Lista zadań. Nr 3 27 marca 2025

Algorytmy i Struktury Danych

IIUWr. II rok informatyki.

1. (0,5pkt) Rozwiąż z dokładnością do Θ następującą rekurencję:

$$T(n) = \begin{cases} 1 & \text{dla } n = 1 \\ 2T(n/2) + n/\log n & \text{dla } n > 1 \end{cases}$$

2. (1,5pkt) Mówimy, że tablica $A[1..n]$ ma *element większościowy*, jeśli więcej niż połowa jej elementów ma tę samą wartość. Ułóż algorytm, który sprawdza, czy dana tablica ma element większościowy i jeśli tak jest, wyznacza jego wartość. Załóż, że na elementach można wykonywać jedynie porównania typu "czy $A[i] = A[j]$?" i koszt każdego takiego porównania jest stały, natomiast nie można wykonywać porówań typu "$A[i] < A[j]$?". Rozważ dwie strategie Dziel i Zwyciężaj, zaczynające się od:

   (a) Podziel A na dwie "połowy"...

   (b) Pogrupuj elementy z A w pary. Pórównaj elementy w każdej parze. Jeśli są różne, usuń tę parę z tablicy, jeśli są równe - usuń jeden z nich a drugi pozostaw ...

3. (2pkt) Dana jest tablica liczbowa $A[1..n]$ i zbiór S par indeksów $(p_1, k_1), \ldots (p_r, k_r)$. Ułóż algorytm, odpowiadający na pytania o największy wspólny dzielnik liczb $\{A[p_i], \ldots, A[k_i]\}$, dla $i = 1, \ldots, r$, oparty na następującej strategii Dziel i Zwyciężaj:

   - ciag zapytań dzielimy na trzy podzbiory:
     - $S_M = \{(p, k) \in S : p \leq n/2 \leq k\}$
     - $S_L = \{(p, k) \in S : p \leq k < n/2\}$
     - $S_R = \{(p, k) \in S : n/2 < p \leq k\}$
   - odpowiadamy na pytania z S_M,
   - rekurencyjnie wykonujemy algorytm dla zbiorów S_L i S_R.

4. (2pkt) Danych jest n prostych $l_1, l_2, \ldots l_n$ na płaszczyźnie ($l_i = a_i x + b_i$), takich że żadne trzy proste nie przecinają się w jednym punkcie. Mówimy, że prosta l_i jest widoczna z punktu p jeśli istnieje punkt q na prostej l_i, taki że odcinek $\overline{pq}$ nie ma wspólnych punktów z żadną inną prostą l_j ($j \neq i$) poza (być może) punktami p i q.

   Ułóż algorytm znajdujący wszystkie proste widoczne z punktu $(0, +\infty)$.

5. (2pkt) Zaproponuj modyfikację algorytmu Karatsuby, która oblicza kwadrat danej liczby. Rozważ podział liczby na k części:

   - dla $k = 2$,
   - dla $k = 3$.

   Postaraj się, by stałe używane przez algorytm były jak najmniejsze.
   Czy dla ustalonego k można otrzymać algorytm podnoszenia do kwadratu, który jest asymptotycznie szybszy od algorytmu mnożenia?

6. (1,5pkt) Dane jest drzewo binarne (możesz założyć dla prostoty, że jest to pełne drzewo binarne), którego każdy wierzchołek v_i skrywa pewną liczbę rzeczywistą x_i. Zakładamy, że wartości skrywane w wierzchołkach są różne. Mówimy, że wierzchołek v jest minimum lokalnym, jeśli wartość skrywana w nim jest mniejsza od wartości skrywanych w jego sąsiadach.

   Ułóż algorytm znajdujący lokalne minimum odkrywając jak najmniej skrywanych wartości.

7. Dane jest nieukorzenione drzewo z naturalnymi wagami na krawędziach oraz liczba naturalna C.

   (a) (2pkt) Ułóż algorytm obliczający, ile jest par wierzchołków odległych od siebie o C.

   (b) (**Z** 2,5pkt) Jak w punkcie (a), ale algorytm ma działać w czasie $O(n \log n)$.

   UWAGA: Można zadeklarować tylko jeden z punktów (a), (b).

8. (1pkt) *Inwersją* w ciągu $A = a_1, \ldots, a_n$ nazywamy parę indeksów $1 \leq i < j \leq n$, taką że $a_i > a_j$. Pokaż jak można obliczyć liczbę inwersji w A podczas sortowania przez scalanie.

9. (2pkt) Niech P_n będzie zbiorem przesunięć cyklicznych ciągu n-elementowego o potęgi liczby 2 nie większe od n. Pokaż konstrukcję sieci przełączników realizujących przesunięcia ze zbioru P_n.

   Uwagi:

   - Możesz założyć, że n jest potęgą dwójki albo szczególną potęgą dwójki, albo ...
   - Sieć Beneša-Waksmana jest dobrym rozwiązaniem wartym 0pkt (tzn. nic niewartym).

10. (**Z** 2pkt) *Dekompozycją centroidową* nazywamy następujący proces: dla danego drzewa T na n wierzchołkach znajdź wierzchołek $u \in T$ taki, że każda spójna składowa $T \setminus \{u\}$ ma rozmiar co najwyżej $n/2$, a następnie powtórz rozumowanie w każdej z tych spójnych składowych (o ile zawierają więcej niż jeden wierzchołek). Taka dekompozycja może być w naturalny sposób reprezentowana jako drzewo T' na n wierzchołkach, którego korzeniem jest u. Naiwna implementacja powyższej procedury działa w czasie $O(n \log n)$. Ułóż algorytm, który konstruuje T' w czasie $O(n)$.