka). Na przykład dla klocków o pozycjach (1, 3), (2, 5), (0, 3), (8, 9), (4, 6) powstaje konstrukcja o wysokości trzech klocków. Proszę podać możliwie jak najszybszy algorytm, który oblicza wysokośc powstałej konstrukcji.