ćwiczenia (licencjat wieczorowy): lista zadań nr 14 *23 czerwca 2009 r.*

ALGORYTMY I STRUKTURY DANYCH

METODA „DZIEL I ZWYCIĘŻAJ", PROGRAMOWANIE DYNAMICZNE, STRATEGIA ZACHŁANNA

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [2 pkt.] Dane są trzy liczby naturalne x, n i p. Skonstruuj rekurencyjny algorytm, który obliczy $x^n \bmod p$. Przedstaw implementację twojej metowy w pseudokodzie i oszacuj jej złożoność obliczeniową.

   Wskazówka. Rozważ najpierw prostszy przypadek, w którym n jest potęgą liczby 2.

2. [1.5 pkt.] Dane jest regularne drzewo binarne T (drzewo jest regularne, gdy każdy węzeł, który nie jest liściem ma 2 synów) z korzeniem w węźle r. Opracuj algorytm, który wyznaczy długość najdłuższej ścieżki w tym drzewie (długość mierzymy liczba krawędzi na ścieżce). Uzasadnij poprawność twojej metody i oszacuj czas jej działania.

3. [1.5 pkt.] Dane jest regularne drzewo binarne T (drzewo jest regularne, gdy każdy węzeł, który nie jest liściem ma 2 synów) z korzeniem w węźle r. Opracuj algorytm, który wyznaczy maksymalną wysokość drzewa pełnego, które można umieścić w zadanym drzewie (korzeń maksymalnego drzewa pełnego nie musi się znajdować w węźle r). Uzasadnij poprawność twojej metody i oszacuj czas jej działania.

4. [2 pkt.] Dana jest n–elementowa nieuporządkowana tablica liczb rzeczywistych $T[0 \ldots n{-}1]$ (liczby w tej tablicy są unikatowe, czyli nie ma w niej dwóch identycznych wartości). Element w tablicy nazywamy *lokalnym minimum*, jeśli sąsiadujące z nim elementy są większe od niego. Opracuj algorytm, który znajdzie jakiekolwiek lokalne minimum w tablicy T.

*

5. [1.5 pkt.] Zmodyfikuj podany na wykładzie algorytm dla problemu *optymalnego mnożenia macierzy*, tak aby wypisywał on ciąg n macierzy wraz z poprawnie rozstawionymi nawiasami na podstawie wyliczonej w trakcie działania algorytmu tablicy pomocniczej $L[1 \ldots n][1 \ldots n]$. Samo wypisanie wspomnianego ciągu macierzy wraz z nawiasami powinno posiadać złożoność czasową rzędu $O(n)$.

6. [2 pkt.] Uogólnij algorytm znajdowania *optymalnego drzewa BST* dla uporządkowanego zbioru n wartości $s_1 < s_2 < \ldots < s_n$ na przypadek, gdy znane są prawdopodobieństwa zapytań o każdą wartość $p_1, p_2, \ldots, p_n$ oraz prawdopodobieństwa zapytań o watości spoza tego zbioru $q_0, q_1, \ldots, q_n$ (q_i to prawdopodobieństwo zapytania o wartość znajdującą się pomiędzy s_i a s_{i+1}, dla $i = 1, \ldots, n-1$; natomiast q_0 to prawdopodobieństwo zapytania o wartość mniejszą od s_1, a q_n o wartość większą od s_n). Uzasadnij poprawność zaproponowanego algorytmu i przeanalizuj jego złożoność obliczeniową.

   Uwaga. Oczywiście wszystkie prawdopodobieństwa sumują się do 1, czyli $\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$.

7. [2 pkt.] Zadana jest liczba naturalna $n > 1$. Liczbę tą należy przedstawić w postaci *sumy kwadratów* liczb naturalnych. Suma ta ma się składać z minimalnej liczby składników. Przedstaw efektywny algorytm rozwiązujący ten problem, uzasadnij jego poprawność i oszacuj złożononość obliczeniową. Wykaż, że strategia zachłanna nie zawsze daje optymalny wynik.

   Przykład. Rozważmy liczby 13, 16 i 27. Można je następująco przedstawić w postaci sumy kwadratów o minimalnej liczbie składników:

   - $13 = 2^2 + 3^2$
   - $16 = 4^2$
   - $27 = 1^2 + 1^2 + 5^2 = 3^2 + 3^2 + 3^2$

*

8. [2 pkt.] Dany jest zbiór n punktów na prostej $p_1, p_2, \ldots, p_n$. Każdy punkt ma określoną współrzędną rzeczywistą, odpowiednio $x_1, x_2, \ldots, x_n$. Opracuj efektywny algorytm, który wyznaczy najmniejszy zbiór jednostkowych przedziałów domkniętych zawierający wszystkie n punktów. Uzasadnij poprawność zaprezentowanego algorytmu i oszacuj jego złożoność obliczeniową.

9. [2 pkt.] Szczególnym przypadkiem dyskretnego problemu plecakowego jest *łatwy problem plecakowy* — mamy z nim do czynienia wtedy, gdy uporządkowane rosnąco wartości przedmiotów tworzą ciąg superrosnący (ciąg liczb jest superrosnący, gdy każda kolejna liczba jest większa od sumy wszystkich wcześniejszych). Ułóż algorytm rozwiązujący taką wersję problemu plecakowego i udowodnij, że znajduje on optymalne rozwiązanie. Jaki jest czas działania twojego algorytmu?

10. [2.5 pkt.] Uogólnij *algorytm Huffmana* dla kodów trójkowych (kodów używających symboli 0, 1 i 2) i czwórkowych (kodów używających symboli 0, 1, 2 i 3). Udowodnij, że twoje algorytmy generują kody optymalne.