

Algorytmy Równoległe i Rozproszone

Część V - Model PRAM II

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

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa

⏪

⏩

◀

▶

Strona 1 z 28

Powrót

Full Screen

Zamknij

Koniec

1. Jakość algorytmów równoległych

- Czas obliczeń (złożoność algorytmu)
- Liczba potrzebnych procesorów.
- Przyjęty model obliczeń
- Przyspieszenie
- Efektywność
- Skalowalność

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 2 z 28

Powrót

Full Screen

Zamknij

Koniec

Przyspieszenie

Rozważmy problem, dla którego sekwencyjny algorytm wymaga czasu T_s . Dysponujemy algorytmem, który działa w czasie T_p przy użyciu P procesorów.

$$\text{Speedup} = \frac{T_s}{T_p}$$

$$\text{Efficiency} = \frac{T_s}{PT_p}$$

W jaki sposób obliczamy czasy T_s i T_p ?

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 3 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Przyspieszenie względne (Relative Speedup)

Niech \mathcal{A} będzie algorytmem równoległym.

$$\text{RelativeSpeedup}(n, p) = \frac{T_s}{T_p}$$

- T_s =Czas rozwiązania problemu P na jednym procesorze
- T_p =Czas rozwiązania problemu P na p procesorach

[Strona główna](#)[Strona tytułowa](#)[◀◀](#) [▶▶](#)[◀](#) [▶](#)[Strona 4 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Przyspieszenie rzeczywiste (Real Speedup)

Niech \mathcal{A} będzie algorytmem równoległym.

$$\text{RealSpeedup}(n, p) = \frac{T_s}{T_p}$$

- T_s = Czas rozwiązania problemu najlepszym znanym algorytmem sekwencyjnym,
- T_p = Czas rozwiązania problemu P na p procesorach.

W obu wypadkach mierzymy czas na maszynie równoległej.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 5 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Przyspieszenie bezwzględne (Absolute Speedup)

Niech \mathcal{A} będzie algorytmem równoległym.

$$\text{AbsoluteSpeedup}(n, p) = \frac{T_s}{T_p}$$

- T_s = Czas rozwiązania problemu najlepszym znanym algorytmem sekwencyjnym na najszybszym znanym procesorze,
- T_p = Czas rozwiązania problemu P na p procesorach.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 6 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Asymptotyczne przyspieszenie rzeczywiste (Asymptotic Real Speedup)

Niech $S(n)$ będzie złożonością obliczeniową najlepszego algorytmu sekwencyjnego, a \mathcal{A} algorytmem równoległym i $P_A(n)$ jego złożonością bez ograniczenia na liczbę procesorów.

$$\text{AsymptoticRealSpeedup}(n, p) = \frac{S(n)}{P(n)}$$

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 7 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 8 z 28

Powrót

Full Screen

Zamknij

Koniec

Cost Normalized Speedup

$$\text{CNS}(n, p) = \frac{\text{Speedup}(n, p)}{\frac{\text{koszt systemu równoległego}}{\text{koszt systemu sekwencyjnego}}}$$

Efektywność

Efektywność jest miarą ściśle związaną z przyspieszeniem.
Ogólnie można zapisać

$$\text{Efficiency} = \frac{\text{Speedup}}{p},$$

gdzie p jest liczbą użytych procesorów.

W zależności od tego jaką przyjmujemy formę przyspieszenia uzyskamy różne miary efektywności.

[Strona główna](#)[Strona tytułowa](#)[Strona 9 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

[Strona główna](#)[Strona tytułowa](#)[◀◀](#)[▶▶](#)[◀](#)[▶](#)[Strona 10 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Skalowalność

Intuicyjnie system/algorytm jest skalowalny, jeśli efektywność maleje wolno wraz ze wzrostem rozmiaru problemu i liczby procesorów.

2. Metoda cyklu Eulera

Niech $G = (V, E)$ spójny graf prosty. Możemy utworzyć graf G' o tym samym zbiorze wierzchołków V oraz zbiorze krawędzi E' otrzymanym przez zastąpienie każdej nieskierowanej krawędzi $E \ni e = \{u, v\}$ poprzez dwie krawędzie skierowane (u, v) i (v, u) .

Fakt 1 *Otrzymany w ten sposób graf jest Eulerowski.*

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

Strona 11 z 28

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

Rozważmy teraz drzewo T . Zaczniemy od znalezienia cyklu Eulera w $T' = (V, E')$.

Niech $v \in V$ będzie wierzchołkiem w T , a $N(v) = \{u_0, u_1, u_2, \dots, u_{\deg(v)-1}\}$ listą sąsiadów. Istotne jest, że dla każdego wierzchołka v zbiór sąsiadów $N(v)$ musi być uporządkowany. Dla każdej krawędzi (u_i, v) definiujemy następnik $\text{succ}(u_i, v) = (v, u_{i+1(\text{mod } \deg(v))})$

Fakt 2 *Tak zdefiniowana funkcja succ – (następnik) definiuje cykl Eulera w T' .*

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

Strona 12 z 28

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

Przykład

Zakładając kolejność sąsiadów:

1 : {2},

2 : {1, 3, 4},

3 : {2},

4 : {2, 5},

5 : {4, 6, 7, 8},

6 : {5},

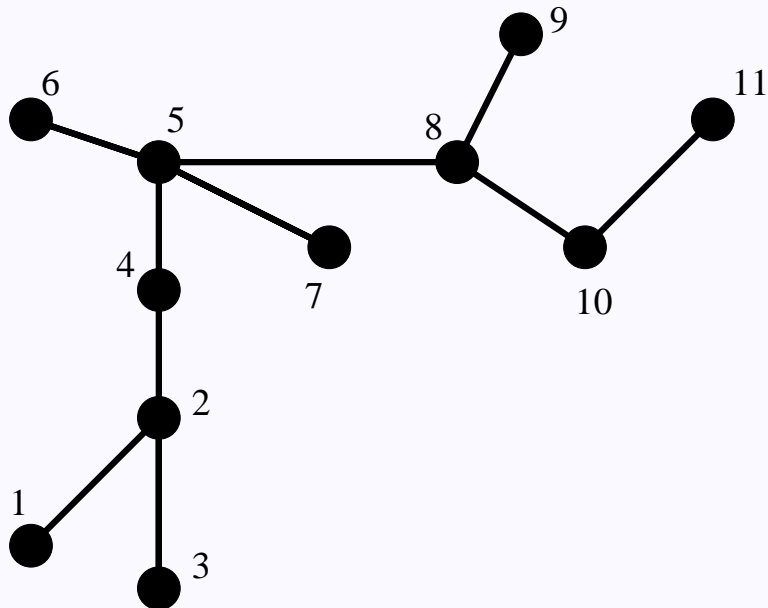
7 : {5},

8 : {5, 9, 10},

9 : {8},

10 : {8, 11},

11 : {10},



Na tym rysunku uzyskamy cykl:

1, 2, 3, 2, 4, 5, 6, 5, 7, 5, 8, 9, 8, 10, 11, 10, 8, 5, 4, 2, 1

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 13 z 28

Powrót

Full Screen

Zamknij

Koniec

Mając krawędzie drzewa ułożone w cykl możemy stosować metody typu „pointer jumping” dla drzew. Otrzymujemy w ten sposób metodę konstruowania algorytmów przy użyciu $O(n)$ procesorów i logarytmicznym czasie działania.

Ćwiczenie 1

Zaprojektuj efektywny algorytm równoległy obliczania sumy wszystkich elementów zapamiętanych w strukturze drzewiastej.

[Strona główna](#)[Strona tytułowa](#)[Strona 15 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Kolejność Postorder

We: Drzewo $T = (V, E)$ z korzeniem r wyróżnionym poprzez relację p , gdzie $p(u) = v$ - oznacza v jest rodzicem u w drzewie T oraz Cykl Eulera w T w formie relacji *succ*.

Wy: Dla każdego wierzchołka jego numer w kolejności Postorder $post(v)$.

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: $post(v) =$ suma prefiksowa wag w na łuku $(v, p(v))$.
- 11: **end for**

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

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 16 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Przykład

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

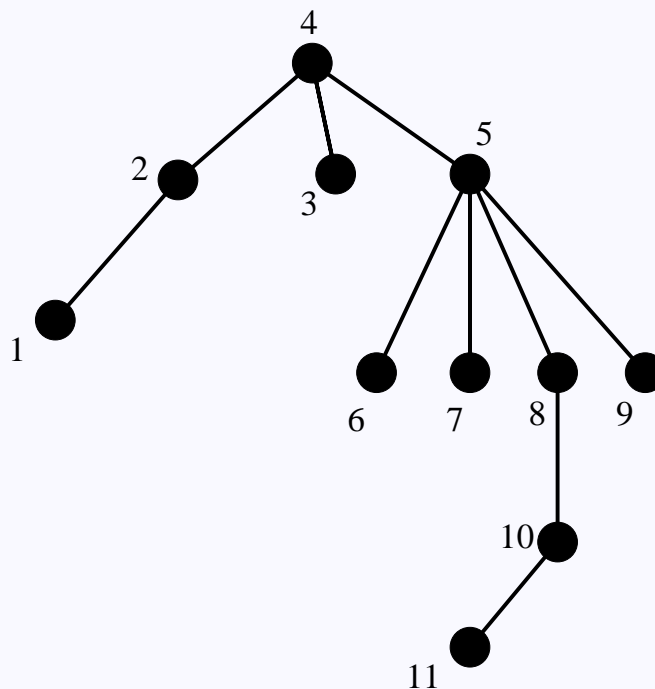
$$w(4, 3) = 0,$$

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

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

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

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



Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa



Strona 17 z 28

Powrót

Full Screen

Zamknij

Koniec

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 18 z 28

Powrót

Full Screen

Zamknij

Koniec

Ćwiczenie 2

Zaprojektuj algorytm typu EREW PRAM, który w czasie $O \log n$ oblicza rozmiary poddrzew o korzeniach we wszystkich węzłach drzewa binarnego.

3. Algorytm Floyd'a-Warshall'a

Rozważmy graf $G = (V, E)$, w którym z każdą krawędzią skojarzono nieujemną wagę w_{ij} . Uzupełniając przekątną zerami: $w_{ii} = 0$ i pozostałe wagi wartością nieskończoność: $w_{ij} = \infty$ (jeśli nie ma krawędzi z i do j) otrzymamy macierz wag $W = (w_{ij})$.

Algorytm Floyd'a Warshall'a pozwala obliczyć długość najkrótszej ścieżki z i do j , jak też i jej przebieg. Odtworzenie każdej ścieżki umożliwi macierz (p_{ij}) , w której element p_{ij} pokazuje wierzchołek poprzedni w stosunku do j w najkrótszej ścieżce z i do j .

Algorytm 2: Floyd-Warshall

```
1: for  $i = 1$  to  $n$  in parallel do
2:   for  $j = 1$  to  $n$  in parallel do
3:      $d_{ij} = w_{ij}$ 
4:      $p_{ij} = i$ 
5:   end for
6: end for
7: for  $k = 1$  to  $n$  do
8:   for każda para  $i, j$ , gdzie  $0 < i, j \leq n$  i  $i, j \neq k$  in parallel
   do
9:     if  $d_{ij} > d_{ik} + d_{kj}$  then
10:       $d_{ij} = d_{ik} + d_{kj}$ 
11:       $p_{ij} = p_{kj}$ 
12:    end if
13:  end for
14: end for
```

We: Graf w postaci macierzy wag w_{ij}

Wy: Macierze d_{ij} i P_{ij}

Model: CREW PRAM.

Czas $O(n)$ i $O(n^2)$ procesorów.

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa

◀◀ ▶▶

◀ ▶

Strona 20 z 28

Powrót

Full Screen

Zamknij

Koniec

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

Strona główna

Strona tytułowa

◀▶

◀▶

Strona 21 z 28

Powrót

Full Screen

Zamknij

Koniec

Ćwiczenie 3

Zaprojektuj algorytm typu CREW PRAM, który w czasie $O(n)$ znajdzie przechodnie domknięcie relacji binarnej.

4. Sortowanie przez *ranking*

Czas $O(\log n)$ i $O(n^2)$ procesorów.

We: Wektor do posortowania $X = [x_1, \dots, x_n]$

Model: CREW PRAM.

Algorytm 3: Sortowanie przez *ranking*

```
1: for każda para  $i, j$ , gdzie  $0 < i, j \leq n$  in parallel do
2:   if  $x_i > x_j$  then
3:      $c_{ij} = 1$ 
4:   else
5:      $c_{ij} = 0$ 
6:   end if
7: end for
8: for  $i = 1$  to  $n$  in parallel do
9:   policz  $r_i = \sum_{j=1}^n c_{ij}$ 
10: end for
11: for  $i = 1$  to  $n$  in parallel do
12:   ustaw element  $i$  na pozycji  $r_i$  w tablicy wynikowej
13: end for
```

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

Strona 23 z 28

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

5. Problemy P-zupełne – przykłady

Zbiór niezależny

Dany jest graf nieskierowany $G = (V, E)$ z identyfikatorami, oraz wyróżniony wierzchołek $v \in V$.

Stwierdzić, czy v należy do leksykograficznie pierwszego maksymalnego zbioru niezależnego (maksymalnej kliky, $\Delta + 1$ pokolorowania wierzchołkowego)

Kod w LtPC: A.2.1, A.2.2, A.2.6.

[Strona główna](#)[Strona tytułowa](#)[◀](#) [▶](#)[◀](#) [▶](#)[Strona 24 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Zachłanny zbiór dominujący

Dany jest graf nieskierowany $G = (V, E)$ z identyfikatorami, oraz wyróżniony wierzchołek u .

Rozstrzygnąć, czy u należy do zbioru dominującego znalezionej algorytmem zachłannym. Algorytm zachłanny dokłada do zbioru za każdym razem wierzchołek o największej liczbie nie zdominowanych sąsiadów i najmniejszym identyfikatorze.

Kod w LtPC: A.2.14

Maksymalny przepływ

Dany jest graf skierowany $G = (V, E)$ z wagami na krawędziach, oraz dwa wyróżnione wierzchołki: źródło (ang. source) i odpływ (ang. sink).

Rozstrzygnąć, czy w G istnieje przepływ f .

Kod w LtPC: A.4.4

Miary algorytmów

Cykl Eulera

Floyd-Warshall

Sortowanie

Problemy P-zupełne

Problemy otwarte

6. Problemy otwarte – przykłady

Kolorowanie krawędziowe

Dany jest graf nieskierowany G .

Znaleźć $\Delta + 1$ kolorowanie krawędziowe G

Kod w LtPC: B.9.3

Strona główna

Strona tytułowa

◀ ▶

◀ ▶

Strona 27 z 28

Powrót

Full Screen

Zamknij

Koniec

[Strona główna](#)[Strona tytułowa](#)[Strona 28 z 28](#)[Powrót](#)[Full Screen](#)[Zamknij](#)[Koniec](#)

Izomorfizm poddrzew

Dane są dwa nieukorzenione drzewa T i T' .

Rozstrzygnąć, czy T jest izomorficzne z pewnym poddrzewem T'

Kod w LtPC: B.9.9