Lista zadań. Nr 7. 21 maja 2013

Algorytmy i Struktury Danych

IIUWr. II rok informatyki.

1. Niech $A = \{a_1, a_2, \ldots, a_n\}$ będzie zbiorem kluczy, (takim, że $\forall_{i=1,\ldots,n-1} a_i < a_{i+1}$), które chcemy pamiętać w słowniku stałym. Znamy także ciąg $p_1, \ldots, p_n$ prawdopodobieństw zapytania o poszczególne klucze. Przyjmujemy, że $p_1 + p_2 + \ldots + p_n = 1$, a więc do słownika nie będą kierowane zapytania o klucze spoza słownika. Chcemy zaimplementować słownik jako drzewo BST. Ułóż algorytm znajdujący takie drzewo, które minimalizuje oczekiwany czas wykonywania operacji na słowniku.

   Punktacja:

   - za algorytm działający w czasie $O(n^3)$ - 1pkt;
   - za algorytm działający w czasie $O(n^2)$ - 2pkt;

2. (1pkt) Rodzinę H funkcji hashujących ze zbioru kluczy U w zbiór indeksów $\{0, .., m-1\}$ nazywamy *uniwersalną* (lub inaczej 2-*uniwersalną*), jeśli

   $$\forall_{x \neq y \in U} Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}.$$

   Rozważmy następującą rodzinę funkcji haszujących. Niech uniwersum $U = \{0, 1, 2, \ldots, n-1\}$ i niech $V = \{0, 1, 2, \ldots, m-1\}$ będzie zbiorem wartości funkcji, gdzie $m \leq n$. Niech ponadto $p > n$ będzie liczbą pierwszą. Funkcje $h_{a,b}$ definiujemy w następujący sposób:

   $$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m.$$

   Udowodnij, że rodzina

   $$H = \{h_{a,b} \mid 1 \leq a \leq p-1,\ 0 \leq b \leq p-1\}$$

   jest 2-uniwersalna.

3. (2pkt) Ułóż algorytm rozwiązujący problem znajdowania najbliżej położonej pary punktów na płaszczyźnie oparty na następującej idei. Niech d będze odległością pomiędzy parą najbliżej położonych punktów spośród punktów $p_1, p_2, \ldots, p_{i-1}$. Sprawdzamy, czy p_i leży w odległości mniejszej niż d, od któregoś z poprzednich punktów. W tym celu dzielimy płaszczyznę na odpowiednio małe kwadraty, tak by w każdym z nich znajdował się nie więcej niż jeden punkt. Te "zajęte" kwadraty pamiętamy w słowniku.

   Twój algorytm powinien działać w oczekiwanym czasie liniowym. Jeśli nie potrafisz zbudować algorytmu opartego na powyższej idei, możesz opracować algorytm oparty na iinej (ale spełniający te same wymagania czasowe).

4. (2pkt) Na wykładzie został przedstawiony dowód tego, że po umieszczeniu n kluczy w n-elementowej tablicy haszującej (z nawlekaniem kluczy o tej samej wartości funkcji haszującej na listy) z dużym prawdopodobieństwem żadna lista nie będzie zawierać więcej niż $\log n / \log \log n$ kluczy, o ile do haszowania użyta zostanie losowa funkcja haszująca h. Przypomnij ten dowód lub podaj inny (ale nadal poprawny:-).

5. (1pkt) Przeprowadź analizę zamortyzowanego kosztu ciągu operacji *insert, deletemin, decrease-key, meld, findmin* wykonywanych na kopcach Fibonacciego, w których kaskadowe wykonanie operacji *cut* wykonywane jest dopiero wtedy, gdy wierzchołek traci trzeciego syna.

1. (1pkt) Uzasadnij stwierdzenie, że funkcje hashujące potrzebne w konstrukcji słownika stałego (podanej na wykładzie) mogą być efektywnie znalezione.

2. (2pkt) Opracuj algorytm, który dla danych macierzy A, B i C o wymiarach $n \times n$ sprawdza, czy C jest równa iloczynowi $a \cdot B$. Twój algorytm powinień dawać poprawną odpowiedź z dużym prawdopodobieństwem.