Lista zadań. Nr 8. 10 czerwca 2013

Algorytmy i Struktury Danych

IIUWr. II rok informatyki.

1. (2pkt) Niech σ będzie ciągiem instrukcji *Union* i *Find*, w którym wszystkie instrukcje *Union* występują przed instrukcjami *Find*. Udowodnij, że algorytm oparty na strukturach drzewiastych wykonuje σ w czasie proporcjonalnym do długości σ.

2. (2pkt) Instrukcja $Delete(i)$ usuwa element i ze zbioru, do którego on aktualnie należy i tworzy jednoelementowy zbiór $\{i\}$. Podaj implementację instrukcji $Delete(i)$, która pozwoli na wykonywanie ciągów złożonych z n instrukcji *Union*, *Find* i *Delete* w czasie $O(n \log^* n)$.

3. (2pkt) Rozważamy ciągi operacji $Insert(i)$, $DeleteMin$ oraz $Min(i)$ wykonywanych na S - podzbiorze zbioru $\{1, \ldots, n\}$. Obliczenia rozpoczynamy z $S = \emptyset$. Instrukcja $Insert(i)$ wstawia liczbę i do S. Instrukcja $DeleteMin$ wyznacza najmniejszy element w S i usuwa go z S. Natomiast wykonanie $Min(i)$ polega na usunięciu z S wszystkich liczb mniejszych od i.

   Niech σ będzie ciągiem instrukcji $Insert(i)$, $DeleteMin$ oraz $Min(i)$ takim, że dla każdego i, $1 \leq i \leq n$, instrukcja $Insert(i)$ występuje co najwyżej jeden raz. Mając dany ciąg σ naszym zadaniem jest znaleźć ciąg liczb usuwanych kolejno przez instrukcje $DeleteMin$. Podaj algorytm rozwiązujący to zadanie.
   UWAGA: Zakładamy, że cały ciąg σ jest znany na początku, czyli interesuje nas wykonanie go *off-line*.
   WSKAZÓWKA: Rozdział 4.8 z książki Aho,... .

4. (2pkt) Rozważmy ciało F_2 liczb całkowitych modulo 2. Ułóż algorytm, który mnoży macierze $n \times n$ nad F_2 w czasie $O(n^{2.81}/(\log n)^{0.4})$.

5. (1pkt) Opisz, jak skonstruować automat, który dla dwóch danych wzorców P i P' znajduje wszystkie ich wystąpienia w tekście. Spróbuj zminimalizować liczbę stanów w Twoim automacie.

6. (2pkt) Słowa Fibonacciego definiujemy w następujący sposób:

$$F_0 = a \quad F_1 = b \quad \text{oraz } F_{k+1} = F_k F_{k-1} \text{ dla } k \geq 1.$$

   Skonstruuj algorytm, który sprawdza w czasie liniowym, czy dany tekst jest słowem Fibonacciego. Postaraj się, by Twój algorytm używał możliwie najmniejszej pamięci pomocniczej.

7. (1pkt)Załóżmy, że wzorzec P może zawierać znak $\diamondsuit$ (tzw. gap character). Znak ten jest zgodny z dowolnym podsłowem (także z podsłowem pustym). Na przykład, wzorzec $ab\diamondsuit ba\diamondsuit c$ występuje w słowie *cabccbacbacab* jako

$$c\underbrace{ab}_{ab}\underbrace{cc}_{\diamondsuit}\underbrace{ba}_{ba}\underbrace{cba}_{\diamondsuit}\underbrace{c}_{c}ab$$

   a także jako

$$c\underbrace{ab}_{ab}\underbrace{ccbac}_{\diamondsuit}\underbrace{ba}_{ba}\underbrace{}_{\diamondsuit}\underbrace{c}_{c}ab.$$

   Podaj algorytm znajdujący wystąpienie takiego wzorca w danym tekście T (oczywiście zakładamy, że $\diamondsuit$ nie występuje w T).

1. (2pkt) Rozważamy ciągi instrukcji: $Link(r, v)$ oraz $Depth(v)$ wykonywanych na lesie rozłącznych drzew o wierzchołkach z etykietami ze zbioru $\{0, ..., n-1\}$ (różne wierzchołki mają różne etykiety). Operacja $Link(r, v)$ czyni r, korzeń jednego z drzew, synem v, wierzchołka innego drzewa. $Depth(v)$ oblicza głębokość wierzchołka v.

   Naszym celem jest napisanie algorytmu, który dla danego ciągu σ wypisze w sposób on-line wyniki instrukcji *Depth* (tzn. wynik każdej instrukcji *Depth* ma być obliczony przed wczytaniem kolejnej instrukcji z ciągu σ). Pokaż jak zastosować drzewiastą strukturę danych dla problemu $Union - Find$ do rozwiązania tego problemu.
   WSKAZÓWKA: Rozdział 4.8 z książki Aho,... .

2. (2pkt) Opracuj strukturę danych, w której pesymisyczny czas wykonania zarówno pojedynczej *Union* jak i pojedynczej *Find* będzie $o(\log n)$. Zauważ, że w metodach podanych na wykładzie pesymistyczny czas wykonania co najmniej jednej z tych operacji wynosi $\Omega(\log n)$.

3. (2pkt) Rozważamy ciągi instrukcji: $Link(r, v)$ oraz $Depth(v)$ wykonywanych na lesie rozłącznych drzew o wierzchołkach z etykietami ze zbioru $\{0, ..., n-1\}$ (różne wierzchołki mają różne etykiety). Operacja $Link(r, v)$ czyni r, korzeń jednego z drzew, synem v, wierzchołka innego drzewa. $Depth(v)$ oblicza głębokość wierzchołka v.

   Naszym celem jest napisanie algorytmu, który dla danego ciągu σ wypisze w sposób on-line wyniki instrukcji *Depth* (tzn. wynik każdej instrukcji *Depth* ma być obliczony przed wczytaniem kolejnej instrukcji z ciągu σ). Pokaż jak zastosować drzewiastą strukturę danych dla problemu $Union - Find$ do rozwiązania tego problemu.
   WSKAZÓWKA: Rozdział 4.8 z książki Aho,... .

4. (2pkt) Udowodnij, że przedstawiony na wykładzie algorytm *Union-Find* oparty na strukturach drzewiastych ma złożoność większą niż cn dla dowolnej stałej c.
   WSKAZÓWKA: Rozdział 4.8 z książki Aho,... .

5. (1pkt) Podaj algorytm, który w czasie liniowym określa, czy tekst T powstał przez przesunięcie cykliczne tekstu T'.

6. (1pkt) Rozważ taką wersję wykonywania kompresji ścieżek, w której wierzchołki wizytowane podczas wykonywania operacji *Find* podwieszane są pod własnego dziadka. Czy analiza złożoności przeprowadzona na wykładzie da się zastosować w tym przypadku? Czy widzisz jakąś zaletę takiej kompresji ścieżek w stosunku do oryginalnej metody?