

Wstęp do informatyki

Lista 8

Dla zależności rekurencyjnych podanych w zadaniach 1, 2 i 3 wykonaj polecenia:

- Wyznacz wartość $T(16)$
- Rozwiąż zależność metodą podstawienia. W zadaniach 2. i 3. wystarczy, że uzyskasz rozwiązanie dla $n=2^k$ i naturalnego k .
- Podaj przykład algorytmu prezentowanego w ramach wykładu lub na liście ćwiczeniowej, którego złożoność czasową można opisać tą zależnością. Odpowiedź uzasadnij.

1. [1] $T(1) = 1$

$$T(n) = T(n-1) + n \text{ dla } n > 1$$

2. [1] $T(1) = 0$

$$T(n) = T(n/2) + 1 \text{ dla parzystego } n > 1$$

$$T(n) = T((n+1)/2) + 1 \text{ dla nieparzystego } n > 1$$

3. [1] $T(1) = 1$

$$T(n) = 2T(n/2) + 1 \text{ dla parzystego } n > 1$$

$$T(n) = T((n-1)/2) + T((n+1)/2) + 1 \text{ dla nieparzystego } n > 1$$

4. [2] Rozważmy następujący algorytm wyznaczania najmniejszej liczby w ciągu n -elementowym: Jeśli $n=1$ to minimum jest równe jednemu elementowi ciągu. W przeciwnym razie dzielimy ciąg na dwa podciągi (prawie) równej długości, wyznaczamy minima tych dwóch ciągów m_1 i m_2 a następnie wybieramy mniejszą z liczb m_1, m_2 .

a. Napisz funkcję rekurencyjną realizującą ten algorytm.

b. Podaj zależność rekurencyjną opisującą czas działania Twojego rozwiązania. Rozwiąż tę zależność dla n postaci $n=2^k$ i naturalnego k . Wynik porównaj z czasem działania „standardowego” iteracyjnego algorytmu wyznaczania minimum.

5. [1] Jak zadziała algorytm Quicksort uruchomiony dla ciągu:

a. uporządkowanego,

b. odwrotnie uporządkowanego (od największej wartości do najmniejszej),

c. ciągu złożonego z wielu powtórzeń jednego, tego samego elementu.

Przyjmij, że procedura `podzial` wybiera pierwszy element rozważanego podciągu jako wartość do podziału (pivot).

6. [1] Sortujemy ciągi złożone z 10, 100, 1000 elementów. Dla każdej z tych wartości ustal

a. głębokość drzewa wywołań rekurencyjnych, gdy stosujemy sortowanie przez scalanie;

b. najmniejszą i największą możliwą głębokość drzewa wywołań rekurencyjnych, gdy sortujemy stosując sortowanie szybkie (quicksort).

Podaj zależność rekurencyjną, której wartością dla danego n jest liczba poziomów drzewa wywołań rekurencyjnych sortowania przez scalenie ciągu n -elementowego.

7. [1] Zapoznaj się z problemem wież z Hanoi. Sformułuj jego specyfikację i przedstaw jego rekurencyjne rozwiązanie w postaci funkcji w języku C/Python.

Zadania dodatkowe, nieobowiązkowe (nie wliczają się do puli punktów do zdobycia na ćwiczeniach, punktacja została podana tylko jako informacja o trudności zadań wg wykładowcy)

8. [1] Podaj nierekurencyjny algorytm znajdowania najmniejszego i największego elementu w ciągu, w którym wykonywane jest co najwyżej $3/2$ n porównań elementów.
9. [0] Sortujemy metodą Quicksort ciąg złożony z 10 elementów. Przedstaw:
 - a. drzewo wywołań rekurencyjnych dla przypadku, gdy algorytm będzie działał najdłużej (gdy mamy „najgorsze” wyniki funkcji podział),
 - b. drzewo wywołań rekurencyjnych dla przypadku, gdy algorytm będzie działał najkrócej (gdy mamy „najlepsze” wyniki funkcji podział).
10. [2] Zaimplementuj algorytm Quicksort bez użycia rekurencji.

Wskazówka: Jeśli znasz strukturę danych nazywaną stosem, spróbuj ją zaimplementować i wykorzystać w tym zadaniu.

Uwaga: Jeśli to zadanie jest dla Ciebie zbyt łatwe, spróbuj zaimplementować Quicksort bez rekurencji, używając jedynie pamięci pomocniczej stałego rozmiaru. Takie rozwiązanie jest warte 4 punkty.
11. [2] Zaimplementuj sortowanie przez scalanie bez użycia rekurencji. Poza funkcją scalającą dwa ciągi, Twoje rozwiązanie powinno używać jedynie pamięci stałego rozmiaru.
12. [2] Podaj iteracyjne (nierekurencyjne) rozwiązanie problemu wież z Hanoi, wymagające jedynie pamięci pomocniczej stałego rozmiaru.
13. [0,5] Uzasadnij, że Quicksort (funkcja `qsort` podana na wykładzie) nie spełnia własności stopu w sytuacji gdy funkcja `podzial(a, l, p)` może zwracać jako wynik wartość p dla argumentów l, p takich, że $l < p$. Sprawdź też, czy sytuacja taka może mieć miejsce w przypadku funkcji `podzial` podanej na wykładzie.