Politechnika Wrocławska

Struktury danych i złożoność obliczeniowa Wykład 3.

Prof. dr hab. inż. Jan Magott

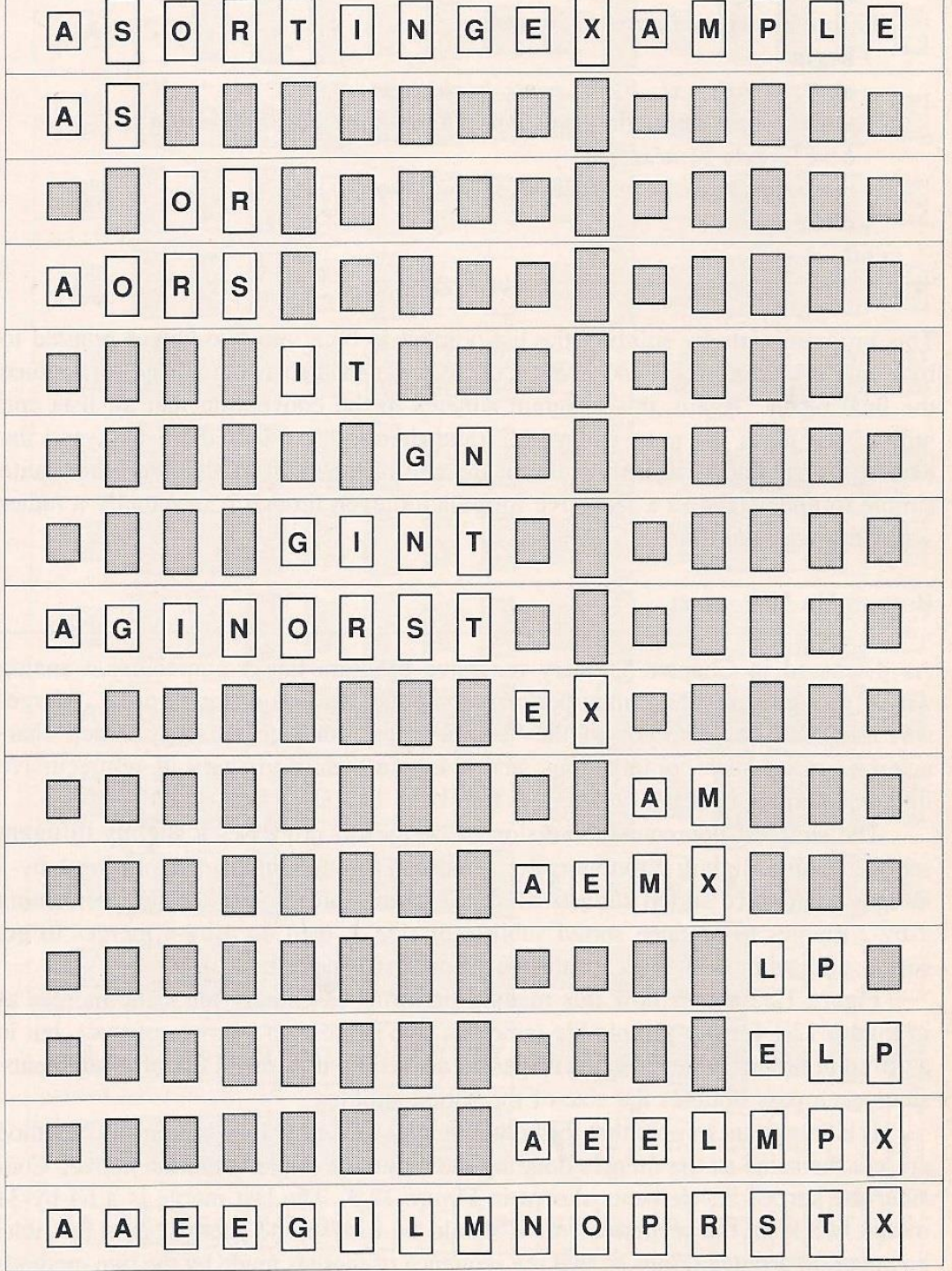
Metody konstrukcji algorytmów:

- Siłowa (ang. brute force),
- **„Dziel i zwyciężaj”** (ang. divide-and-conquer),
- **Zachłanna** (ang. greedy),
- **Transformacyjna konstrukcja algorytmu,**
- Programowanie dynamiczne,
- Przeszukiwanie z powrotami i metoda podziałów i ograniczeń,
- Algorytmów genetycznych.

Metoda „dziel i zwyciężaj”

Metoda „dziel i zwyciężaj”

1. Dziel: Podziel problem na podproblemy,
2. Zwyciężaj: Podproblemy rozwiąż rekurencyjnie lub bezpośrednio, jeśli są odpowiednio małe,
3. Scal: Połącz rozwiązania podproblemów w celu wyznaczenia rozwiązania problemu.



n – liczba elementów, $\lceil n/2 \rceil$ ($\lfloor n/2 \rfloor$) – liczba elementów lewej (prawej) części

Źródło:
[R. Sedgewick, Algorithms, Second Edition, Addison-Wesley, 1988.]

Metoda „dziel i zwyciężaj”

n – liczba elementów,

Uzasadnienie formuł:

$\lceil n/2 \rceil$ ($\lfloor n/2 \rfloor$) - liczba elementów lewej (prawej) części,

n parzyste czyli $n = 2k,\ k \in N_+,$

$\lceil 2k/2 \rceil = k = \lfloor 2k/2 \rfloor$

n nieparzyste czyli $n = 2k + 1,\ k \in N_+,$

$\lceil \frac{2k+1}{2} \rceil = k + 1, \qquad \lfloor \frac{2k+1}{2} \rfloor = k$

```
MERGE(A, p, q, r)
 1  n1 ← q − p + 1
 2  n2 ← r − q
 3  utwórz tablice L[1..n1 + 1] i R[1..n2 + 1]
 4  for i ← 1 to n1
 5    do L[i] ← A[p + i − 1]
 6  for j ← 1 to n2
 7    do R[j] ← A[q + j]
 8  L[n1 + 1] ← ∞
 9  R[n2 + 1] ← ∞
10  i ← 1
11  j ← 1
12  for k ← p to r
13    do if L[i] ≤ R[j]
14          then A[k] ← L[i]
15               i ← i + 1
16          else A[k] ← R[j]
17               j ← j + 1
```

Tablice $A[p..q]$ i $A[q+1..r]$ są posortowane,
∞ jest wartownikiem większym od każdego z elementów tablicy A,
MERGE scala powyższe tablice w posortowaną tablicę $A[p..r]$.

Złożoność czasowa $\Theta(n)$, gdzie $n = r - p + 1$.

Źródło: [CLRS, Wprowadzenie do algorytmów]

Metoda „dziel i zwyciężaj”

```
i ← 1
j ← 1
for k ← p to r
  do if L[i] ≤ R[j]
        then A[k] ← L[i]
             i ← i + 1
        else A[k] ← R[j]
             j ← j + 1
```

Niezmiennik pętli:
Przed wykonaniem pętli dla k, tablica $A[p..k-1]$ zawiera $k-p$ najmniejszych elementów tablic $L[1..n_1+1]$ i $R[1..n_2+1]$ będących posortowanymi.
$L[i]$, $R[j]$ są najmniejszymi elementami tablic L, R, które jeszcze nie zostały skopiowane do tablicy A.

Metoda „dziel i zwyciężaj”

```
MERGE-SORT(A, p, r)
1   if p < r
2       then q ← ⌊(p + r)/2⌋
3            MERGE-SORT(A, p, q)
4            MERGE-SORT(A, q + 1, r)
5            MERGE(A, p, q, r)
```

Dziel

Zwyciężaj

Scal

Źródło: [CLRS, Wprowadzenie do algorytmów]

Metoda „dziel i zwyciężaj”

$T(i)$

```
MERGE-SORT(A, p, r)
1   if p < r
2       then q ← ⌊(p + r)/2⌋
3            MERGE-SORT(A, p, q)
4            MERGE-SORT(A, q + 1, r)
5            MERGE(A, p, q, r)
```

Dziel	$\Theta(1)$
Zwyciężaj	$2 \cdot T(i/2)$
Scal	$\Theta(i)$

$T(i)$ – czas sortowania i elementów, gdy $1 < i$

$$T(i) = 2 \cdot T(i/2) + \Theta(i)$$

Metoda „dziel i zwyciężaj”

$T(1)$

```
MERGE-SORT(A, p, r)
1   if p < r
2       then q ← ⌊(p + r)/2⌋
3            MERGE-SORT(A, p, q)
4            MERGE-SORT(A, q + 1, r)
5            MERGE(A, p, q, r)
```

Dziel $\Theta(1)$

$T(1)$ – czas sortowania i elementów, gdy $1 = i$

$T(1) = \Theta(1)$

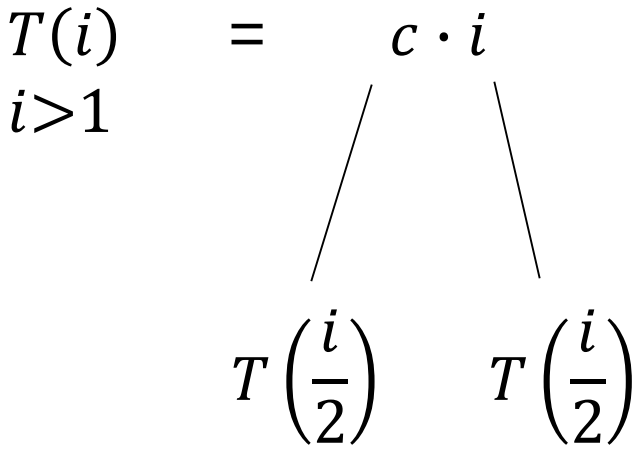
Metoda „dziel i zwyciężaj”

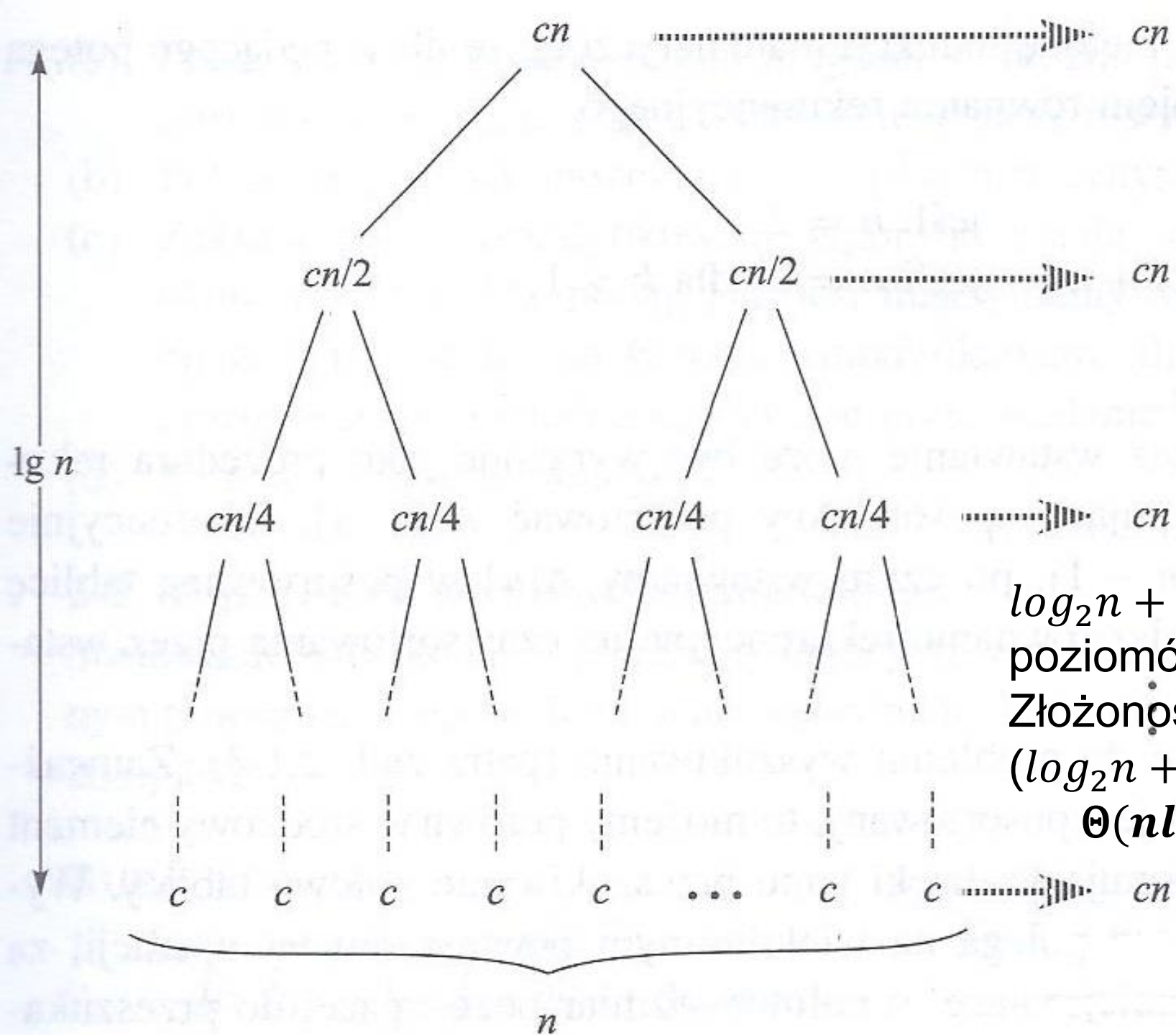
$$T(i) = \begin{cases} \Theta(1) & \text{jeśli } i = 1 \\ 2 \cdot T\left(\frac{i}{2}\right) + \Theta(i) & \text{jeśli } i > 1 \end{cases}$$

$$T(i) = \begin{cases} c & \text{jeśli } i = 1 \\ 2 \cdot T\left(\frac{i}{2}\right) + c \cdot i & \text{jeśli } i > 1 \end{cases}$$

Metoda „dziel i zwyciężaj”

$$T(i) = \begin{cases} c & \text{jeśli } i = 1 \\ 2 \cdot T\left(\frac{i}{2}\right) + c \cdot i & \text{jeśli } i > 1 \end{cases}$$





$log_2 n + 1$ poziomów,
Złożoność:
$(log_2 n + 1) \cdot cn =$
$\boldsymbol{\Theta(n log_2 n)}$

Metoda zachłanna

Dyskretny problem plecakowy - wersja decyzyjna

Dane:

Skończony zbiór elementów $A = \{a_1, a_2, \ldots, a_n\}$.

Rozmiar $s(a_i) > 0$ i waga (wartość) $w(a_i) > 0$ elementu a_i .

Pojemność plecaka $b > 0$ i stała $y > 0$.

Zadanie:

Czy istnieje podzbiór $A' \subset A$ taki, że:

$$\sum_{a_i \in A'} s(a_i) \leq b$$

$$\sum_{a_i \in A'} w(a_i) \geq y \quad ?$$

Metoda zachłanna

Dyskretny problem plecakowy – wersja optymalizacyjna

Dane:

Skończony zbiór elementów $A = \{a_1, a_2, \dots, a_n\}$.

Rozmiar $s(a_i) > 0$ i waga (wartość) $w(a_i) > 0$ elementu a_i .

Pojemność plecaka $b > 0$ i stała $y > 0$.

Zadanie:

Wyznacz podzbiór $A' \subset A$ taki, że:

$$\sum_{a_i \in A'} s(a_i) \leq b$$

$$\min \sum_{a_i \in A'} w(a_i)$$

Metoda zachłanna

Algorytm zachłanny dla wersji optymalizacyjnej

1. Dla każdego elementu $i \in \{\overline{1,n}\}$ oblicz $p_i = w(a_i)/s(a_i)$.
   $A' = \emptyset$ i $S = 0$.
2. Wyznacz sekwencję $\sigma =< \sigma(1), \ldots, \sigma(n) >$ taką, że
   $p_{\sigma(j)} \geq p_{\sigma(j+1)}$ dla $j \in \{\overline{1,n-1}\}$.
3. Dla $j \in \{\overline{1,n}\}$ wykonuj: Jeżeli $S + s(a_{\sigma(j)}) \leq b$, to
   $A' := A' \cup \{a_{\sigma(j)}\}$ i $S := S + s(a_{\sigma(j)})$.
4. Stop

Metoda zachłanna

i	$s(a_i)$	$w(a_i)$
1	1	9
2	2	8
3	6	7
4	7	10
5	1	8
6	4	7

b=12

Metoda zachłanna

i	$s(a_i)$	$w(a_i)$	$w(a_i)/s(a_i)$
1	1	9	9
2	2	8	4
3	6	7	1,17
4	7	10	1,43
5	1	8	8
6	4	7	1,75

b=12

$$\sigma =< 1, 5, 2, 6, 4, 3 >$$

Metoda zachłanna

$$\boldsymbol{S} = < \mathbf{1}, \mathbf{5}, \mathbf{2}, \mathbf{6}, \mathbf{4}, \mathbf{3} >$$

S$\leftarrow 0$

Weź a_1 i sprawdź czy $S + s(a_1) \leq b$ czyli $0 + 1 \leq 12$.
Zatem $A' \leftarrow \{a_1\}$ i $S \leftarrow 1$.

Weź a_5 i sprawdź czy $S + s(a_5) \leq b$ czyli $1 + 1 \leq 12$.
Zatem $A' \leftarrow \{a_1, a_5\}$ i $S \leftarrow 2$.

Weź a_2 i sprawdź czy $S + s(a_2) \leq b$ czyli $2 + 2 \leq 12$.
Zatem $A' \leftarrow \{a_1, a_5, a_2\}$ i $S \leftarrow 4$.

Weź a_6 i sprawdź czy $S + s(a_6) \leq b$ czyli $4 + 4 \leq 12$.
Zatem $A' \leftarrow \{a_1, a_5, a_2, a_6\}$ i $S \leftarrow 8$.

Weź a_4 i sprawdź czy $S + s(a_4) \leq b$ czyli $8 + 7 \leq 12$.
Zatem $A' \leftarrow \{a_1, a_5, a_2, a_6\}$ i $S \leftarrow 8$.

Weź a_3 i sprawdź czy $S + s(a_3) \leq b$ czyli $8 + 6 \leq 12$.
Zatem $A' \leftarrow \{a_1, a_5, a_2, a_6\}$ i $S \leftarrow 8$.

Metoda zachłanna

Jaka jest złożoność obliczeniowa algorytmu?

1. Dla każdego elementu $i \in \{\overline{1,n}\}$ oblicz $p_i = w(a_i)/s(a_i)$.
   $A' \leftarrow \emptyset$ i $S \leftarrow 0$.
2. Wyznacz sekwencję $\sigma =< \sigma(1), \ldots, \sigma(n) >$ taką, że
   $p_{\sigma(j)} \geq p_{\sigma(j+1)}$ dla $j \in \{\overline{1,n-1}\}$.
3. Dla $j \in \{\overline{1,n}\}$ wykonuj: Jeżeli $S + s(a_{\sigma(j)}) \leq b$, to
   $A' \leftarrow A' \cup \{a_{\sigma(j)}\}$ i $S \leftarrow S + s(a_{\sigma(j)})$.
4. Stop

Metoda zachłanna

Jaka jest złożoność obliczeniowa algorytmu?

1. Dla każdego elementu $i \in \{\overline{1,n}\}$ oblicz $p_i = w(a_i)/s(a_i)$.
   $A' \leftarrow \emptyset$ i $S \leftarrow 0$. $\boldsymbol{O(n)}$
2. Wyznacz sekwencję $\sigma =< \sigma(1), \dots, \sigma(n) >$ taką, że $p_{\sigma(j)} \geq p_{\sigma(j+1)}$ dla $j \in \{\overline{1,n-1}\}$.
3. Dla $j \in \{\overline{1,n}\}$ wykonuj: Jeżeli $S + s(a_{\sigma(j)}) \leq b$, to
   $A' \leftarrow A' \cup \{a_{\sigma(j)}\}$ i $S \leftarrow S + s(a_{\sigma(j)})$.
4. Stop

Metoda zachłanna

Jaka jest złożoność obliczeniowa algorytmu?

1. Dla każdego elementu $i \in \{\overline{1,n}\}$ oblicz $p_i = w(a_i)/s(a_i)$.
   $A' \leftarrow \emptyset$ i $S \leftarrow 0$. $\boldsymbol{O(n)}$
2. Wyznacz sekwencję $\sigma =< \sigma(1), \dots, \sigma(n) >$ taką, że $p_{\sigma(j)} \geq p_{\sigma(j+1)}$ dla $j \in \{\overline{1,n-1}\}$. $\boldsymbol{O(n \log n)}$
3. Dla $j \in \{\overline{1,n}\}$ wykonuj: Jeżeli $S + s(a_{\sigma(j)}) \leq b$, to
   $A' \leftarrow A' \cup \{a_{\sigma(j)}\}$ i $S \leftarrow S + s(a_{\sigma(j)})$.
4. Stop

Metoda zachłanna

Jaka jest złożoność obliczeniowa algorytmu?

1. Dla każdego elementu $i \in \{\overline{1,n}\}$ oblicz $p_i = w(a_i)/s(a_i)$.
   $A' \leftarrow \emptyset$ i $S \leftarrow 0$. $\boldsymbol{O(n)}$
2. Wyznacz sekwencję $\sigma =< \sigma(1), \dots, \sigma(n) >$ taką, że
   $p_{\sigma(j)} \geq p_{\sigma(j+1)}$ dla $j \in \{\overline{1,n-1}\}$. $\boldsymbol{O(n \log n)}$
3. Dla $j \in \{\overline{1,n}\}$ wykonuj: Jeżeli $S + s(a_{\sigma(j)}) \leq b$, to
   $A' \leftarrow A' \cup \{a_{\sigma(j)}\}$ i $S \leftarrow S + s(a_{\sigma(j)})$. $\boldsymbol{O(n)}$
4. Stop

Metoda zachłanna

Złożoność obliczeniowa algorytmu

$$\boldsymbol{O}(\boldsymbol{n}) + \boldsymbol{O}(\boldsymbol{n}\log\boldsymbol{n}) + \boldsymbol{O}(\boldsymbol{n}) = \boldsymbol{O}(\boldsymbol{n}\log\boldsymbol{n})$$

Czy rozwiązanie:

$$A' = \{a_1, a_5, a_2, a_6\}$$

zajętość plecaka:

$$S = 8$$

sumaryczna wartość zapakowanych elementów:

$$\sum_{a_i \in X} w(a_i) = 9 + 8 + 8 + 7 = 32$$

jest optymalnym ?

Metoda zachłanna

i	$s(a_i)$	$w(a_i)$
1	1	9
2	2	8
3	6	7
4	7	10
5	1	8
6	4	7

b=12

Metoda zachłanna

Lepszym rozwiązaniem jest:

$$A' = \{a_1, a_5, a_2, a_4\}$$

zajętość plecaka:

$$S = 11$$

sumaryczna wartość zapakowanych elementów:

$$\sum_{a_i \in X} w(a_i) = 9 + 8 + 8 + 10 = 35$$

Metoda zachłanna

Ciągły problem plecakowy – wersja decyzyjna

Dane:

Skończony zbiór produktów $A = \{a_1, a_2, \dots, a_n\}$.

Rozmiar $s(a_i) \epsilon N$ i wartość $w(a_i) \epsilon N$ elementu a_i.

Pojemność plecaka $b > 0$ i stała $y > 0$.

Zadanie:

Czy istnieje taki zbiór wartości $x_1, x_2, \dots, x_n \epsilon [0,1]$, że:

$$\sum_{i \in \{1,n\}} s(a_i) \cdot x_i \leq b$$

$$\sum_{i \in \{1,n\}} w(a_i) \cdot x_i \geq y \quad ?$$

Czy algorytm zachłanny oparty na $p_i = w(a_i)/s(a_i)$ da rozwiązanie?

Metoda zachłanna

Przykłady interpretacji

Dyskretny problem plecakowy

- Ładowanie sztabek złota do plecaka,
- Ładowanie bloków betonowych na platformę.

Ciągły problem plecakowy

- Ładowanie złotego piasku,
- Ładowanie substancji sypkich na platformę do przewozu materiałów różnych gatunków.

Transformacyjna konstrukcja algorytmu

Przykład 1.

Wyznaczanie najmniejszej wspólnej wielokrotnej (NWW) liczb $m, n \epsilon N$ poprzez **redukcję problemu:**

$$NWW(m,n) = (m \cdot n)/NWP(m,n)$$

Transformacyjna konstrukcja algorytmu

Przykład 2.

Startujemy od mało efektywnego (**naiwnego**) algorytmu i **konstruujemy algorytm efektywniejszy**.

Problem: Wyznaczenie pary najbliższych spośród n elementów tablicy.

- Metoda siłowa o złożoności $O(n^2)$,
- Transformacja do postaci:
  1. Wstępne sortowanie tablicy: $O(n\log n)$,
  2. Skanowanie tablicy z wyznaczaniem najmniejszej różnicy między sąsiednimi elementami: $O(n)$,

  Sumaryczny koszt: $O(n\log n)$.

Transformacyjna konstrukcja algorytmu

Przykłady 3.

Startujemy od mało efektywnego (**naiwnego**) algorytmu i **konstruujemy algorytm efektywniejszy**.

Problem:

Dane: Posortowane niemalejąco tablice A, B o n elementach $A[i], B[j] \in N$, $i, j \epsilon \{\overline{1, n}\}$, liczba $x \epsilon N$.

Pytanie: Czy istnieją takie $A[i], B[j]$, $i, j \epsilon \{\overline{1, n}\}$, że:

$A[i] + B[j] = x$?

Transformacyjna konstrukcja algorytmu

Problem:

Dane: Posortowane niemalejąco tablice A, B o n elementach $A[i], B[j] \in N$, $i, j \epsilon \{\overline{1, n}\}$, liczba $x \epsilon N$.

Pytanie: Czy istnieją takie $A[i], B[j]$, $i, j \epsilon \{\overline{1, n}\}$, że:

$A[i] + B[j] = x$?

Metoda siłowa

BOOLEAN EQUAL-SUM-BF(A, B, n)

for $i \leftarrow 1$ **to** n

for $j \leftarrow 1$ **to** n $O(n^2)$

do if $(A[i] + B[j] = x)$ **then return true**

return false

Transformacyjna konstrukcja algorytmu

Dane: Posortowane niemalejąco tablice A, B o n elementach $A[i], B[j] \in N$, $i, j \epsilon \{\overline{1, n}\}$, liczba $x \epsilon N$.

Pytanie: Czy istnieją takie $A[i], B[j]$, $i, j \epsilon \{\overline{1, n}\}$, że: $A[i] + B[j] = x$?

Efekt transformacji

BOOLEAN EQUAL-SUM-TR(A, B, n)

$i \leftarrow 1$

$j \leftarrow n$

while$(i \leq n$ **and** $j>0)$

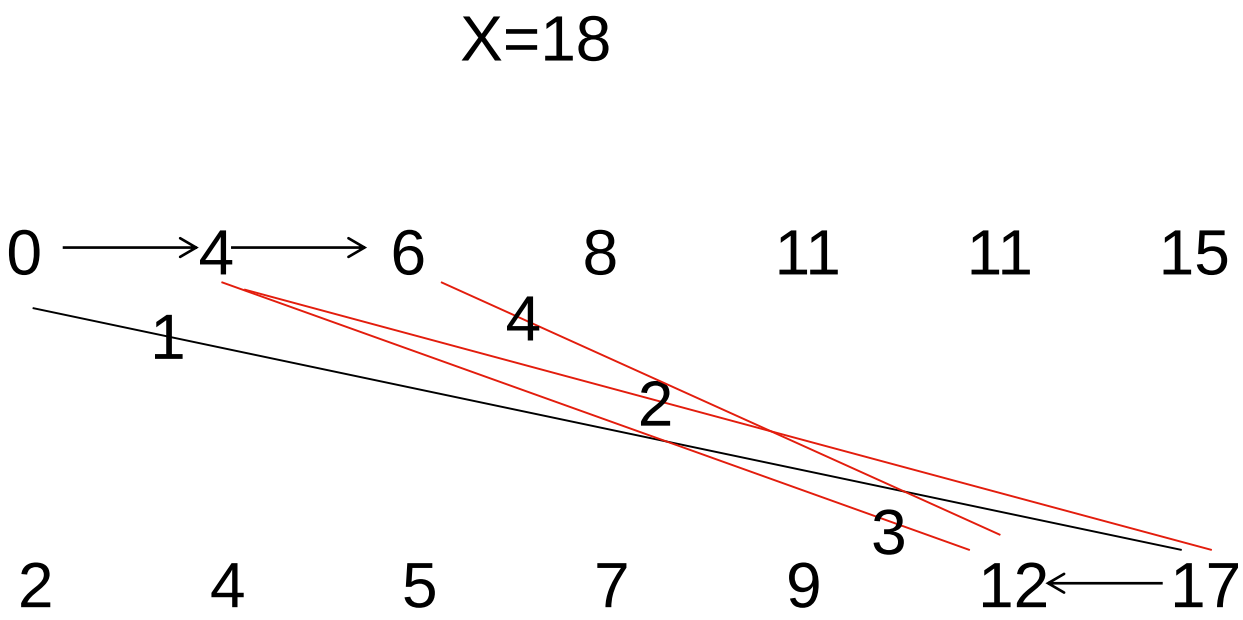
do if $(A[i] + B[j] = x)$ **then return true**

else if $(A[i] + B[j] < x)$ **then** $i \leftarrow i + 1$

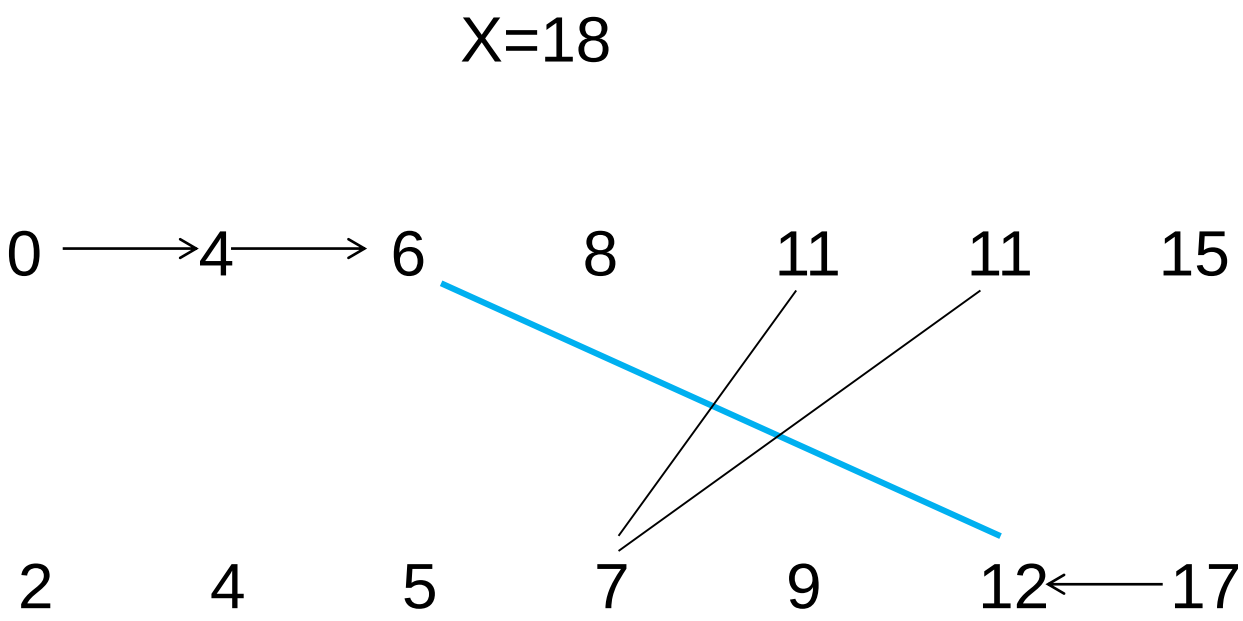
else $j \leftarrow j - 1$

return false

Transformacyjna konstrukcja algorytmu



Transformacyjna konstrukcja algorytmu

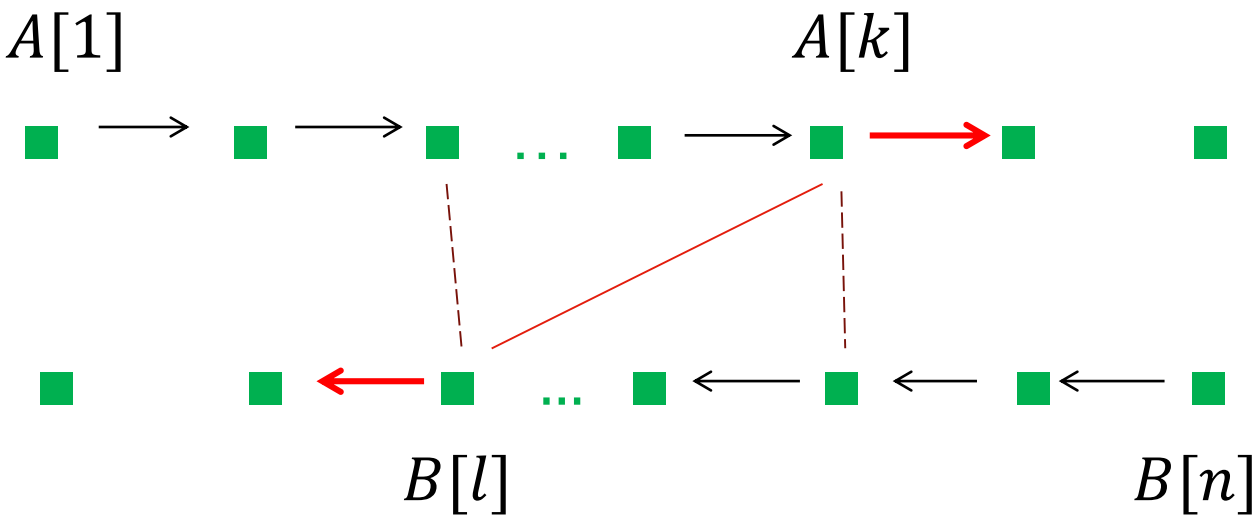


Wyróżniona para liczb $A[k], B[l]$ spełniająca wymagania:

$$k = min\{i: (\exists j \epsilon \{\overline{1,n}\})(A[i] + B[j] = x)\},$$

$$l = max\{j: (\exists i \epsilon \{\overline{1,n}\})(A[i] + B[j] = x)\}.$$

Transformacyjna konstrukcja algorytmu



Groźba pominięcia pary $A[i] + B[j] = x$

Pominięcie wyróżnionej pary gdy:

przy warunku $i < k \wedge j = l$ nastąpi $j \leftarrow l - 1$

lub

przy warunku $i = k \wedge l < j$ nastąpi $i \leftarrow k + 1$.

?

Transformacyjna konstrukcja algorytmu

Niezmiennik: Nie pominięto wyróżnionej pary tzn. $i \leq k \wedge l \leq j$.

Inicjowanie: Przed pierwszym wykonaniem pętli dla $i = 1 \wedge j = n$, wyróżniona para nie została pominięta.

Niezmienniczość: Nie pominięto wyróżnionej pary przed $(i + n - j)$ - tym wykonaniem pętli, to:

Jeśli $A[i] + B[j] = x$, tzn. $i = k, j = l$ i nastąpi wyjście z pętli z wartością **true**

albo

Jeśli $A[i] + B[j] \neq x$, to zmiana wartości zmiennych i, j nie spowoduje pominięcia wyróżnionej pary.

Kończenie: Dla $i = k, j = l$ następuje wyjście z pętli z wartością **true.**

Transformacyjna konstrukcja algorytmu

BOOLEAN EQUAL-SUM-TR(A, B, n)

$i \leftarrow 1$

$j \leftarrow n$

while($i \leq n$ **and** j>0)

do **if** ($A[i] + B[j] = x$) **then return true**

else **if** ($A[i] + B[j] < x$) **then** $i \leftarrow i + 1$

else $j \leftarrow j - 1$

return false

X=18

A:	16	16	16 …	16	17	
B:	1	3	3 …	3	3	

$O(n)$