Lista zadań. Nr 3. 18 marca 2006

ALGORYTMY I STRUKTURY DANYCH

IIUWr. II rok informatyki.

1. (2pkt) Przeanalizuj następujący algorytm jednoczesnego znajdowania maksimum i minimum w zbiorze $S = \{a_1, \ldots, a_n\}$:

```
Procedure MaxMin(S:set)
  if |S|= 1 then return {a1, a1}
  else
    if |S|= 2 then return (max(a1, a2), min(a1, a2))
    else
      podziel S na dwa równoliczne (z dokładnością do jednego elementu) podzbiory S1, S2
      (max1, min1) ← MaxMin(S1)
      (max2, min2) ← MaxMin(S2)
      return (max(max1, max2), min(min1, min2))
```

   - Dla jakich danych osiąga on dolną granicę na wymaganą dla tego problemu liczbę porównań? Podaj wzorem wszystkie takie wartości.
   - Jak bardzo może różnić się liczba porównań wykonywanych przez algorytm od dolnej granicy?
   - Popraw algorytm, tak by osiągał on tę granicę dla każdej wartości n?

2. (2pkt) *Nieporządkiem* w ciągu $a_1, \ldots, a_n$ nazywamy każdą parę indeksów $\{i, j\}$ taką, że $i < j$ oraz $a_i > a_j$. Ułóż algorytm obliczający liczbę nieporządków w danym ciągu n-elementowym.

3. (1 pkt) Niech u i v będą liczbami o n i m cyfrach (odpowiednio). Załóżmy, że $m \leq n$. Klasyczny algorytm oblicza iloczyn tych liczb w czasie $O(mn)$. Algorytm *multiply* z wykładu potrzebuje $O(n^{\log 3})$ czasu, co jest nie do zaakceptowania gdy m jest znacznie mniejsze od n. Pokaż, że w takim przypadku można pomnożyć liczby u i v w czasie $O(nm^{\log(3/2)})$.

4. (2pkt) Element tablicy nazywamy *lokalnym minimum*, jeśli sąsiadujące z nim elementy są mniejsze od niego. Ułóż algorytm, który znajduje (jakieś) lokalne minimum w tablicy n różnych elementów.

5. (2pkt) Macierz A rozmiaru $n \times n$ nazywamy macierzą Toeplitza, jeśli jej elementy spełniają równanie $A[i, j] = A[i-1, j-1]$ dla $2 \leq i, j \leq n$.
   (a) Podaj reprezentację macierzy Toeplitza, pozwalającą dodawać dwie takie macierze w czasie $O(n)$.
   (b) Podaj algorytm, oparty na metodzie "dziel i zwyciężaj", mnożenia macierzy Toeplitza przez wektor. Ile operacji arytmetycznych wymaga takie mnożenie?

6. (2pkt) Przeanalizuj sieć permutacyjną omawianą na wykładzie (tzw. sieć Beneša-Waksmana)
   - Pokaż, że ostatnią warstwę przełączników sieci Beneša-Waksmana można zastąpić inną warstwą, która zawiera $n/2 - 1$ przełączników (a więc o jeden mniej niż w sieci oryginalnej) a otrzymana sieć nadal będzie umożliwiać otrzymanie wszystkich permutacji.
   - Uogólnij sieć na dowolne n (niekoniecznie będące potęgą liczby 2).

7. (2pkt) Jakie jest prawdopodobieństwo wygenerowania permutacji identycznościowej przez sieć Beneša-Waksmana, w której przełączniki ustawiane są losowo (jak w poprzednim zadaniu).

Zadania dla grupy zaawansowanej

1. (2pkt) Skonstruuj rodzinę sieci przełączników $\{S_n | n \in N\}$ o głębokości asymptotycznie mniejszej od $\log n$, taką że S_n umożliwia realizację wszystkich przesunięć cyklicznych o potęgi liczby 2 nie większe od n lub udowodnij, że taka rodzina nie istnieje.

2. (2pkt) Rozważ algorytm mnożenia długich liczb, w którym argumenty są dzielone na niekoniecznie stałą liczbę części. Jaką złożoność potrafisz w ten sposób osiągnąć? Pamiętaj, że wraz ze wzrostem liczby części wzrastają wielkości współczynników w kombinacjach liniowych.

3. (2pkt) Ułóż algortytmy znajdowania lokalnego minimum (patrz zadanie 4 z regularnej listy) w:
   - pełnym drzewie binarnym o $n = 2^k - 1$ wierzchołkach;
   - kracie o rozmiarze $n \times n$.

Zadania dodatkowe - nie będą rozwiązywane w czasie ćwiczeń

1. (0 pkt) Przypomnij sobie algorytm scalający dwie posortowane tablice U i V w czasie liniowym, tj. w czasie liniowo proporcjonalnym do sumy długości tych tablic.

2. (1 pkt) Złożoność podanego na wykładzie algorytmu sortowania przez scalenia wyraża się wzorem:

$$\begin{aligned} &T(1) = a \\ &T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + bn \qquad \text{dla } n > 1 \end{aligned}$$

   dla pewnych $a, b > 0$. Udowodnij, że $T(n) \in O(n \log n)$.

3. (1 pkt) Zamiast dzielić tablicę T na dwie połówki, algorytm sortowania przez scalanie mógłby dzielić ją na części o rozmiarach $\lceil n/3 \rceil$, $\lceil (n+1)/3 \rceil$ oraz $\lceil (n+2)/3 \rceil$, sortować niezależnie każdą z tych części, a następnie scalać je. Podaj bardziej formalny opis tego algorytmu i przeanalizuj czas jego działania.

4. (1pkt) Rozważ wersje algorytmu *muliply* dzielące czynniki na trzy i cztery części. Oblicz współczynniki w kombinacjach liniowych określających wartości c_i.

5. (1pkt) Udowodnij, że podana na wykladzie sieć przełączników (sieć Beneša-Waksmana) jest asymptotycznie optymalna pod względem głębokości i liczby przełączników.

6. (1pkt) Załóżmy, że przełączniki ustawiane są losowo (każdy przełącznik z jednakowym prawdopodobieństwem ustawiany jest w jeden z dwóch stanów). Sieć zbudowaną z takich przełączników można traktować jako generator losowych permutacji. Udowodnij, że nie istnieje sieć przełączników, generująca permutacje z rozkładem jednostajnym.

Krzysztof Loryś