Wstęp do Informatyki 2024/2025
Lista 8

Instytut Informatyki, Uniwersytet Wrocławski

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 $n = 2^k, k \in \mathbb{N}$.
- Podaj przykład algorytmu prezentowanego w ramach wykładu lub na liście ćwiczeniowej, którego złożoność czasową można opisać tą zależnością rekurencyjną. Odpowiedź uzasadnij.

1. [1]

$$\begin{aligned} &T(1) = 1 \\ &T(n) = T(n-1) + n, && n > 1 \end{aligned}$$

2. [1]

$$\begin{aligned} &T(1) = 0 \\ &T(n) = T(\tfrac{n}{2}) + 1, && \text{dla parzystego } n > 1 \\ &T(n) = T(\tfrac{n+1}{2}) + 1, && \text{dla nieparzystego } n > 1 \end{aligned}$$

3. [1]

$$\begin{aligned} &T(1) = 1 \\ &T(n) = 2T(\tfrac{n}{2}) + 1, && \text{dla parzystego } n > 1 \\ &T(n) = T(\tfrac{n+1}{2}) + T(\tfrac{n-1}{2}) + 1, && \text{dla nieparzystego } n > 1 \end{aligned}$$

4. [2] Uzasadnij, że asymptotycznie (czyli w notacji dużego-O) funkcje podane w zadaniach 2 i 3 są ograniczone przez funkcje wyznaczone w tych zadaniach metodą podstawienia dla każdego naturalnego argumentu n, nie tylko dla potęg dwójki.

   Wskazówka. Można np. udowodnić, że podane funkcje są monotoniczne i wykorzystać ten fakt. Należy jednak pamiętać, że w dowodzie monotoniczności można wykorzystywać tylko rekurencyjną postać funkcji, ponieważ zwarty „wzór”został udowodniony w zadaniach tylko dla n będących potęgami dwójki.

5. [1] Rozważmy następujący algorytm wyznaczania najmniejszej liczby w ciągu n-elementowym: Jeśli $n = 1$ to minimum jest równe jedynemu elementowi ciągu. W przeciwnym razie dzielimy ciąg na dwa podciągi (prawie) równej długości, wyznaczamy minima tych dwóch ciągów m_1, m_2 a następnie wybieramy mniejszą z liczb m_1, m_2.

(a) Napisz funkcję rekurencyjną realizującą ten algorytm.
(b) Podaj zależność rekurencyjną opisującą liczbę porównań elementów z ciągu wejściowego w Twoim rozwiązaniu. Rozwiąż tę zależność dla n postaci $n = 2^k$, gdzie $k \in \mathbb{N}$. Wynik porównaj z liczbą porównań w „standardowym” iteracyjnym algorytmie wyznaczania minimum.

6. [1] Jak zadziała algorytm Quicksort uruchomiony dla ciągu:

   (a) uporządkowanego,
   (b) odwrotnie uporządkowanego (od największej wartości do najmniejszej),
   (c) złożonego z wielu powtórzeń jednego, tego samego elementu.

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

7. [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 scalanie (mergesort) ciągu n-elementowego.

8. [1] Zapoznaj się z problemem wież z Hanoi. Sformułuj jego specyfikację i przedstaw jego rekurencyjne rozwiązanie w pseudokodzie oraz w postaci funkcji w języku C/Python.

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

9. [1] Podaj nierekurencyjny algorytm znajdowania najmniejszego i największego elementu w ciągu, w którym wykonywane jest co najwyżej $\frac{3}{2}n$ porównań elementów.

10. [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 `podzial`),
    (b) drzewo wywołań rekurencyjnych dla przypadku, gdy algorytm będzie działał najkrócej (gdy mamy „najlepsze” wyniki funkcji `podzial`).

11. [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.

12. [2] Zaimplementuj sortowanie przez scalanie bez użycia rekurencji. Poza funkcją scalającą dwa ciągi, Twoje rozwiązanie powinno używać jedynie pamięci stałego rozmiaru.

13. [2] Podaj iteracyjne (nierekurencyjne) rozwiązanie problemu wież z Hanoi, wymagające jedynie pamięci pomocniczej stałego rozmiaru.

14. [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.