AiSD

Rafał Włodarczyk

INA 4, 2025

Contents

1 Lecture I - Sortowanie

Definiujemy problem:

1. Input: $A = (a_1, \ldots, a_n), |A| = n$
2. Output: Permutacja tablicy wyjściowej $(a'_1, a'_2, \ldots, a'_n)$, takie że: $a'_1 \leqslant a'_2 \leqslant \cdots \leqslant a'_n$.

1.1 Worst-case analysis

$$T(n) = \max_{\text{wszystkie wejścia}} \{\#\text{operacji po wszystkich } |\text{n}|\text{-wejściach}\} \tag{1}$$

1.2 Average-case analysis

Zakładamy pewien rozkład prawdopodobieństwa na danych wejściowych. Z reguły myślimy o rozkładzie jednostajnym. Niech T - zmienna losowa liczby operacji wykonanych przez badany algorytm.

$$\mathbf{E}(T) - \text{wartość oczekiwana T} \tag{2}$$

Później możemy badać wariancję, oraz koncentrację.

1.3 Analiza losowego sortowania

Dla poprzedniego algorytmu zobaczmy, że: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ $\left[\text{czyli } f(n) \sim g(n) \equiv \lim_{n\to\infty} \frac{f(n)}{g(n)} = 1\right]$.
To jest tragiczna złożoność.

1.4 Insertion Sort (A, n)

$(A, n) = ((a_1, a_2, \ldots, a_n), n)$

```
for j = 2...n
{
    key = A[j]
    i=j-1
    while(i>0 && A[i]>key) {
        A[i+1] = A[i]
        i = i - 1
    }
    A[i+1] = key
}
```

Przykład: $A = (8, 2, 4, 9, 3, 6), n = 6$

- $8_i, 2_j, 4, 9, 3, 6 \quad j = 2, i = 1, key = 2$ while
- $2, 8_j, 4, 9, 3, 6$
- $2, 8_i, 4_j, 9, 3, 6 \quad j = 3, i = 2, key = 4$ while
- $2, 4, 8, 9, 3, 6$
- $2, 4, 8_i, 9_j, 3, 6 \quad j = 4, i = 3, key = 9$ no while
- $2, 4, 8, 9_i, 3_j, 6 \quad j = 5, i = 4, key = 3$ while
- $2, 3, 4, 8, 9, 6$
- $2, 3, 4, 8, 9_i, 6_j \quad j = 6, i = 5, key = 6$ while
- $2, 3, 4, 6, 8, 9$

```
| <= x | > x | x | ... |
| <= x | x | > x | ... |
```

Porównujemy element ze wszystkim co jest przed nim - wszystko przed j-tym elementem będzie posortowane. Insertion sort nie swapuje par elementów w tablicy, a przenosi tam gdzie jest jego miejsce.

1.4.1 Worst-case analysis - Insertion Sort (A, n)

Odwrotnie posortowana tablica powoduje najwięcej przesunięć. Ponieważ ustaliśmy że liczba operacji w while zależy od j, wtedy:

$$T(n) = \sum_{j=2}^{n} O(j-1) = \sum_{j=1}^{n-1} O(j) = O\left(\sum_{j=1}^{n-1} j\right) = \tag{3}$$

$$= O\left(\frac{1+n-1}{2} \cdot (n-1)\right) = O\left(\frac{(n-1)\cdot(n)}{2}\right) = O\left(\frac{n^2}{2}\right) = O(n^2) \tag{4}$$

c

1.4.2 Average-case analysis - Insertion Sort (A, n)

Policzmy dla uproszczenia, że na wejściu mamy n-elementowe permutacje, z których każda jest jednakowo prawdopodobna $p = \frac{1}{n!}$. Spróbujmy wyznaczyć $\mathbf{E}$, korzystając z inwersji permutacji. Wartość oczekiwana liczby inwersji w losowej permutacji wynosi:

$$\mathbf{E} \sim \frac{n^2}{4} \tag{5}$$

Pominęliśmy stałe wynikające z innych operacji niż porównywanie. W average-case będziemy około połowę szybiciej niż w worst-case.

Pseudokod bez przykładu jest słaby.

1.5 Przykład złożoności

Patrzymy na wiodący czynnik.

$$13n^2 + 91n\log n + 4n + 13^{10} = O(n^2) \tag{6}$$

$$= 13n^2 + O(n\log n) \tag{7}$$

Chcielibyśmy gdzie to konieczne, zapisać *lower order terms.*

Pytanie o dzielenie liczb - istnieją algorytmy, które ze względu na arytmetyczne właściwości liczb sprawiają, że mniejsze liczby mogą dzielić się dłużej niż większe. Podczas tego kursu nie omawiamy złożoności dla takich algorytmów.

2 Lecture II - Merge Sort

2.1 Merge sort $(A, 1, n)$

Niech złożoność T(n) - złożność algorytmu.

Funkcja merge sort

```
O(1)          | if |A[1...n]| == 1 return A[1...n]
              | else
T(floor(n/2)) |     B = MERGE_SORT(A,1,floor(n/2))
T(ceil(n/2))  |     C = MERGE_SORT(A,floor(n/2)+1, n)
O(n)          |     return MERGE(B,C)
```

Funkcja merge

```
MERGE(X[1...k], Y[1...l])
if k = 0 return Y[1...l]
if l = 0 return X[1...k]
if X[1] <= Y[1]
    return X[1] o MERGE(X[2...k], Y[1...l])
else
    return Y[1] o MERGE(X[1...k], Y[2...l])
```

```
MERGE(A,B)
2 1 ---> [1] + MERGE(A,B (bez 1))
7 9
13 10
19 11
20 14

2 9  ---> [1,2] + MERGE(A (bez 2),B)
7 10
13 11
19 14
20 .

... ---> [1,2,7,9,10,11,13,14]
19 .
20 .

... ---> [1,2,7,9,10,11,13,14,19,20]

[10], [2], [5], [3], [7], [13], [1], [6]
[2, 10], [3,5], [7,13], [1,6]
[2,3,5,10], [1,6,7,13]
[1,2,3,5,6,7,10,13]
```

Złożoność obliczeniowa merge-a wynosi $O(k+l)$ - w najgorszym przypadku bierzemy najpierw z jednej strony, potem z drugiej i na zmianę.

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n) \tag{8}$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n) \tag{9}$$

Rozpiszmy tzw drzewo rekursji:

```
        cn                                | cn
       /  \                               |
  cn/2      cn/2                          | cn
  /   \     /   \                         |
cn/4 cn/4 cn/4 cn/4                       | cn
                                          |
      [...]                               | ...
                                          |
O(1) ... O(1) ... O(1) # liści mamy n     | cn
```

Musimy dodać wszystkie koszty, które pojawiły się w drzewie. Dodajmy piętra, a następnie zsumumjmy. Żeby znać wysokość drzewa interesuje nas dla jakiego h zajdzie $\frac{n}{2^h} = 1$

$$\frac{n}{2^h} = 1 \implies 2^h = n \implies h = \log_2 n \tag{10}$$

Zatem złożność:

$$\sum_{i=1}^{\log n} cn = cn \log n \sim O(n \log n) \tag{11}$$

3 Lecture III - Narzędzia do analizy algorytmów

Dzisiejszy wykład prowadzi GODfryd

3.1 Notacja asymptotyczna

- Big-O (O-duże) $f : \mathbb{N} \to \mathbb{R}$
- Big-Ω (Ω-duże) $f : \mathbb{N} \to \mathbb{R}$
- Big-Θ (Θ-duże) $f : \mathbb{N} \to \mathbb{R}$
- Small-o (o-małe) $f : \mathbb{N} \to \mathbb{R}$

3.2 Notacja Big-O

Definition. Notacja Big-O. Funkcja $f(n) \in O(g(n))$, gdy:

$$f(n) = O(g(n)) \equiv (\exists c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geqslant n_0)\,(|f(n)| \leqslant c \cdot |g(n)|)$$

Przykład: $2n^2 = O(n^3)$, dla $n_0 = 2, c = 1$ definicja jest spełniona.

Pomijamy tutaj stałe - interesuje nas rząd wielkości

$$O(g(n)) = \left\{f \in \mathbb{N}^{\mathbb{R}} : \text{f spełnia definicję}\right\}$$

$O(g(n))$ jest klasą funkcji, ale jako informatycy możemy zapisywać $f = O(g)$, zamiast $f \in O(g)$. Notacja nie ma symetrii, to znaczy $f = O(g) \nrightarrow g = O(f)$

Fact. Definicja Big-O za pomocą granicy. Możemy zapisać alternatywnie:

$$f(n) = O(g(n)) \equiv \limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \leqslant \infty$$

Uwaga. Jeśli $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty$ (istnieje), to:

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

Przykłady: $\begin{cases} f(n) = n^2 \\ g(n) = (-1)^n n^2 \end{cases}$
Granica nie istnieje, ale $\limsup = 1$
$\left\{\frac{f(n)}{g(n)} = \begin{cases} 1, & 2 \mid n \\ \frac{1}{n}, & 2 \nmid n \end{cases}\right.$
Granica nie istnieje.

Fact. Dokładność zapisu Big-O. Pomijamy składniki niższego rzędu jako mniej istotne, ale podkreślamy że istnieją:

$$f(n) = n^3 + O(n^2) \equiv \left(\exists h(n) = O(n^2)\right)\left(f(n) = n^3 + h(n)\right) \tag{12}$$

Rozważmy następnie stwierdzenie:

$$n^2 + O(n) = O(n^2) \equiv (\forall f(n) = O(n))\left(\exists h(n) = O(n^2)\right)\left(n^2 + f(n) = h(n)\right) \tag{13}$$

Rozumiemy to następująco - dodając dowolną funkcję z klasy funkcji liniowych do n^2 otrzymamy funkcję z klasy funkcji kwadratowych.

3.3 Notacja Big-Ω

Definition. Notacja Big-Ω. Funkcja $f(n) \in \Omega(g(n))$, gdy:

$$f(n) = \Omega(g(n)) \equiv (\exists c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geqslant n_0)\,(|f(n)| \geqslant c \cdot |g(n)|) \tag{14}$$

biorąc $c' = \frac{1}{c} > 0$ mamy: $(|g(n)| \leqslant c' \cdot |f(n)|)$, czyli $g(n) = O(f(n))$.
Przykład:

$$2n^2 = O(n^3) \tag{15}$$
$$n^3 = \Omega(2n^2) \tag{16}$$
$$n = \Omega(\log n) \tag{17}$$

Każda funkcja jest Omega od siebie samej.

3.4 Notacja Big-Θ

Definition. Notacja Big-Θ. Funkcja $f(n) \in \Theta(g(n))$, gdy:

$$f(n) = \Theta(g(n)) \equiv (\exists c_1, c_2 > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geqslant n_0)\,(c_1 \cdot |g(n)| \leqslant |f(n)| \leqslant c_2 \cdot |g(n)|) \tag{18}$$

Przykład:

$$n^2 = \Theta(2n^2) \tag{19}$$
$$n^3 = \Theta(n^3) \tag{20}$$
$$n^4 + 3n^2 + \log n = \Theta(n^4) \tag{21}$$

Fact. Dokładność zapisu Theta.

$$f(n) = \Theta(g(n)) \equiv f(n) = O(g(n)) \wedge f(n) = \Omega(g(n)) \tag{22}$$
$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n)) \tag{23}$$

Rozważmy przypadek patologiczny

$$f(n) = n^{1+\sin \frac{\pi \cdot n}{2}} \quad g(n) = n \tag{24}$$
$$f \neq O(g), g \neq O(f) \tag{25}$$

3.5 Notacja small-o

Definition. Notacja small-o. Funkcja $f(n) \in o(g(n))$, gdy:

$$f(n) = o(g(n)) \equiv (\forall c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geqslant n_0)\,(|f(n)| < c \cdot |g(n)|) \tag{26}$$

Równoważnie:

$$f(n) = o(g(n)) \equiv \lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = 0 \tag{27}$$

Przykład:

$$n = o(n^2) \tag{28}$$
$$n^2 = o(n^3) \tag{29}$$
$$n^3 = o(2^n) \tag{30}$$

3.6 Notacja small-ω

Definition. Notacja small-ω. Funkcja $f(n) \in \omega(g(n))$, gdy:

$$f(n) = \omega(g(n)) \equiv (\forall c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geqslant n_0)\,(|f(n)| > c \cdot |g(n)|) \tag{31}$$

Równoważnie:

$$f(n) = \omega(g(n)) \equiv \lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = \infty \tag{32}$$

Przykład:

$$3.14n^2 + n = O(n^3) = \omega(n) \tag{33}$$

3.7 Metody rozwiązywania rekurencji

- Metoda podstawienia (indukcji) - Cormen
- Metoda drzewa rekursji
- Metoda master theorem

3.8 Rozwiązywanie rekurencji

1. Zgadnij odpowiedź (wiodący składnik)
2. Sprawdź przez indukcję, czy dobrze zgadliśmy
3. Wylicz stałe

Information. Historyjka. Dwóch przyjaciół zgubiło się podczas podróży balonem.

- "Gdzie jesteśmy?"
- "W balonie."

Osoba, którą spotkali, była matematykiem.
Odpowiedź była precyzyjna, dokładna i całkowicie bezużyteczna.

3.9 Metoda podstawiania - Metoda dowodu indukcyjnego

Przykład 1. Rozwiążmy równanie rekurencyjne:

$$T(n) = 4T\left(\frac{n}{2}\right) + n \quad T(1) = \Theta(1) \tag{34}$$

Załóżmy, że $T(n) = O(n^3)$ - pokazać, że $T(n) \leqslant c \cdot n^3$. dla dużych n.

1. Krok początkowy $T(1) = \Theta(1) \leqslant c \cdot 1^3 = c$ ok.
2. Założmy, że $\forall_{k<n} T(k) \leqslant c \cdot k^3$ (zał. indukcyjne, nie $\Theta(k^3)$ - chcemy konkretną stałą c)
3. $T(n) = 4T\left(\frac{n}{2}\right) + n \leqslant 4c\left(\frac{n}{2}\right)^3 + n = \frac{1}{2}cn^3 + n = cn^3 - \frac{1}{2}cn^3 + n \leqslant cn^3$.
4. Wystarczy wskazać c, takie że $\frac{1}{2}cn^3 - n \geqslant 0$, np $c \geqslant 2$
5. Pokazaliśmy, że $T(n) = O(n^3)$

Załóżmy, że $T(n) = O(n^2)$ - pokazać, że $T(n) \leqslant c \cdot n^2$. dla dużych n.

1. Krok początkowy $T(1) = \Theta(1) \leqslant c \cdot 1^2 = c$ ok.
2. Założmy, że $\forall_{k<n} T(k) \leqslant c \cdot k^2$ (zał. indukcyjne)
3. $T(n) = 4T\left(\frac{n}{2}\right) + n \leqslant 4c\left(\frac{n}{2}\right)^2 + n = cn^2 + n = cn^2 - cn^2 + n \leqslant cn^2$.
4. Tego się nie da pokazać - nie jest prawdą, że $T(n) = O(n^2)$

Wzmocnijmy zatem założenie indukcyjne:

1. $T(n) \leqslant c_1 n^2 - c_2 n$ (zał. indukcyjne)
2. $T(n) = 4T\left(\frac{n}{2}\right) + n \leqslant 4(c_1 \frac{n}{2}^2 - c_2 \frac{n}{2}) + n$
3. $= c_1 n^2 - 2c_2 n + n = c_1 n^2 - (2c_2 - 1)n \leqslant$
4. $\leqslant c_1 n^2 - c_2 n$
5. Weźmy $c_1 = 1, c_2 = 2$, wtedy $T(n) \leqslant n^2 - 2n = O(n^2)$

Przykład 2. Weźmy paskudną rekursję $T(n) = 2T(\sqrt{n}) + \log n$.
Załóżmy, że n jest potęgą 2 oraz oznaczny $n = 2^m, m = \log_2 n$.

$$T(2^m) = 2T((2^m)^{\frac{1}{2}}) + m \tag{35}$$

Oznaczmy $T(2^m) = S(m)$. Wtedy:

$$S(m) = 2S\left(\frac{m}{2}\right) + m \tag{36}$$

(dobrze znana rekurencja - $S(n) = O(m \log m)$) - patrz Lecture 2. Przejdźmy z powrotem na T, n:

$$T(2^m) = S(m)T(2^m) = O(m \log m)T(n) = O(\log n \log \log n) \tag{37}$$

Formalnie pokazaliśmy to tylko dla potęg 2 - musielibyśmy jeszcze indukcyjnie to udowodnić.

Kiedy podłogi i sufity mają znaczenie?