

# Algorytmy Równoległe i Rozproszone

## Część I - Wprowadzenie i Sieci porównujące

Łukasz Kuszner  
pokój 209, WETI  
<http://www.sphere.pl/~kuszner/>  
kuszner@sphere.pl

Oficjalna strona wykładu  
<http://www.sphere.pl/~kuszner/ARiR/>  
Wykład 15 godzin, Projekt 15 godzin

2006/07

### Spis treści

<b>1</b>	<b>Zaliczenie</b>	<b>1</b>
<b>2</b>	<b>Plan wykładu</b>	<b>2</b>
<b>3</b>	<b>Literatura</b>	<b>3</b>
<b>4</b>	<b>Sieci porównujące</b>	<b>4</b>
4.1	Schematy komparatora . . . . .	4
<b>5</b>	<b>PRAM - przykład</b>	<b>5</b>
5.1	Iloczyn skalarny . . . . .	5
<b>6</b>	<b>Model rozproszony</b>	<b>6</b>
6.1	Rozproszony algorytm dla problemu: „wybór lidera” . . . . .	6

<b>7 Sieci porównujące</b>	<b>7</b>
7.1 Głębokość sieci . . . . .	8
7.2 Sieci sortujące . . . . .	8
<b>8 Zasada zero-jedynkowa</b>	<b>9</b>
<b>9 Sieci bitoniczne</b>	<b>11</b>
<b>10 Sieć sortująca</b>	<b>12</b>

## 1 Zaliczenie

Na zaliczenie przedmiotu składają się następujące elementy:

- $C$  - suma punktów uzyskanych na ćwiczeniach,
- $K$  - maksymalna suma punktów uzyskana z kolokwii, które odbędą się w trakcie trwania semestru.
- $D$  - suma punktów dodatkowych

Notatki

Końcowy wynik  $S$  obliczamy według wzoru:

$$S = \left( \frac{C + D + K}{100} \right) \cdot 100\%$$

Ocenę z przedmiotu wyznaczamy w zależności od  $S$  w następujący sposób:

ocena	wynik
2	$\max\{K_1 + K_2, E\} < 100 \vee C < 100$
3	$S \geq 50\% \wedge S < 60\%$
3+	$S \geq 60\% \wedge S < 70\%$
4	$S \geq 70\% \wedge S < 80\%$
4+	$S \geq 80\% \wedge S < 90\%$
5	$S \geq 90\% \wedge S < 100\%$
5+	$S \geq 100\%$

Notatki

## 2 Ogólny plan wykładu

- Wstęp
- Sieci porównujące,
- Układy kombinacyjne,
- Algorytmy w modelu PRAM,
- Algorytmy w modelu rozproszonym.

Notatki

## 3 Literatura

- T. H. Cormen, C. E. Leiserson and R. L. Rivest, „Introduction to Algorithms”, The MIT Press/McGraw-Hill Company, 1990 (wydanie polskie WNT).
- Raymond Greenlaw, H. James Hoover, Walter L. Ruzzo „Limits to Parallel Computation: P-Completeness Theory”, Oxford University Press, 1998.
- Hagit Attiya „Lecture Notes for Course Distributed Algorithms”, 1994.

Notatki

- C. Xavier, S. S. Iyengar, „Introduction to Parallel Algorithms”, Wiley-IEEE, 1998.
- Gerard Tel, „Introduction to Distributed Algorithms”, Cambridge University Press, 2nd edition, 2000.
- Shlomi Dolev. Self-Stabilization. MIT Press, 2000.

Notatki

- J. Jaja, „An Introduction to Parallel Algorithms”, Addison-Wesley, Reading, MA, 1992. ■
- S.G. Akl, „The Design and Analysis of Parallel Algorithms”, Prentice-Hall, 1989. ■
- Lecture Notes Repository: Uniwersytet w Paderborn.

Notatki

- Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar „Introduction to Parallel Computing”, Addison Wesley, 2003. ■
- Hagit Attiya, Jennifer Welch „Distributed Computing: Fundamentals, Simulations, and Advanced Topics”, 2nd Edition. ■
- Guy E. Blelloch, Bruce M. Maggs: Parallel Algorithms. The Computer Science and Engineering Handbook, 1997: 277-315.

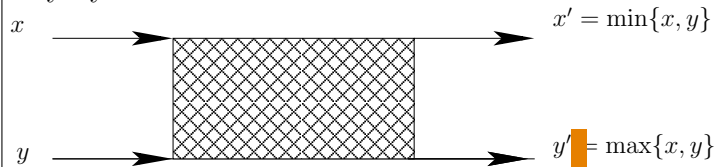
Notatki

## 4 Sieci porównujące - przykład

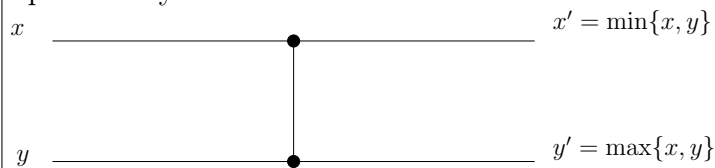
Sieć porównująca składa się z połączonych ze sobą komparatorów.

### 4.1 Schematy komparatora

zwykły:

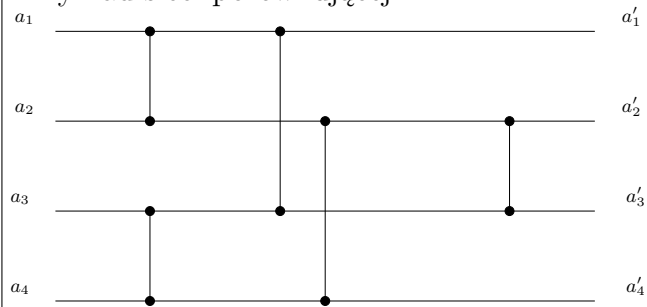


uproszczony:



Notatki

### Przykład sieci porównującej



Notatki

### Ćwiczenie 1

Sprawdź, że sieć porównująca z poprzedniego slajdu jest siecią sortującą.

Notatki

## 5 PRAM - przykład

### 5.1 Iloczyn skalarny

We: Tablice współrzędnych  $a[1:n]$  i  $b[1:n]$

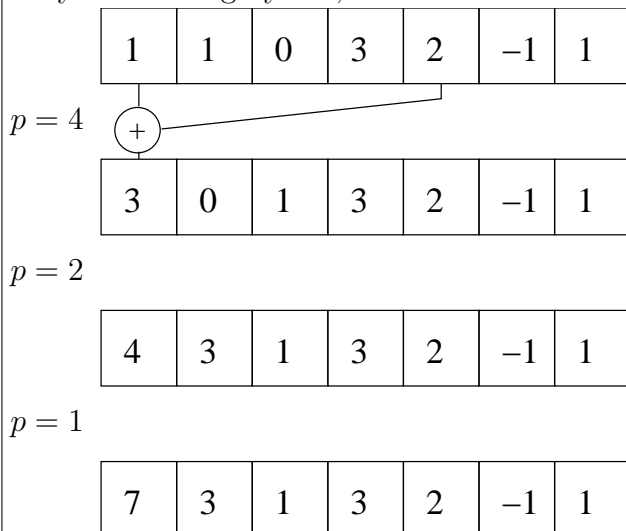
Wy: Liczba będąca iloczynem skalarnym wektorów  $a$  i  $b$ .

**Algorytm 1:** Iloczyn skalarny

```
1: for  $i = 1$  to  $n$  in parallel do
2:    $c_i = a_i * b_i$ 
3: end for
4:  $p = n/2$ 
5: while  $p > 0$  do
6:   for  $i = 1$  to  $p$  in parallel do
7:      $c_i = c_i + c_{i+p}$ 
8:   end for
9:    $p = p/2$ 
10: end while
```

Notatki

Przykład dla algorytmu, dla  $n = 7$ .



Notatki

## 6 Model rozproszony - przykład

### 6.1 Rozproszony algorytm dla problemu: „wybór lidera”

Przypuśćmy, że mamy rozproszony system asynchroniczny, w którym jednostki obliczeniowe tworzą pierścień.

Problem polega na skonstruowaniu algorytmu, który pozwoli wyróżnić jeden z procesorów. Oczekujemy, że po zakończeniu działania algorytmu dokładnie jeden procesor zasygnalizuje „jestem liderem”, a wszystkie pozostałe „nie jestem liderem”.

Notatki

Przypuśćmy, że każdy z procesorów ma unikalny identyfikator. Procesory tworzą pierścień, możemy więc mówić o lewym i prawym sąsiedzie. Inicjalnie każdy z procesorów wysyła wiadomość do swojego lewego sąsiada, w której zapisuje swój identyfikator.

Następnie każdy procesor odbierając wiadomość od prawego sąsiada przesyła ją dalej, jeśli zapisany na niej identyfikator jest większy od jego własnego identyfikatora i kasuje, jeśli jest mniejszy.

Procesor, który otrzymał z powrotem swój własny identyfikator ogłasza się liderem i wysyła wiadomość kończącą działanie algorytmu.

Notatki

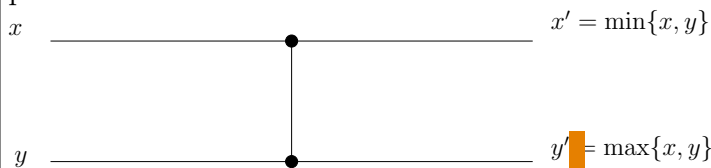
### Ćwiczenie 2

Uzasadnij, że liczba wiadomości wysłanych w trakcie działania algorytmu jest  $O(n^2)$ .

Notatki

## 7 Sieci porównujące

Sieć porównująca składa się tylko z przewodów i komparatorów.



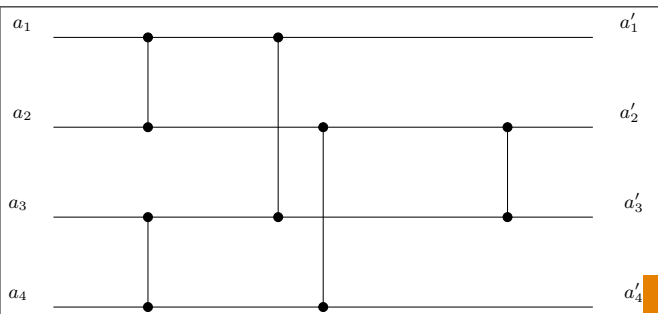
Zakładamy, że komparator działa w czasie  $O(1)$ , to znaczy czas pomiędzy pojawieniem się danych na wejściu (na rysunku po lewej stronie), a pojawieniem się wyników na wyjściu (na rysunku po prawej stronie) jest stały. Przyjmujemy, że sieć ma  $n$  wejść  $a_1, a_2, \dots, a_n$  i  $n$  wyjść  $b_1, b_2, \dots, b_n$ , a graf sieci jest acykliczny.

Notatki

### 7.1 Głębokość sieci

Wejście sieci ma głębokość 0. Jeśli wejścia komparatora mają głębokość  $d_x$  i  $d_y$ , to wyjście ma głębokość  $\max(d_x, d_y) + 1$ . Głębokość sieci definiujemy jako największą głębokość wyjścia. Głębokość identyfikujemy z czasem potrzebnym na posortowanie wszystkich elementów.

Notatki



poniższa sieć porównująca ma głębokość 3.

## 7.2 Sieci sortujące

*Sieć sortująca*, to sieć porównująca, której ciąg wyjściowy jest niemalejący dla każdego ciągu wejściowego.

### Ćwiczenie 3

Wykaż, że sieć sortująca o  $n$  wejściach ma głębokość co najmniej  $\lg n$ .

Notatki

## 8 Zasada zero-jedynkowa

**Lemat 1** *Jeśli sieć porównująca dla ciągu wejściowego  $a = (a_1, a_2, \dots, a_n)$  wyznacza ciąg wyjściowy  $b = (b_1, b_2, \dots, b_n)$ , to dla dowolnej funkcji nie-malejącej ta sama sieć z ciągiem wejściowym  $f(a) = (f(a_1), f(a_2), \dots, f(a_n))$  wyznacza ciąg wyjściowy  $f(b) = (f(b_1), f(b_2), \dots, f(b_n))$ .*

### Ćwiczenie 4

Przeprowadź pełny dowód lematu 1.

**Wskazówka 1:** Rozważ pojedynczy komparator. Z monotoniczności  $f$  wynika, że  $\max\{f(x), f(y)\} = f(\max\{x, y\})$  i  $\min\{f(x), f(y)\} = f(\min\{x, y\})$

**Wskazówka 2:** Zastosuj indukcję na głębokość sieci.

**Wskazówka 3:** zob. Cormen str. 718

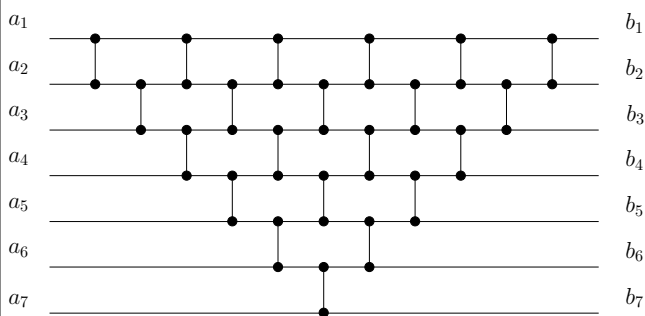
**Twierdzenie 2** *Jeśli sieć porównująca o  $n$  wejściach sortuje poprawnie wszystkie  $2^n$  ciągi zer i jedynek, to sortuje poprawnie dowolne ciągi liczb.*



**Dowód:** Dowód nie wprost. Przypuśćmy, że sieć sortuje poprawnie wszystkie ciągi zer i jedynek, ale istnieje ciąg liczb  $a = (a_1, a_2, \dots, a_n)$ , dla którego wynik nie jest posortowany poprawnie. Istnieją elementy  $a_i, a_j$  takie, że  $a_i < a_j$  oraz w ciągu wyjściowym  $a_j$  występuje przed  $a_i$ . Niech  $f(x) = 0$ , jeśli  $x \leq a_i$  i  $f(x) = 1$ , jeśli  $x > a_i$ . Funkcja  $f$  jest niemalejąca. Z lematu 1 wnioskujemy, że jeśli na wejściu znajduje się ciąg  $f(a) = (f(a_1), f(a_2), \dots, f(a_n))$ , to na wyjściu wartość  $f(a_j) = 1$  znajduje się przed  $f(a_i) = 0$ , sprzeczność z założeniem o poprawnym sortowaniu ciągów zero-jedynkowych.  $\square$

Notatki

Rodzina sieci opartych na algorytmie sortowania przez wstawianie.



### Ćwiczenie 5

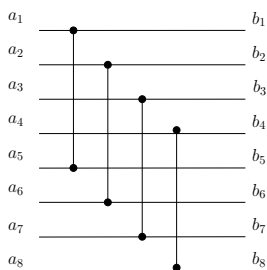
Jaka jest głębokość takiej sieci w zależności od liczby wejść?

Notatki

## 9 Bitoniczne sieci sortujące

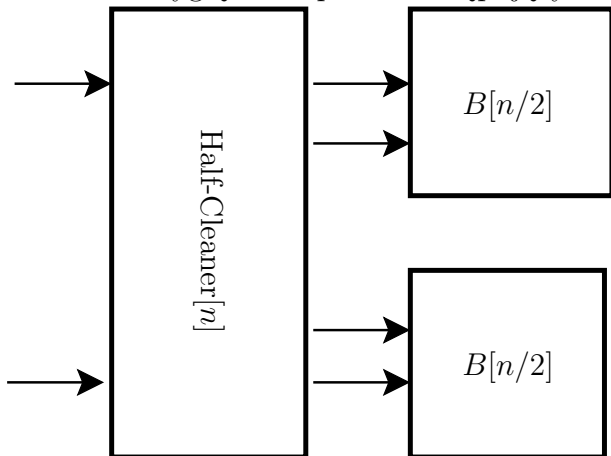
Ciągiem bitonicznym nazywamy ciąg, który można przeciąć na dwie części: pierwszą niemalejącą i drugą nierosnącą lub pierwszą nierosnącą i drugą niemalejącą. Zero-jedynkowe ciągi bitoniczne mają postać  $0^i 1^j 0^k$  albo  $1^i 0^j 1^k$ .

Elementem półczyszczącym (Half-Cleaner[n]), dla  $n$  parzystych nazywamy sieć porównującą o głębokości 1 złożoną z  $n/2$  komparatorów, z których każdy łączy wejście o numerze  $i$  z wejściem  $i + n/2$ , dla  $i = 1, 2, \dots, n/2$ .



Notatki

Sieć sortującą ciągi bitoniczne  $B[n]$  budujemy w następujący sposób. Dla  $n = 1$  sieć jest pusta, dla  $n = 2$  sieć składa się z jednego komparatora, natomiast dla  $n > 2$  sieć wygląda w sposób następujący:



Notatki

### Ćwiczenie 6

Narysuj sieć  $B[8]$ .

### Ćwiczenie 7

Jak skonstruować sieć bitoniczną, gdy  $n$  nie jest potęgą 2?

### Ćwiczenie 8

Uzasadnij, że jeśli każdy element jednego z dwóch zero-jedynkowych ciągów bitonicznych jest nie większy niż każdy element drugiego ciągu, to jeden z ciągów składa się z samych 0, albo z samych 1.

### Ćwiczenie 9

Przeczytaj ze zrozumieniem pełen dowód poprawności działania sieci  $B[n]$ , (Cormen str. 721) i odtwórz go nie zaglądając do książki.

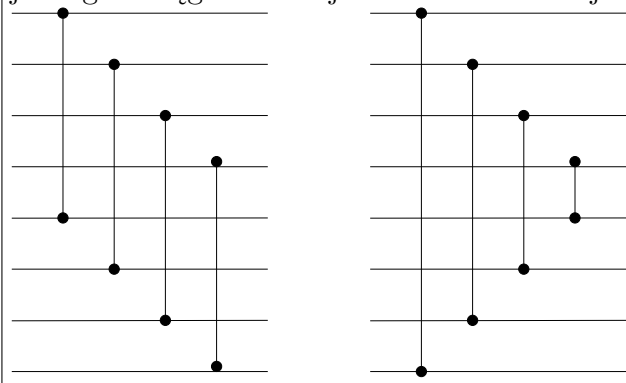
Notatki

## 10 Sieć sortująca o głębokości $\log^2 n$

Pokażemy teraz jak skonstruować sieć opartą na algorytmie `mergesort`. Do tego celu użyjemy pokazanych wcześniej sieci bitonicznych i sieci scalających. Konstrukcja sieci scalającej dwa posortowane ciągi w jeden opiera się na prostej obserwacji: jeśli w jednym z dwóch posortowanych ciągów odwrócimy porządek, to otrzymamy ciąg bitoniczny, a ciągi bitoniczne umiemy już sortować.

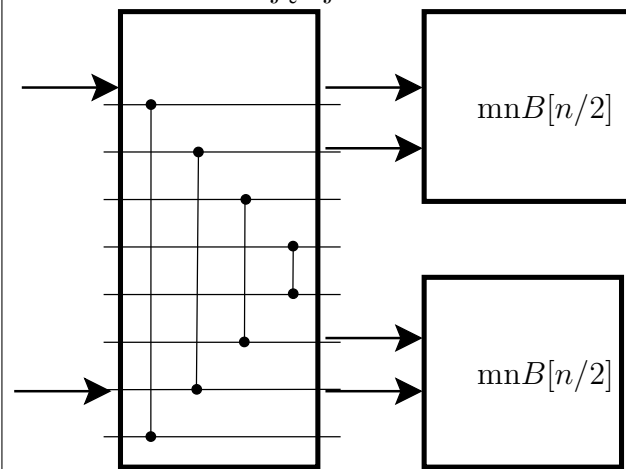
Notatki

Poniższy rysunek przedstawia jak odwrócić porządek jednego z ciągów na wejściu sieci bitoniczej.



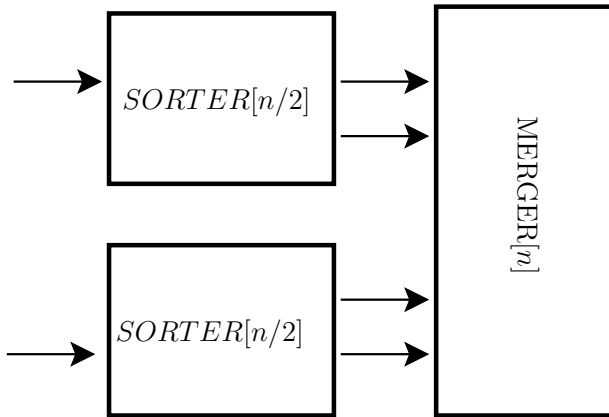
Notatki

Schemat sieci scalającej MERGER[ $n$ ]



Notatki

Mając już wszystkie potrzebne elementy możemy zbudować sieć sortującą  $SORTER[n]$



Notatki

### Ćwiczenie 10

Narysuj sieć  $SORTER[8]$ .

### Ćwiczenie 11

Wykaż, że głębokość sieci  $SORTER[n]$  jest  $\Theta(\log^2(n))$ .

### Ćwiczenie 12

Wykaż, że głębokość sieci  $SORTER[n]$  jest równa  $\lg n(\lg n + 1)/2$ .

Notatki