**Zadanie 1 (25%)**
Załóżmy, że do rozwiązania pewnego problemu masz wybrać jeden z 3 algorytmów:
**Algorytm** $A$ rozwiązuje problem poprzez rekurencyjne rozwiązanie 7 pod-problemów trzy razy mniejszych, a następnie scala ich rozwiązania w czasie $O(n \log n)$.
**Algorytm** $B$ rozwiązuje problem wielkości $n$ przez rekurencyjne rozwiązywanie dwóch pod-problemów o rozmiarze $n-3$, następnie scalając ich rozwiązania w czasie stałym.
**Algorytm** $C$ rozwiązuje problem wielkości $n$ poprzez rekurencyjne rozwiązanie pod-problemów o rozmiarach $n/3$ i $n/2$, następnie scalając ich rozwiązania w czasie $O(n^2)$.
Podaj asymptotyczną złożoność obliczeniową tych algorytmów i który algorytm byś użył. Odpowiedz uzasadnij.

*Rozwiązanie:*
Dla algorytmu $A$ możemy ułożyć następujące równanie rekurencyjne: $A(n) = 7A\left(\frac{n}{3}\right) + O(n\log(n))$. Do równania tego nie możemy bezpośrednio użyć podanego na wykładzie Master Theorem. Przyjrzyjmy się natomiast następującym równaniom rekurencyjnym:

$$A_1(n) = 7A_1\left(\frac{n}{3}\right) + O(n)$$
$$A_2(n) = 7A_2\left(\frac{n}{3}\right) + O(n^{3/2})$$

Patrząc na koszt dzielenia/łączenia rozwiązań możemy zauważyć, że jeśli $f(n) = O(n)$ to $f(n) = O(n\log(n))$ oraz jeśli $g(n) = O(n\log(n))$ to $g(n) = O(n^{3/2})$. Z tego wiemy, że $A_1(n) = O(A(n))$ oraz $A(n) = O(A_2(n))$.
Użyjmy teraz Master Theorem do obliczenia złożoności asymptotycznej

- $A_1(n)$: $a = 7, b = 3, d = 1$, więc $\log_3(7) > 1$, bo $\log_3(7) \approx 1.77124$, zatem $A_1(n) = O\left(n^{\log_3(7)}\right)$.
- $A_2(n)$: $a = 7, b = 3, d = \frac{3}{2}$, więc $\log_3(7) > \frac{3}{2}$, bo $\log_3(7) \approx 1.77124$, zatem $A_2(n) = O\left(n^{\log_3(7)}\right)$.

Wnioski: Jeśli $A_1(n) = O(A(n)), A(n) = O(A_2(n))$ oraz $A_1(n)$ jest asymptotycznie (w sensie dużego $O$) równe $A_2(n)$ to $A(n)$ jest również asymptotycznie (w sensie dużego $O$) im równe, czyli $A(n) = O\left(n^{\log_3(7)}\right)$.

Dla algorytmu $B$ możemy ułożyć następujące równanie rekurencyjne: $B(n) = 2B\,(n-3)+c$, gdzie $c = \Theta(1)$ jest stałym czasem scalania. Rozwiążemy tę rekurencję metodą iteracyjną:

$$\begin{aligned} B(n) &= c + 2B(n-3) \\ &= c + 2(c + 2B(n-6)) \\ &= c + 2^1c + 2^2c + 2^3B(n - 3\cdot 3)) \end{aligned}$$

Iterujemy tak aż osiągniemy warunek brzegowy, $B(1) = \Theta(1)$.
$i$-ty składnik sumy wynosi $2^{i-1}c$, a iterowanie kończymy gdy gdy $n - 3\cdot i = 1$ lub równoważnie, gdy $i$ przekracza $\frac{n-1}{3}$. Zatem

$$B(n) = O\left(c + 2^1c + 2^2c + \ldots + 2^{\frac{n}{3}}c\right) = O(2^{n/3})$$

Dla algorytmu $C$ możemy ułożyć następujące równanie rekurencyjne: $C(n) = C\left(\frac{n}{3}\right) + C(\frac{n}{2}) + O(n^2)$. W celu rozwiązania tej rekurencji możemy użyć np. drzewa rekursji (jest szansa na trochę zawiłe obliczenia) lub podejścia podobnego do tego użytego dla algorytmu $A$. Łatwo zauważyć, że koszt scalania będzie tutaj dominujący, bo rekurencyjne podproblemy są mniejsze niż wejściowy problem, ale scalanie kosztuje aż $O(n^2)$. Zatem zdefiniujmy sobie następujące równania rekurencyjne:

$$C_1(n) = 2C_1\left(\frac{n}{3}\right) + O(n^2)$$
$$C_2(n) = 2C_2(\frac{n}{2}) + O(n^2).$$

Ponownie możemy zauważyć, że $C_1(n) = O(C(n))$ i $C(n) = O(C_2(n))$ przez to, że mamy odpowiednio mniejszy lub większy rozmiar podproblemów niż w oryginalnym algorytmie $C$.

Użyjmy teraz Master Theorem do obliczenia złożoności asymptotycznej

- $C_1(n)$: $a = 2, b = 3, d = 2$, więc $\log_3(2) < 2$, bo $\log_3(2) \approx 0.63093$, zatem $C_1(n) = O\left(n^2\right)$.
- $C_2(n)$: $a = 2, b = 2, d = 2$, więc $\log_2(2) < 2$, bo $\log_2(2) = 1$, zatem $C_2(n) = O\left(n^2\right)$.

Wnioski: Jeśli $C_1(n) = O(C(n)), C(n) = O(C_2(n))$ oraz $C_1(n)$ jest asymptotycznie (w sensie dużego $O$) równe $C_2(n)$ to $C(n)$ jest również asymptotycznie (w sensie dużego $O$) im równe, czyli $C(n) = O\left(n^2\right)$.

Użyłbym algorytmu $A$ ponieważ ma on najmniejszą asymptotyczną złożoność obliczeniową.