Ćwiczenia 2

"Ukorzenienie" lasu, $O(n)$ procesorów CREW, czas $O(\log n)$

Dane: Dany las w postaci tablicy $parent[1..n]$, $parent[i]$ zawiera wskaźnik do ojca węzła i (lub nil gdy i jest korzeniem)
Wynik: Tablica $root[1..n]$, w $root[i]$ powinien zostać zapisany wskaźnik do korzenia drzewa zawierającego węzeł i

---

1. inicjalizacja tablicy $root[1..n]$:
   - $root[i]$: i gdy $parent[i] = nil$,
   - nil gdy $parent[i] \neq nil$,
2. niezmiennik: dla węzła i, jeśli $parent[i] = nil$ to $root[i]$ zawiera prawidłową wartość.
3. A:=$\{1..\text{n}\}$
   while $A \neq$ **do**
     for $i \in A$ **pardo**
       if $parent[i] = nil$ **then**
         A:=$A - \{i\}$
       else begin
         $root[i] := root[parent[i]]$
         $parent[i] := parent[parent[i]]$
       end

Obliczanie głębokości wierzchołków, $O(n)$ procesorów CREW, czas $O(\log n)$

Dane: Drzewo (las) dany w postaci tablicy $parent[1..n]$
Wynik: Tablica $depth[1..n]$ powinna zawierać głębokości poszczególnych wierzchołków

---

1. inicjalizacja tablicy $depth[1..n]$ wartościami: 0 gdy $parent[i] = nil$, 1 gdy $parent[i] \neq nil$,
2. niezmiennik: dla węzła i, jeśli $parent[i] = nil$ to $depth[i]$ zawiera głębokość węzła i, wpp. jest w niej zapisana odległość pomiędzy i a $parent[i]$.
3. A:=$\{1..\text{n}\}$
   while $A \neq$ **do**
     for $i \in A$ **pardo**
       if $parent[i] = nil$ **then**
         A:=$A - \{i\}$

else begin

$depth[i] := depth[i] + depth[\,parent[i]\,]$

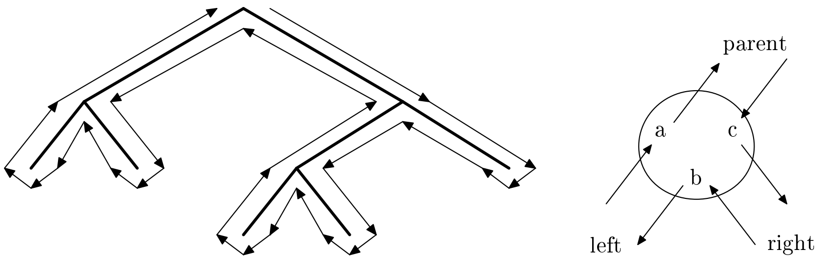
$parent[i] := parent[\,parent[i]\,]$

end

Szukanie głębokości wierzchołka w drzewie binarnym, n procesorów EREW, czas: $O(\log n)$

Dane: Drzewo binarne. W każdym wierzchołku zapisane są wartości $parent$, $left$, $right$

Wynik: Każdy wierzchołek zawiera wypełnione pole $depth$



Rys. 1. Wierzchołki połączone w listę, oraz zbliżenie na pojedynczy węzeł drzewa.

Do rozwiązania problemu, można użyć metody cyklu Eulera.

- każdemu wierzchołkowi przypisane są trzy procesory (a, b i c), które są połączone w listę (tak jak na rysunku 1),
- procesorom w węzłach przypisane są następujące wartości: $a = 1$, $b = 0$, $c = -1$,
- obliczenie sum prefiksowych dla listy, (ten krok zajmuje $O(\log n)$ czasu), (dla ustalenia uwagi, sumy prefiksowe liczymy w ten sposób, że element listy będzie zawierał sumę wartości elementów go poprzedzających na liście, i jego samego)
- dla każdego wierzchołka: $depth = sum[b]$.

Obliczenie numerów wierzchołków, n procesorów EREW, czas: $O(\log n)$

Dane: Drzewo binarne. W każdym wierzchołku zapisane są wartości $parent$, $left$, $right$

Wynik: Każdy wierzchołek zawiera wypełnione pola: *in_nr* (inorder), *pre_nr* (preorder), *post_nr* (postorder)

Problem jest bardzo podobny do poprzedniego, również można rozwiązać go stosując metodę cyklu Eulera. Należy jednak inaczej zainicjalizować wartości w węzłach:

Porządek	a	b	c	Wynik
inorder	0	1	0	$sum[b]$
preorder	0	0	1	$sum[c]$
postorder	1	0	0	$sum[a]$

Zliczanie liczby potomków w drzewie, n procesorów EREW, czas: $O(\log n)$

Dane: Drzewo binarne. W każdym węźle zapisane są wartości *parent*, $left$, *right*
Wynik: Każdy węzeł zawiera wypełnione pole *size* (liczba węzłów w poddrzewie przez niego wyznaczonym)

Tu również można zastosować metodę cyklu Eulera:

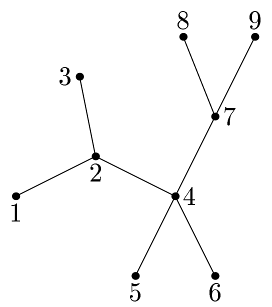
- wartości początkowe w procesorach są analogiczne do użytych w obliczaniu *postorder* (tzn. $a = 1$, $b = 0$, $c = 0$),
- obliczenie sum prefikowych,
- dla każdego węzła: $size = 1 + sum[c] - sum[a]$.

Cykl Eulera dla drzewa dowolnego stopnia

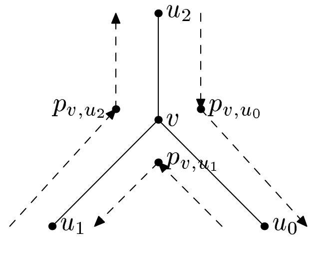
Dane: drzewo dowolnego stopnia dane w postaci grafu (lista sąsiedztwa)
Wynik: cykl Eulera, ustalenie wskaźnika parent dla każdego wierzchołka drzewa

---

v	$\{u_i\} = adj(v)$
1	2
2	1, 3, 4
3	2
4	2, 5, 6, 7
5	4
6	4
7	8, 4, 9
8	7
9	7



Każdemu wierzchołkowi v przypisano d procesorów (gdzie d — stopień wierzchołka), należy teraz tak ustalić wskaźniki *next* procesorom (przerywane linie na rysunku), by wszystkie zostały połączone w cykl.



Można tu zastosować następującą metodę:

- niech wierzchołek v ma następujących sąsiadów: $adj(v) = u_0, u_1, u_2, \ldots, u_{d-1}$, i zostały mu przypisane procesory: $p_{v,u_0}, p_{v,u_1}, \ldots, p_{v,u_{d-1}}$.
- dla $0 \le i < d$, $NEXT(p_{u_i,v}) = p_{v,(i+1)\ mod\ d}$.

Dla drzewa z przykładu cykl ma postać:

$$p_{4,5} \to p_{5,4} \to p_{4,6} \to p_{6,4} \to p_{4,7} \to p_{7,9} \to p_{9,7} \to p_{7,8} \to p_{8,7} \to p_{7,4} \to p_{4,2} \to$$

$$p_{2,1} \to p_{1,2} \to p_{2,3} \to p_{3,2} \to p_{2,4} \to cykl$$

Ukorzenienie drzewa:

- dla wybranego jako korzeń wierzchołka i, należy przerwać cykl przed dowolnym procesorem $p_{i,j}$,
- każdemu procesorowi przypisywana jest wartość +1,
- obliczenie sum prefiksowych na otrzymanych listach,
- dla każdego węzła v (o stopniu większym od 1), t.że: $adj(v) = u_0, u_1, u_2, \ldots, u_{d-1}$, z obliczonymi sumami prefiksowymi $s_{v,u_0}, s_{v,u_1}, \ldots, s_{v,u_{d-1}}$, jako ojca wierzchołka v należy wybrać u_i, takie, że $s_{v,i} < s_{v,(i+1)\ mod\ d}$.

Przykład:
jeśli dla przykładowego drzewa wybierze się węzeł 4, a cykl przerwie przed procesorem $p_{4,5}$, to dla wierzchołka 2 ($adj(2) = 1,3,4$), otrzymane zostaną sumy: $s_{2,1} = 5$, $s_{2,3} = 3$, $s_{2,4} = 1$, tak więc ojcem węzła 2 zostanie $u_3 = 4$.

Quicksort, n randomizowanych procesorów CRCW, czas (oczekiwany): $O(\log n)$

Dane: Tablica $A[1..n]$ zawierająca liczby całkowite
Wynik: Tablica $B[1..n]$ zawierająca posortowane wartości

Randomizowany CRCW oznacza taki model, w którym, gdy p procesorów próbuje jednocześnie zapisać pewne wartości do jednej komórki, to zostaje w niej zapisana jedna z tych wartości (każda z prawdopodobieństwem $1/p$).

1. inicjalizacja:
   (a) każdemu elementowi z tablicy zostaje przypisany jeden procesor,
   (b) wszystkie procesory zapisują do ustalonej komórki X swój indeks, (odpowiada to losowaniu jednego z procesorów),
   (c) wszystkie procesory czytają wartość z X, dla procesora i, jeśli $X = i$, to procesor i staje się nieaktywny, wpp, $p[i] = X$,
2. dopóki chociaż jeden procesor jest aktywny:
   (a) dla wszystkich aktywnych procesorów, procesor i porównuje wartości $A[i]$ i $A[p[i]]$, jeśli $A[i] > A[p[i]]$, to zapisuje $right[p[i]] := i$, wpp. zapisuje: $left[p[i]] := i$,
   (b) procesory które próbowały coś zapisać, sprawdzają jakie wartości zostały ostatecznie zapisane, gdy zapis się udał (tzn. wpisały swoją wartość), procesor staje się nieaktywny, wpp. ustawia wartość $p[i]$ na tą która została wpisana.

3. po zakończeniu poprzedniego kroku, wartości p, $left$, *right* wyznaczają pewne losowe drzewo BST, należy teraz obliczyć numery węzłów w porządku inorder, są one równocześnie indeksami odpowiadających im wartości z A w posortowanym ciągu,

4. **for** `i:=1` **to** `n` **pardo** `B[inorder[i]]:=A[i].`