

# Algorytmy Równoległe i Rozproszone

## Część VI - PRAM - Algorytmy numeryczne

Łukasz Kuszner  
pokój 209, WETI

<http://www.sphere.pl/~kuszner/>  
kuszner@sphere.pl

Oficjalna strona wykładu  
<http://www.sphere.pl/~kuszner/ARiR/>

2005/06

Postorder (poprawka)

Wyznacznik macierzy

Transformata Fourier'a

Różniczkowanie i...

Problemy P-zupełne

Strona główna

Strona tytułowa



Strona 1 z 22

Powrót

Full Screen

Zamknij

Koniec

# 1. Postorder (poprawka)

**Algorytm 1:** Kolejność Postorder

```
1: for każda krawędź  $(u, v)$  in parallel do  
2:   if  $u = p(v)$  then  
3:     krawędź ma wagę  $w(u, v) = 0$   
4:   else  
5:     krawędź ma wagę  $w(u, v) = 1$   
6:   end if  
7: end for  
8: Znajdź sumę wag na krawędziach stosując „pointer jumping”  
  
9: for każdy wierzchołek  $(v)$  in parallel do  
10:   $post(v) =$  znaleziona suma wag  $w$  na krawędzi  $(v, p(v))$ .  
11: end for  
12:  $post(r) = n$ 
```

Model: CREW PRAM, czas  $O(\log n)$  i  $O(n)$  procesorów.

Postorder (poprawka)

Wyznacznik macierzy

Transformata Fourier'a

Różniczkowanie i ...

Problemy P-zupełne

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 2 z 22

Powrót

Full Screen

Zamknij

Koniec

## Przykład

$$w(4, 2) = 0,$$

$$w(2, 1) = 0,$$

$$w(1, 2) = 1,$$

$$w(5, 7) = 0,$$

$$w(7, 5) = 1,$$

$$w(5, 8) = 0,$$

$$w(8, 10) = 0,$$

$$w(10, 11) = 0,$$

$$w(11, 10) = 1,$$

$$w(10, 8) = 1,$$

$$w(8, 5) = 1,$$

$$w(5, 9) = 0,$$

$$w(9, 5) = 1,$$

$$w(5, 4) = 1,$$

$$w(2, 4) = 1,$$

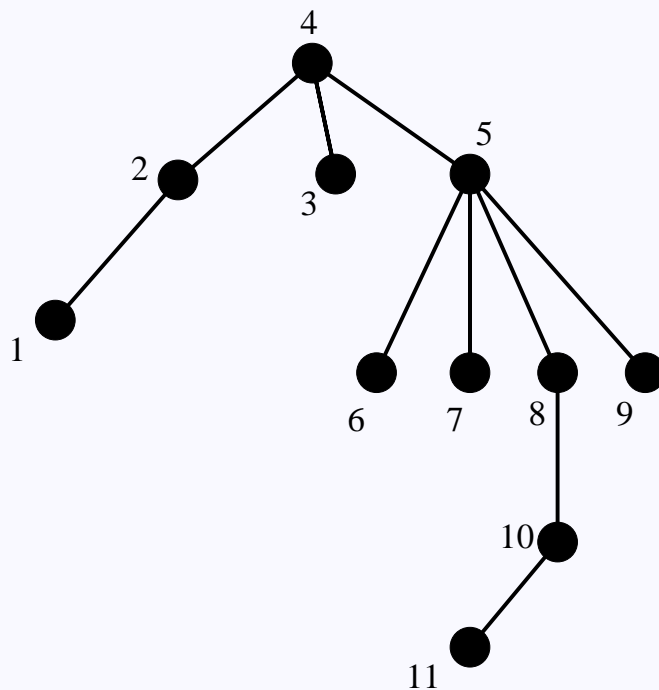
$$w(4, 3) = 0,$$

$$w(3, 4) = 1,$$

$$w(4, 5) = 0,$$

$$w(5, 6) = 0,$$

$$w(6, 5) = 1,$$



Postorder (poprawka)

Wyznacznik macierzy

Transformata Fourier'a

Różniczkowanie i ...

Problemy P-zupełne

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 3 z 22

Powrót

Full Screen

Zamknij

Koniec

## 2. Wyznacznik macierzy

**Algorytm 2:** Wyznacznik macierzy

```

1: for  $k = 1$  to  $n - 1$  do
2:   for  $i = k + 1$  to  $n$  in parallel do
3:      $ratio = a_{ik} / a_{kk}$ 
4:     for  $j = 1$  to  $n$  in parallel do
5:        $a_{ij} - = ratio * a_{kj}$ 
6:     end for
7:   end for
8: end for
9:  $det = a_{11} * a_{22} * \dots * a_{nn}$ 

```

We:  $A[1:n][1:n]$

Wy:  $\det(a)$

Model: CREW PRAM, czas  $O(n)$  i  $O(n^2)$  procesorów..

[Strona główna](#)
[Strona tytułowa](#)
[◀◀](#)
[▶▶](#)
[◀](#)
[▶](#)
[Strona 4 z 22](#)
[Powrót](#)
[Full Screen](#)
[Zamknij](#)
[Koniec](#)

### 3. Transformata Fourier'a

**Algorytm 3:** Transformata Fourier'a

```
1: for  $k = 0$  to  $n - 1$  in parallel do
2:    $\omega_k = \cos 2\pi k/n + i \sin 2\pi k/n$ 
3: end for
4: for  $i = 0$  to  $n - 1$  in parallel do
5:   for  $j = 0$  to  $n - 1$  in parallel do
6:      $W_{ij} = \omega_{i*j}$ 
7:   end for
8: end for
9: for  $i = 0$  to  $n - 1$  in parallel do
10:   $Y_i = \sum_{j=0}^{n-1} W_{i,j} * x_j$ 
11: end for
```

We:  $X[0:n-1]$

Wy:  $X[0:n-1]$

Model: CREW PRAM, czas  $O(\log n)$  i  $O(n^2)$  procesorów..

Postorder (poprawka)

Wyznacznik macierzy

Transformata Fourier'a

Różniczkowanie i...

Problemy P-zupełne

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 5 z 22

Powrót

Full Screen

Zamknij

Koniec

## Jak to zrobić efektywniej?

Najpierw przyjrzymy się macierzom  $W$ .

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & -1 & -\omega & -\omega^2 & -\omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & -\omega^2 & \omega & -1 & -\omega^3 & \omega^2 & -\omega \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\omega & \omega^2 & -\omega^3 & -1 & \omega & -\omega^2 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & -\omega^3 & -\omega^2 & -\omega & -1 & \omega^3 & \omega^2 & \omega \end{bmatrix}$$

[Strona główna](#)
[Strona tytułowa](#)
[◀](#)
[▶](#)
[◀](#)
[▶](#)
[Strona 6 z 22](#)
[Powrót](#)
[Full Screen](#)
[Zamknij](#)
[Koniec](#)

Zauważmy następujące fakty dla  $n$  parzystych:

**Fakt 1** *Jeśli  $i$  jest parzyste oraz  $0 \leq i \leq n - 1$ , to*

$$W_{i,j} = W_{i,j+n/2},$$

*dla  $0 \leq j \leq n/2$*

**Fakt 2** *Jeśli  $i$  jest nieparzyste oraz  $0 \leq i \leq n - 1$ , to*

$$W_{i,j} = -W_{i,j+n/2},$$

*dla  $0 \leq j \leq n/2$*

## Ćwiczenie 1

Uzasadnij powyższe.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 7 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Obliczając  $Y = WX$  mamy:

**Fakt 3** *Jeśli  $i$  jest parzyste oraz  $0 \leq i \leq n - 1$ , to*

$$y_i = \sum_{j=0}^{n/2-1} W_{i,j}(x_j + x_{j+n/2}),$$

*dla  $0 \leq j \leq n/2$*

**Fakt 4** *Jeśli  $i$  jest nieparzyste oraz  $0 \leq i \leq n - 1$ , to*

$$y_i = \sum_{j=0}^{n/2-1} W_{i,j}(x_j - x_{j+n/2}),$$

*dla  $0 \leq j \leq n/2$*

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 8 z 22

Powrót

Full Screen

Zamknij

Koniec

Podstawmy  $i = 2t$

$$\begin{aligned} y_{2t} &= \sum_{j=0}^{n/2-1} W_{i,j}(x_j + x_{j+n/2}) \\ &= \sum_{j=0}^{n/2-1} \omega^{ij}(x_j + x_{j+n/2}) \\ &= \sum_{j=0}^{n/2-1} (\omega^2)^{tj}(x_j + x_{j+n/2}) \end{aligned}$$

A więc wektor

$$\begin{bmatrix} Y_0 \\ Y_2 \\ \vdots \\ Y_{n-2} \end{bmatrix} \text{ jest transformatą } \begin{bmatrix} x_0 + x_{n/2} \\ x_1 + x_{n/2+1} \\ \vdots \\ x_{n/2-1} + x_{n-1} \end{bmatrix}$$

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 9 z 22

Powrót

Full Screen

Zamknij

Koniec

Podobnie podstawmy  $i = 2t + 1$  mamy:

$$\begin{aligned} y_{2t+1} &= \sum_{j=0}^{n/2-1} W_{i,j}(x_j - x_{j+n/2}), \\ &= \sum_{j=0}^{n/2-1} \omega^{(2t+1)j}(x_j - x_{j+n/2}), \\ &= \sum_{j=0}^{n/2-1} (\omega^2)^{tj} \omega^j (x_j - x_{j+n/2}) \end{aligned}$$

a więc wektor

$$\begin{bmatrix} Y_1 \\ Y_3 \\ Y_5 \\ \vdots \\ Y_{n-1} \end{bmatrix} \text{ jest transformata } \begin{bmatrix} x_0 - x_{n/2} \\ \omega(x_1 - x_{n/2+1}) \\ \omega^2(x_2 - x_{n/2+2}) \\ \vdots \\ \omega^{n/2-1}(x_{n/2-1} - x_{n-1}) \end{bmatrix}$$

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 10 z 22

Powrót

Full Screen

Zamknij

Koniec

Postorder (poprawka)

Wyznacznik macierzy

Transformata Fourier'a

Różniczkowanie i . . .

Problemy P-zupełne

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 11 z 22

Powrót

Full Screen

Zamknij

Koniec

Jeśli  $n = 2$  sytuacja jest następująca:

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 + x_1 \\ x_0 - x_1 \end{bmatrix}$$

Możemy teraz podać algorytm równoległy typu dziel i rządź.

**Algorytm 4:** Szybka transformata Fourier'a

1: **if**  $n = 1$  **then**  
 2:

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 + x_1 \\ x_0 - x_1 \end{bmatrix}$$

3: **end if**

4: Rekurencyjnie obliczaj współrzędne parzyste i nieparzyste według wzorów z poprzednich slajdów.

We:  $X[0:n-1]$

Wy:  $X[0:n-1]$

Model: EREW PRAM, czas  $O(\log n)$  i  $O(n)$  procesorów.

Komentarz: Kroki rekurencji nie kolidują ze sobą i mogą być obliczane równoległe, za każdym razem dzielimy problem na pół, mamy więc  $\log n$  kroków wykonywanych w czasie stałym.

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 12 z 22

Powrót

Full Screen

Zamknij

Koniec

## 4. Różniczkowanie i całkowanie

Niech  $f \in C^1$ ,  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Przypuśćmy, że mamy tabelę wartości funkcji  $f(x_i)$ ,  $i = 0 \dots n$  w punktach  $x_0 + hi$ . Przybliżenie pierwszej pochodnej funkcji  $f$  możemy otrzymać na podstawie jednego z wzorów:

$$f'_i = \frac{f_{i+1} - f_i}{h}$$

lub

$$f'_i = \frac{f_{i+1} - f_{i-1}}{2h}$$

Na podstawie pierwszego wzoru uzyskujemy następujący algorytm:

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Strona 13 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 14 z 22

Powrót

Full Screen

Zamknij

Koniec

## Algorytm 5: Różniczkowanie numeryczne

1: **for**  $i = 0$  **to**  $n - 1$  **in parallel do**

2:  $f'[i] = \frac{f[i+1] - f[i]}{h}$

3: **end for**

Czas  $O(1)$  i  $O(n)$  procesorów.

We: Wektor  $f = [f(x_0), \dots, f(x_0 + nh)]$ ,

Wy: Wektor  $f'$ .

Model: CREW PRAM.

Do przybliżenia kolejnych pochodnych możemy użyć np:

$$f_i'' = \frac{1}{h^2}(f_{i-1} - 2f_i + f_{i+1})$$

$$f_i'' = \frac{1}{12h^2}(11f_{i-1} - 20f_i + 6f_{i+1} + 4f_{i+2} + f_{i+3})$$

Podobnie możemy zrównoleglić całkowanie metodą trapezów:

**Algorytm 6:** Całkowanie metodą trapezów

- 1: **for**  $i = 0$  **to**  $n - 1$  **in parallel do**
- 2:    $integral[i] = h(f[i + 1] + f[i])/2$
- 3: **end for**
- 4:  $Integral = \sum_{i=0}^{n-1} integral[i]$

Czas  $O(\log n)$  i  $O(n)$  procesorów.

We: Wektor  $f = [a = f(x_0), \dots, f(x_0 + nh) = b]$ ,

Wy:  $Integral \approx \int_a^b f(x) dx$

Model: CREW PRAM.

[Strona główna](#)
[Strona tytułowa](#)
[◀◀](#)
[▶▶](#)
[◀](#)
[▶](#)
[Strona 16 z 22](#)
[Powrót](#)
[Full Screen](#)
[Zamknij](#)
[Koniec](#)

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 17 z 22

Powrót

Full Screen

Zamknij

Koniec

**Algorytm 7:** Całkowanie metodą Simpsona

- 1: **for**  $i = 0$  **to**  $n - 2$  **step** 2 **in parallel do**
- 2:  $integral[i] = h(f[i] + 4f[i + 1] + f[i + 2])/3$
- 3: **end for**
- 4:  $Integral = \sum_{i=0}^{n/2-1} integral[2 * i]$

Czas  $O(\log n)$  i  $O(n)$  procesorów.

We: Wektor  $f = [a = f(x_0), \dots, f(x_0 + nh) = b]$ ,  $n$  parzyste

Wy:  $Integral \approx \int_a^b f(x) dx$

Model: CREW PRAM.

## Ćwiczenie 2

Podaj algorytm CREW PRAM dla przybliżania  $f''$  na podstawie tabeli węzłów funkcji  $f$ .

## Ćwiczenie 3

Podaj algorytm EREW PRAM dla przybliżania  $f'$  na podstawie tabeli węzłów funkcji  $f$  używając  $O(n)$  procesorów w czasie  $O(1)$ .

## Ćwiczenie 4

Zapisz algorytm 7 w taki sposób, aby nie marnować pamięci w tablicy *integral*.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 18 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## 5. Problemy $P$ -zupełne – przykłady II

### Otoczki wypukłe

Dany jest skończony zbiór punktów na płaszczyźnie  $S \subset \mathbb{R} \times \mathbb{R}$ , liczba  $k$  i punkt  $p \in \mathbb{R} \times \mathbb{R}$ .

Stwierdzić, czy  $p$  należy do zbioru powstałego z  $S$  po usunięciu  $k$  otoczek wypukłych.

Kod w LtPC: A.9.5

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)

Strona 19 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## Planarny układ kombinacyjny

Dany jest kod  $\bar{\alpha}$  planarnego układu kombinacyjnego.  $\alpha$ , ciąg wejściowy  $x_1, \dots, x_n$  oraz wybrane wyjście  $y$ .

Stwierdzić, czy dla ciągu wejściowego  $x_1, \dots, x_n$  na wyjściu  $y$  jest **TRUE**. Kod w LtPC: A.1.7

## Kot i mysz

Dany jest graf skierowany  $G = (V, E)$  wraz z trzema wyróżnionymi wierzchołkami:  $c$ ,  $m$  i  $g$ .

Problem: stwierdzić, czy mysz ma strategię wygrywającą w grze kot i mysz w grafie  $G$ .

Gra przebiega w następujący sposób: kot znajduje się w wierzchołku  $c$ , a mysz w  $m$ . Gracze wykonują ruchy na przemian, w kolejnej turze najpierw mysz a potem kot. Ruch polega na przesunięciu się do sąsiedniego wierzchołka, lub pozostaniu w miejscu. Kot nie może znaleźć się w  $g$ . Kot wygrywa, jeśli znajdzie się w tym samym wierzchołku, co mysz natomiast mysz wygrywa, jeśli dotrze do wyróżnionego wierzchołka  $g$ .

Kod w LtPC: A.11.2

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)

Strona 21 z 22

[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## Izomorfizm grup

Dany jest zbiór skończony  $S$  oraz  $F$ , wolna grupa generowana przez  $S$ . Niech  $\hat{S} = \{s, s^{-1} \mid s \in S\}$ ,  $S^*$  zbiór wszystkich skończonych słów nad  $\hat{S}$ , a  $e$  słowem pustym.

Dla danych  $U = \{u_1, \dots, u_m\}$ ,  $V = \{v_1, \dots, v_p\} \subseteq S^*$  stwierdzić, czy  $\langle U \rangle = \langle V \rangle$ .

Kod w LtPC: A.8.12

## Przekrój podgrup

Dany jest zbiór skończony  $S$  oraz  $F$ , wolna grupa generowana przez  $S$ . Niech  $\hat{S} = \{s, s^{-1} \mid s \in S\}$ ,  $S^*$  zbiór wszystkich skończonych słów nad  $\hat{S}$ , a  $e$  słowem pustym.

Dla danych  $U = \{u_1, \dots, u_m\}$ ,  $V = \{v_1, \dots, v_m\} \subseteq S^*$  stwierdzić, czy  $\langle U \rangle \cap \langle V \rangle = \langle e \rangle$ .

Kod w LtPC: A.8.15

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)[Strona 24 z 22](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

## Przynależność do gramatyki bezkontekstowej

Dla gramatyki bezkontekstowej  $G = (N, T, P, S)$  i słowa  $x \in T^*$  sprawdzić, czy  $x \in L(G)$ .