

ALGORYTMY I STRUKTURY DANYCH

METODY UKŁADANIA ALGORYTMÓW METODA „DZIEL I ZWYCIĘŻAJ”

Metoda polega na tym, że zamiast rozwiązywać wejściowy problem, który może być problemem trudnym, próbuje się wskazać *jeden lub więcej innych problemów*, które są łatwiejsze do rozwiązania, a ich rozwiązanie pozwala na skonstruowanie rozwiązania problemu wejściowego.

Algorytm typu „dziel i zwyciężaj” dzieli problem na **niezależne** podproblemy, rozwiązuje je **rekurencyjnie**, a następnie **łączy** rozwiązania wszystkich podproblemów w celu utworzenia rozwiązania pierwotnego problemu.

Dziel: Dzielimy problem na pewną liczbę podproblemów, stanowiących mniejsze egzemplarze tego samego problemu.

Zwyciężaj: Rozwiązujemy podproblemy rekurencyjnie. Jeśli jednak rozmiary podproblemów są dostatecznie małe, to rozwiązujemy podproblemy bezpośrednio.

Połącz: Łączymy rozwiązania podproblemów w rozwiązanie pierwotnego problemu.

Formalny zapis metody

dziel_i_zwyciezaj(N)

jeśli N wystarczająco małe

zwróć *Przypadek_elementarny(N)*;

w przeciwnym przypadku

Podziel $Pb(N)$ na mniejsze egzemplarze: $Pb(N_1), Pb(N_2), \dots, Pb(N_k)$;

dla $i = 1, \dots, k$

oblicz wynik cząstkowy $w_i = \text{dziel_i_zwyciezaj}(N_i)$;

zwróć *PołączRozwiazania*(w_1, w_2, \dots, w_k);

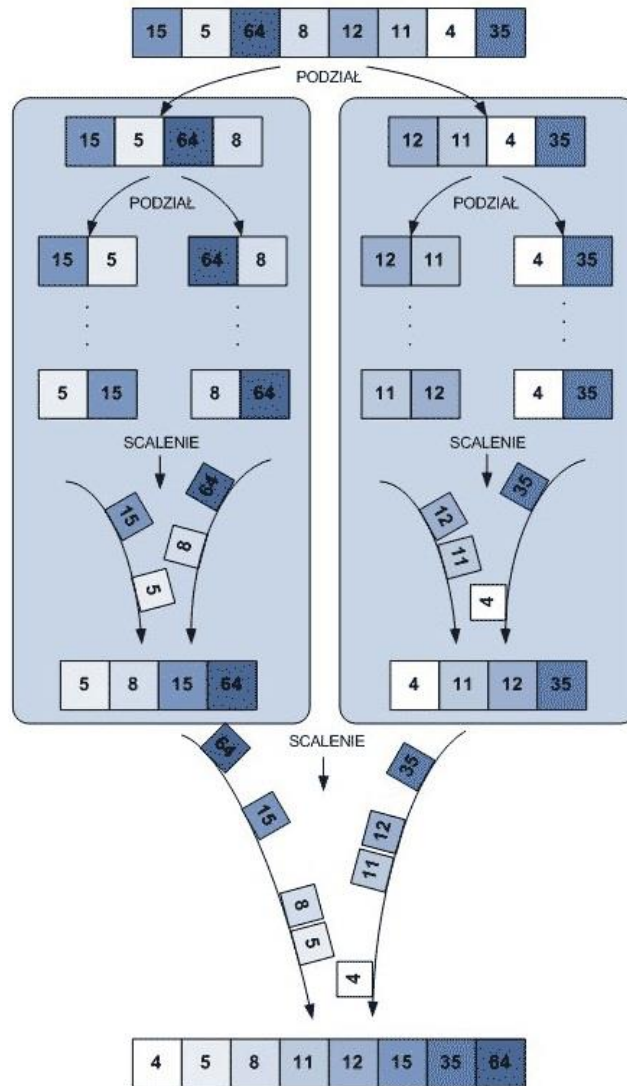
Zadanie 1. Wykonaj analizę działania algorytmu sortowania szybkiego (*Quicksort*) na przykładowych danych. (*Pliki do wykorzystania*: [zadania_dziel_i_rzadz.xlsx](#), arkusz [zadanie_1](#)).

Zadanie 2. Zaimplementuj funkcję *Merge()* scalającą dwa posortowane wektory. (*Pliki do wykorzystania*: [merge.cpp](#)).

Wykonaj analizę działania algorytmu sortowania przez scalanie (*Mergesort*) na przykładowych danych. (*Pliki do wykorzystania*: [zadania_dziel_i_rzadz.xlsx](#), arkusz [zadanie_2](#)).

Zadanie 3. Stosując metodę "dziel i zwyciężaj", ułóż algorytm wyznaczania największego elementu wektora.

Zadanie 4. Dana jest tablica n liczb całkowitych. Przedstaw algorytm liczący sumę elementów w tablicy z zastosowaniem metody „dziel i zwyciężaj”.



ANALIZA ZŁOŻONOŚCI OBLICZENIOWEJ SORTOWANIA PRZEZ SCALANIE

Założmy, że rozmiar pierwotnego problemu n jest potęgą dwójki. Wtedy w *każdym kroku* podziału dzielimy problem na podproblemy rozmiaru dokładnie $n/2$.

Dziel: W kroku dzielenia znajdujemy środek przedziału, co zajmuje czas stały, więc $\Theta(1)$.

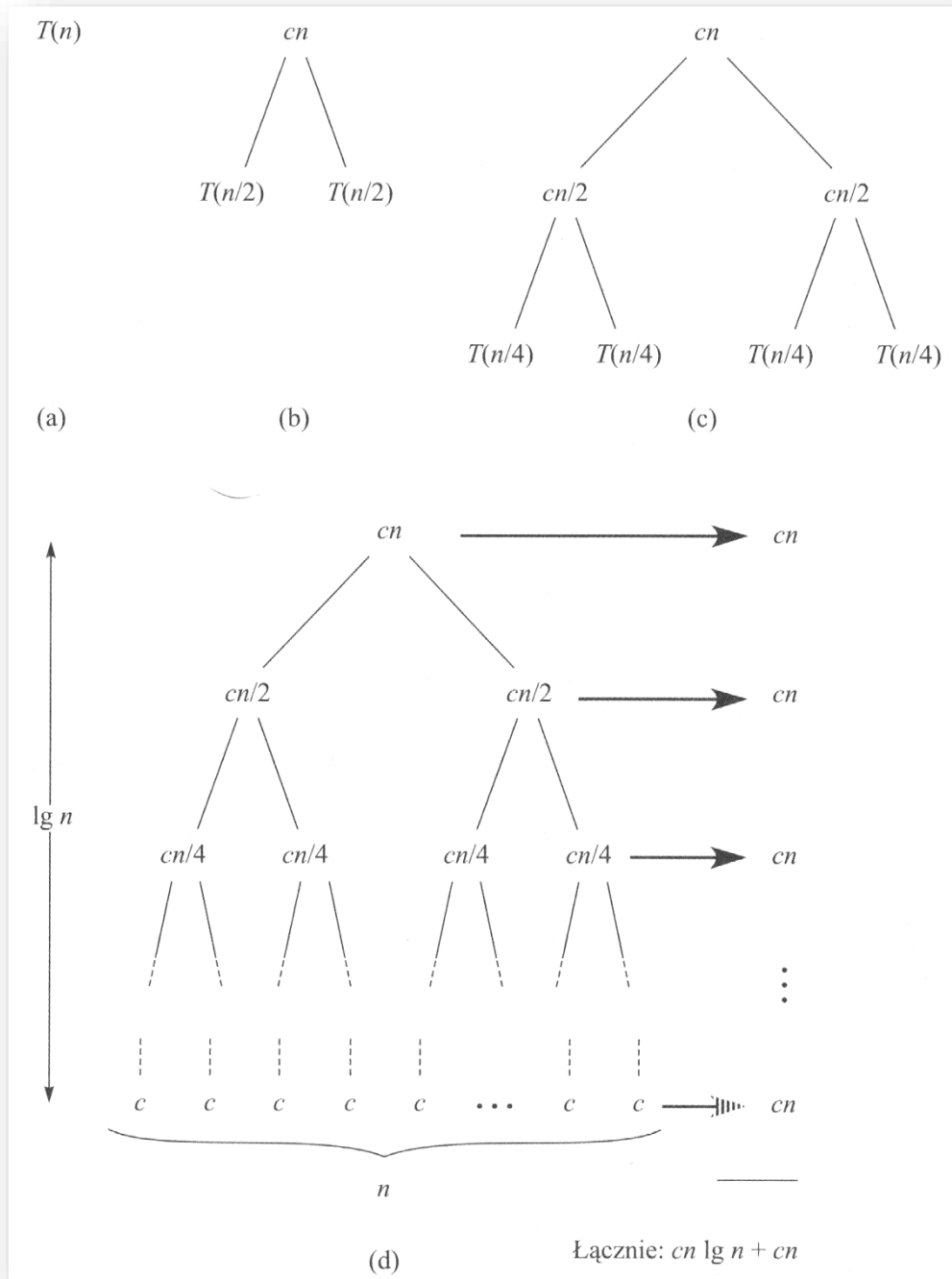
Zwycięzaj: Rozwiązujemy rekurencyjnie dwa podproblemy, każdy rozmiaru $n/2$, co daje w sumie czas $2 * T(n/2)$.

Połącz: Procedura `merge()` działa dla n elementowej podtablicy w czasie $\Theta(n)$.

Czas działania sortowania przez scalanie w przypadku pesymistycznym wynosi:

$$T(n) = \begin{cases} \Theta(1) & \text{jeśli } n = 1 \\ 2T(n/2) + \Theta(n) & \text{jeśli } n > 1 \end{cases} = \begin{cases} c & \text{jeśli } n = 1 \\ 2T(n/2) + cn & \text{jeśli } n > 1 \end{cases}$$

$$T(n) = \begin{cases} c & \text{jeśli } n = 1 \\ 2T(n/2) + cn & \text{jeśli } n > 1 \end{cases}$$



Zsumujemy teraz koszty na każdym poziomie drzewa. Koszt na najwyższym poziomie to cn , drugi poziom wnosi koszt $c \left(\frac{n}{2}\right) + c \left(\frac{n}{2}\right) = cn$, kolejny poziom wnosi koszt $c \left(\frac{n}{4}\right) + c \left(\frac{n}{4}\right) + c \left(\frac{n}{4}\right) + c \left(\frac{n}{4}\right) = cn$. Na najniższym poziomie jest n węzłów, każdy wnoszący koszt c , co w sumie daje koszt cn .

Łączna liczba poziomów "drzewa rekursji" jest równa $\log_2 n + 1$, a każdy poziom wnosi koszt cn , co daje łączny koszt $cn(\log_2 n + 1) = cn \log_2 n + cn$. Pomijając składnik niższego rzędu i stałą c , otrzymujemy wynik $\Theta(n \log_2 n)$.