Notatki z Algorytmów i Struktur Danych

Jakub Kogut

10 marca 2025

Spis treści

1 Wstęp

To będa notatki z przedmiotu Algorytmy i struktury danych na Politechnice Wrocławskiej na kierunku Informatyka Algorytmiczna rok 2025 semestr letni.

1.1 Informacje

Prowadzący Przedmiot: **Zbychu Gołębiewski**

- Należy kontaktować się przez maila: mail
- Konsultacje **216/D1**:
  - Wtorek 13:00-15:00
  - Środa 9:00-11:00
- Wiecej info na stronie przedmiotu
- Literatura
  - Algorithms, Dasgupta, Papadimitriou, Vazirani
  - Algorithms, Sedgewick, Wayne (strona internetowa książki)
  - Algorithms Designs, Jon Kleinberg and Eva Trados
  - Wprowadzenie do algorytmów, Cormen, Leiserson, Rivest, Stein
  - Sztuka programowania (wszystkie tomy), Donald E. Knuth

1.2 Ocenianie

Ocena z kursu składa się z:

- Oceny z egzaminu – E
- Oceny z ćwiczeń – C
- Oceny z laboratorium – L

Wszystkie oceny są z zakresu $[0, 100]$. Ocena końcowa jest wyliczana ze wzoru:

$$K = \frac{1}{2}E + \frac{1}{4}C + \frac{1}{4}L$$

2 Wykład 2025-03-03

2.1 Przykładowy Problem

Sortowanie:

- Input: n liczb $a_1, a_2, \ldots, a_n, |A|$, gdzie $|A|$ to długośc tablicy
- Output: permutacja $a'_1, a'_2, \ldots, a'_n$ taka, że $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Najważniejsze w algorytmach jest to, żeby były POPRAWNE: edge case, ...

2.2 Jak mierzyć złożoność algorytmów

1. Worst Case Analysis T(n) $\leftarrow$ stosowane najcześciej
2. Average Case Analysis
   - zakładamy pewnien rozkład prawdopodobieństwa na danych wejściowych
   - T – zmienna losowa liczby operacji wykonanych przez algorytm

     $$T(n) = \max\{\#\text{operacji dla danego wejścia}\}$$

   - $E[T]$ – wartość oczekiwana $T \rightarrow$ średnia liczba operacji, to co nas interesuje

2.3 Przykład algorytmu

W tej sekcji mamy pokazany przykład jak pisać pseudo kod:

Algorithm 1 Merge Sort

```
1: procedure MergeSort(A, 1, n)
2:     if |A[1..n]| == 1 then
3:         return A[1..n]
4:     else
5:         B = MergeSort(A, 1, ⌊n/2⌋)
6:         C = MergeSort(A, ⌊n/2⌋, n)
7:         return Merge(B, C)
8:     end if
9: end procedure
```

Algorithm 2 Merge

```
1: procedure Merge(X[1..k], Y[1..n])
2:     if X = ∅ then
3:         return Y
4:     else if Y = ∅ then
5:         return X
6:     else if X[1] ≤ Y[1] then
7:         return [X[1]] × Merge(X[2..k], Y[1..n])
8:     else
9:         return [Y[1]] × Merge(X[1..k], Y[2..n])
10:    end if
11: end procedure
```

2.4 Przykład działania Merge Sort

Example: Sorting the array [10, 2, 5, 3, 7, 13, 1, 6] **step by step**

1. **Initial split:**

$$[\,10,\,2,\,5,\,3,\,7,\,13,\,1,\,6\,] \quad \longrightarrow \quad [\,10,\,2,\,5,\,3\,] \quad \text{and} \quad [\,7,\,13,\,1,\,6\,].$$

2. **Sort the left half** [10, 2, 5, 3]**:**
   (a) Split into [10, 2] and [5, 3].
   (b) MergeSort([10, 2]):
   - Split into [10] and [2].
   - Each is already sorted (single element).
   - Merge: [2, 10].

   (c) MergeSort([5, 3]):
   - Split into [5] and [3].
   - Each is already sorted.
   - Merge: [3, 5].

   (d) Merge [2, 10] and [3, 5] to get [2, 3, 5, 10].

3. **Sort the right half** [7, 13, 1, 6]**:**
   (a) Split into [7, 13] and [1, 6].

(b) MergeSort([7, 13]):
- Split into [7] and [13].
- Each is already sorted.
- Merge: [7, 13].

(c) MergeSort([1, 6]):
- Split into [1] and [6].
- Each is already sorted.
- Merge: [1, 6].

(d) Merge [7, 13] and [1, 6] to get [1, 6, 7, 13].

4. **Final merge:** Merge the two sorted halves:

$$[\,2,\,3,\,5,\,10\,] \quad \text{and} \quad [\,1,\,6,\,7,\,13\,] \quad \longrightarrow \quad [\,1,\,2,\,3,\,5,\,6,\,7,\,10,\,13\,].$$

Hence, after all the recursive splits and merges, the final sorted array is:

$$[\,1,\,2,\,3,\,5,\,6,\,7,\,10,\,13\,].$$

2.5 Złożoność Merge Sort

- Złożoność czasowa
  - $T(n) = 2T(n/2) + \Theta(n)$
  - $T(n) = \Theta(n \log n)$
- Złożoność pamięciowa
  - $M(n) = n + M(n/2)$
  - $M(n) = \Theta(n)$

3 Wykład 2025-03-10

3.1 Notacja Asypmtotyczna

Na wykładzie będziemy omawiali:

- Notację dużego O $O(n)$ //ograniczenie górne
  - Definicja $O(n)$:

$$O(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq f(n) \leq c \cdot g(n)\}$$

  - Uwaga!
    Jeśli

$$\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

    to

$$\limsup_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f(n)}{g(n)}$$

  - Przykład:
    * $2n^2 = O(n^3)$ dla $n_0 = 2, c = 1$ Definicja jest spełniona
    * $f(n) = n^3 + O(n^2)$ jest to jeden z sposobów użycia $O(n)$

$$\exists h(n) = O(n^2) \quad \text{takie, że} \quad f(n) = n^3 + h(n)$$

- Notację omega //ograniczenie dolne
  - Definicja
$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq c \cdot g(n) \leq f(n)\}$$
  - Przykład
    * $n^3 = \Omega(2n^2)$
    * $n = \Omega(\log n)$
- Notację theta $\theta(n)$ //ograniczenie z dwóch stron
  - Definicja
$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$
  - Przykład
    * $n^3 = \Theta(n^3)$
    * $n^3 = \Theta(n^3 + 2n^2)$
    * $logn + 8 + \frac{1}{12n} = \Theta(\log n)$
  - Uwaga!
$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$
    Można to zapisać jako klasy funkcji:
$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$
- Patologiczny przykład: mamy funkcje $g(n) = n$ oraz $f(n) = n^{1+sin\frac{\pi n}{2}}$, a więc
$$f(n) = \begin{cases} n^2 & \text{dla n parzystych} \\ n & \text{dla n nieparzystych} \end{cases}$$
  wtedy
$$\limsup_{n\to\infty} \frac{f(n)}{g(n)} = \infty \limsup_{n\to\infty} \frac{g(n)}{f(n)} = \infty$$
  zatem $f \neq O(g)$ oraz $g \neq O(f)$
- o małe
  - Definicja
$$o(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq f(n) < c \cdot g(n)\}$$
    Równoważnie
$$lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$
  - Przykład
    * $n^2 = o(n^3)$ i $n^2 O(n^3)$ ale $n^2 \neq o(n^2)$
    * $n = o(n^2)$

3.2 Rekurencja

- Metoda podstawienia (metoda dowodu indukcyjnego)

  1. Zadnij Odpowiedź (bez stałych)
  2. Sprawdź przez indukcję czy odpowiedź jest poprawna
  3. Wylicz stałe

  - Przykład
    * $T(n) = T(\frac{n}{2}) + n$
    * Pierwotny strzał: $T(n) = O(n^3)$
    * cel: Pokazać, że $\exists c > 0 : T(n) \leq c \cdot n^3$
      · warunek początowy: $T(1) = 1 \leq c$
      · krok indukcyjny: załóżmy, że $\forall k \leq n : T(k) \leq ck^3$

      $$T(n) = 4T(\frac{n}{2}) + n \leq 4c(\frac{n}{2})^3 + n = \frac{1}{2}cn^3 + n \leq cn^3 \quad \text{dla} \quad c \geq 2$$

      jednakże "Przestrzeliliśmy" znacznie, spróbojmy wzmocnić założenie indukcyjne:

      $$T(n) \leq c_1k^2 - c_2k, k < n$$

      wtedy mamy:

      $$T(n) = 4T(\frac{n}{2}) + n \leq 4(c_1(\frac{n}{2})^2 - c_2(\frac{n}{2})) + n = c_1n^2 - 2c_2n + n \leq c_1n^2 - c_2n$$

      zatem $c_1 = 1, c_2 = 1$ i $T(n) = O(n^2)$ □
  - Przykład
    * $T(n) = 2T(\sqrt{n}) + \log n$
      załóżmy, że n jest potęgą liczby 2, czyli $n = 2^m$

      $$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

      Co implikuje

      $$T(2^{\frac{m}{2}}) \to S(m)$$

      wtedy

      $$S(m) = 2S(\frac{m}{2}) + m$$

      rozwiązując rekurencję otrzymujemy

      $$S(m) = m \log m$$

      zatem

      $$T(n) = \log n \log \log n$$

4 Ćwiczenia

tu beda pojawialy sie notatki z cwiczen do przedmiotu Algorytmy i struktury danych na Politechnice Wrocławskiej na kierunku Informatyka Algorytmiczna rok 2025 semestr letni.

4.1 Lista 2

robiona na zajęciach 2025-03-10

```
1: function f(n)
2: if n > 1 then
3:     print_line('still going')
4:     f(n/3)
5:     f(n/3)
6: end if
```

4.1.1 zadanie 1

Wylicz ile linijek wypisze poniższy program (podaj wynik będacy funkcją od n w postaci asymptotycznej $\Theta(\cdot)$). Można założyć, że n jest potęgą liczby 3. w pseudo kodzie pojawia sie nastepujaca rekurencja:

$$T(n) = 2T(\frac{n}{3}) + 1$$

rozwiąże ją używając metody podstawienia. Niech $n = 3^k, k = \log_3 n$, wtedy:

$$T(3^k) = 2T(3^{k-1}) + 1$$

Zatem przyjmując $S(k) = T(3^k)$ mamy:

$$S(k) = 2S(k-1) + 1$$

rozwiązując rekurencję otrzymujemy:

$$S(k) = 2^k - 1$$

zatem

$$T(n) = 2^{\log_3 n} - 1 = n^{\log_3 2} - 1 = \Theta(n^{\log_3 2})$$

analogicznie liczmy jaka jest wykonana "praca" wykonana przez program w drzweie rekursji.

4.1.2 zadanie 2

Niech $f(n)$ i $g(n)$ będą funkcjami asymptotycznie nieujemnymi (tzn. nieujemnymi dla dostatecznie dużego n). Korzystając z definicji notacji Θ, udowodnij, że:

$$\max\{f(n), g(n)\} = \Theta(f(n) + g(n)).$$

Dowód. Z definicji notacji Θ mamy:

$$f(n) = \Theta(g(n)) \iff \exists c_1, c_2 > 0, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

skoro $f(n)$ i $g(n)$ są asymptotycznie nieujemne to:

$$\exists n_f : \forall n \geq n_f, f(n) \geq 0$$

$$\exists n_g : \forall n \geq n_g, g(n) \geq 0$$

zatem

$$n_0 = \max\{n_f, n_g\}$$

a więc

$$f(n) \leq \max\{f(n), g(n)\}$$

$$g(n) \leq \max\{f(n), g(n)\}$$

dodając obie nierówności otrzymujemy:

$$f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\}$$

zatem

$$\forall n \geq n_0 : \max\{f(n), g(n)\} \leq f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\}$$

a więc z definicji mamy

$$\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$$

□

4.1.3 zadanie 3

Wylicz asymptotyczną złożoność (używając notacji Θ) poniższych fragmentów programów:

Algorithm 3 Pierwszy fragment kodu

```
1: for i = 1 to n do
2:     j = i
3:     while j < n do
4:         sum = P(i, j)
5:         j = j + 1
6:     end while
7: end for
```

Algorithm 4 Drugi fragment kodu

```
1: for i = 1 to n do
2:     j = i
3:     while j < n do
4:         sum = R(i, j)
5:         j = j + j
6:     end while
7: end for
```

Gdzie:

- koszt wykonania procedury $P(i, j)$ wynosi $\Theta(1)$,
- koszt wykonania procedury $R(i, j)$ wynosi $\Theta(j)$.

Dowód.

- Pierwszy fragment kodu
  - Wewnętrzna pętla wykonuje się $n - i$ razy
  - Koszt wykonania procedury $P(i, j)$ wynosi $\Theta(1)$
  - Zatem koszt wykonania wewnętrznej pętli wynosi $\Theta(n - i)$
  - Zatem koszt wykonania całego fragmentu wynosi

$$\sum_{i=1}^{n} \Theta(n - i) = \Theta(n^2)$$

- Drugi fragment kodu
  - Wewnętrzna pętla wykonuje się $\log_2 n$ razy
  - Koszt wykonania procedury $R(i, j)$ wynosi $\Theta(j)$
  - Zatem koszt wykonania wewnętrznej pętli wynosi $\Theta(\log_2 n)$
  - Zatem koszt wykonania całego fragmentu wynosi

$$\sum_{i=1}^{n} \Theta(\log_2 n) = \Theta(n \log_2 n)$$

□

5 Podsumowanie

Podsumowanie lub zakończenie notatek.