ASD - ćwiczenia II

Notacja asymptotyczna i nie tylko

O notacji:

- $f(n) = O(g(n))$ – funkcję $f(n)$ można ograniczyć „od góry", dla $n \to \infty$ od pewnego $n^* < \infty$, przez funkcję $c \cdot g(n)$, gdzie $c \in \mathbb{R}^+$,
- $f(n) = \Omega(g(n))$ – funkcję $f(n)$ można ograniczyć „od dołu", dla $n \to \infty$ od pewnego $n^* < \infty$, przez funkcję $c \cdot g(n)$, gdzie $c \in \mathbb{R}^+$,
- $f(n) = \Theta(g(n))$ – funkcję $f(n)$ można ograniczyć:
  - „od góry", dla $n \to \infty$ od pewnego $n_1^* < \infty$, przez funkcję $c_1 \cdot g(n)$, gdzie $c_1 \in \mathbb{R}^+$,
  - „od dołu", dla $n \to \infty$ od pewnego $n_2^* < \infty$, przez funkcję $c_2 \cdot g(n)$, gdzie $c_2 \in \mathbb{R}^+$,
  - **wniosek:** funkcje $f(n)$ i $g(n)$ są tego samego rzędu.

Zadania

1. Określ (TAK/NIE) czy podane ograniczenia funkcji $f(n)$ są poprawne:

   (a) $f(n) = \Theta(\sqrt{n})$, $f(n) = O\left(n^{\frac{3}{2}} \log(n)\right)$, $f(n) = \Omega\left(\left|\frac{1}{n} - 1\right|\right)$,
   gdzie $f(n) = n \sin(n)$,

   (b) $f(n) = \Theta(n \log(n))$, $f(n) = O\left(n^{\log(3)}\right)$, $f(n) = \Omega(n\sqrt{n})$,
   gdzie $f(n) = \log\left(n^{\sqrt{n}}\right)$,

   (c) $f(n) = \Theta\left(n^2\right)$, $f(n) = O(n \log(\log(n)))$, $f(n) = \Omega\left(\log\left(\frac{n!}{n}\right)\right)$,
   gdzie $f(n) = \sqrt{n}\,|\sin(n)|$,

   (d) $f(n) = \Theta\left(n^{\sqrt{3}}\right)$, $f(n) = O\left(n^{\frac{5}{2}}\right)$, $f(n) = \Omega\left(\log^2\left(\frac{1}{2}n\right)\right)$,
   gdzie $f(n) = n^2 \log(n)$.

2. Oszacuj za pomocą notacji Θ i uporządkuj niemalejąco następujące funkcje zmiennej n:

   (a) $f_1(n) = \log^2(3n)$, $f_2(n) = n \log(n!) - n$, $f_3(n) = \sqrt{n} + n$,
   $f_4(n) = 3^{\sqrt{n}} + n \log(n)$, $f_5(n) = n\sqrt{\cos^2(n)}$, $f_6(n) = \frac{3}{n} + 2$,

   (b) $f_1(n) = n^2 + \sin^2\left(\frac{n}{2}\right)$, $f_2(n) = \log\left(n^3\right) - n$, $f_3(n) = 2^n + \log(n!)$,
   $f_4(n) = n \log(n) + 2\sqrt{n}$, $f_5(n) = 8^{\log(n)} + n$, $f_6(n) = \log\left(\log\left(n^2\right)\right)$,

   (c) $f_1(n) = 3^{2n} + 3^n$, $f_2(n) = \log(n) + n^{\frac{1}{4}}$, $f_3(n) = n^3 \log(n) + n^2\,|\sin(2n)|$,
   $f_4(n) = 2^n + n2^{\log(n)}$, $f_5(n) = \sqrt{\cos^2(n)} + \log^2(n)$, $f_6(n) = \log\left(n^{2^{\log(n)}}\right) + \frac{1}{n^2}$,

   (d) $f_1(n) = n \log(n) + 2^{\frac{1}{2}\log(n)}$, $f_2(n) = \log(n!) + 3n$, $f_3(n) = 4^{n+2} + 2^{3n}$,
   $f_4(n) = \sqrt{n} + \frac{1}{2}\log\left(n^{2+\sqrt{n}}\right)$, $f_5(n) = n + \sin^2(n)$, $f_6(n) = 4n^3 - \sqrt{|\log(n!)|}$.

3. Niech $\mathcal{A}$ będzie algorytmem, którego złożoność wyraża się pewną określoną funkcją $f(n)$, gdzie n jest rozmiarem danych wejściowych. Czas wykonania algorytmu $\mathcal{A}$ dla danych rozmiaru x na komputerze $\mathcal{K}$ wynosi t sekund. Oblicz:

   - ile czasu zajmie wykonanie algorytmu $\mathcal{A}$ dla danych rozmiaru p-krotnie mniejszego na komputerze $\mathcal{K}$,
   - maksymalny rozmiar danych jakie algorytm $\mathcal{A}$ możne przetworzyć na komputerze $\mathcal{K}$ w ciągu t' sekund,

- czas w jakim komputer $\mathcal{K}'$ p'-krotnie szybszy od komputera $\mathcal{K}$ obliczy rezultat algorytmu dla danych wejściowych rozmiaru x',

gdzie:

(a) $f(n) = 5n,\ x = 12,\ t = 60,\ p = 5,\ t' = 20,\ p' = 2,\ x' = 100,$
(b) $f(n) = \log(n),\ x = 128,\ t = 7,\ p = 2,\ t' = 20,\ p' = 8,\ x' = 512,$
(c) $f(n) = n^3,\ x = 6,\ t = 54,\ p = 3,\ t' = 432,\ p' = 3,\ x' = 10,$
(d) $f(n) = 2^n,\ x = 6,\ t = 64,\ p = 3,\ t' = 40,\ p' = 4,\ x' = 8.$

4. Która z wymienionych własności jest zawsze prawdziwa, jeżeli wiadomo, że:

   - $g(n) = O(f(n))$ i $h(n) = O(f(n))$,
   - $g(n) = O(f(n))$ i $h(n) = \Omega(f(n))$,

   gdzie $f(n)$, $g(n)$, $h(n)$ są funkcjami określonymi nad zbiorem liczb rzeczywistych:

   (a) $g(n) + h(n) = O(f(n))$,
   (b) $g(n) - h(n) = \Omega(f(n))$,
   (c) $g(n) \cdot h(n) = O(f(n))$,
   (d) $g(n)/h(n) = \Omega(f(n))$.

Zadanie o fałszywej monecie

W pewnym pudełku C znajduje się $n > 1$ identycznie wyglądających monet $C[1], C[2], \ldots, C[n]$, z których $n-1$ zostało wykonanych ze złota a dokładnie jedna jest falsyfikatem odlanym z ołowiu (masę pojedynczej monety $C[i]$ oznaczamy za pomocą symbolu $\|C[i]\|$). Każdą z monet opisuje niezdefiniowana struktura danych typu `Coin`:

```
typedef str_Coin Coin;

struct str_Coin Coin {
  ...
}
```

Jedynym sposobem na wykrycie fałszywej monety jest porównanie jej masy względem innych monet na specjalnej wadze szalkowej przy czym, każda z szalek może pomieścić dowolną liczbę monet. Zakładamy, że dysponujemy funkcją

```
int WEIGHT(Coin C[], int l1, int r1, int l2, int r2),
```

której parametry wejściowe to:

- $l1, r1$ - indeksy krańcowe zbioru kolejnych monet znajdujących się na pierwszej szalce,
- $l2, r2$ - indeksy krańcowe zbioru kolejnych monet znajdujących się na drugiej szalce,

przy wartościach zwracanych:

- 0 - gdy $\sum_{i=l1}^{r1} \|C[i]\| = \sum_{i=l2}^{r2} \|C[i]\|$, tj. sumaryczne masy monet znajdujących się na obu szalkach są równe,

- 1 - gdy $\sum_{i=l1}^{r1} \|C[i]\| > \sum_{i=l2}^{r2} \|C[i]\|$, tj. sumaryczna masa monet znajdujących się na pierwszej szalce jest większa niż sumaryczna masa monet znajdujących się na drugiej szalce,
- 2 - gdy $\sum_{i=l1}^{r1} \|C[i]\| < \sum_{i=l2}^{r2} \|C[i]\|$, tj. sumaryczna masa monet znajdujących się na drugiej szalce jest większa niż sumaryczna masa monet znajdujących się na pierwszej szalce.

Zaproponuj algorytm postępowania w postaci funkcji

`int FORGERY(Coin C[], int n)`,

który pozwoli na możliwie szybkie wykrycie monety fałszywej jedynie za pomocą operacji `WEIGHT()`. Określ złożoność podanej metody postępowania względem liczby przeprowadzonych czynności ważenia, tj. liczby wywołań funkcji `WEIGHT()`.