Fontes principais

1. J. Jaja, An introduction to Parallel Algorithms, Addison Wesley, 92
   - ▷ Algoritmos paralelos

2. E. Cáceres, H. Mongeli, S. Song: Algoritmos paralelos usando CGM/PVM/MPI: uma introdução
   `http://www.ime.usp.br/~song/papers/jai01.pdf`

Algoritmos Paralelos em Grafos

Obtenção de circuitos de Euler em árvores

▷ Circuito de Euler: caminho fechado que passa por cada aresta do grafo exatamente uma vez.

▷ Um grafo direcionado é euleriano se e somente se, para cada vértice v do grafo, o grau de entrada(v) é igual ao grau de saída(v)

Obtenção de circuitos de Euler em árvores

▷ Dada uma árvore T qualquer, T pode ser transformado em um grafo direcionado D, substituindo-se cada aresta da árvore por duas arestas direcionadas anti-paralelas.

▷ O grafo obtido é euleriano. O algoritmo encontrará o circuito de Euler neste grafo.

Representação da árvore e do grafo direcionado

▷ Para cada vértice v da árvore existe uma lista de arestas adjacentes a v.

▷ Esta representação já transforma a árvore no grafo direcionado.

Representação da árvore e do grafo direcionado

Entrada:

$\triangleright$ n: número de vértices de T

$\triangleright$ $inicio[i]$: ponteiro para o início da lista de arestas adjacentes ao vértice.

$\triangleright$ $prox[(i,j)]$: ponteiro para a aresta seguinte à aresta (i,j) na lista de arestas adjacentes ao vértice i. Se (i,j) é a última aresta da lista, $prox[(i,j)] = nil$.

$\triangleright$ $reverso[(i,j)]$: ponteiro para (j,i), a aresta reversa de (i,j), na lista de arestas adjacentes ao vértice j.

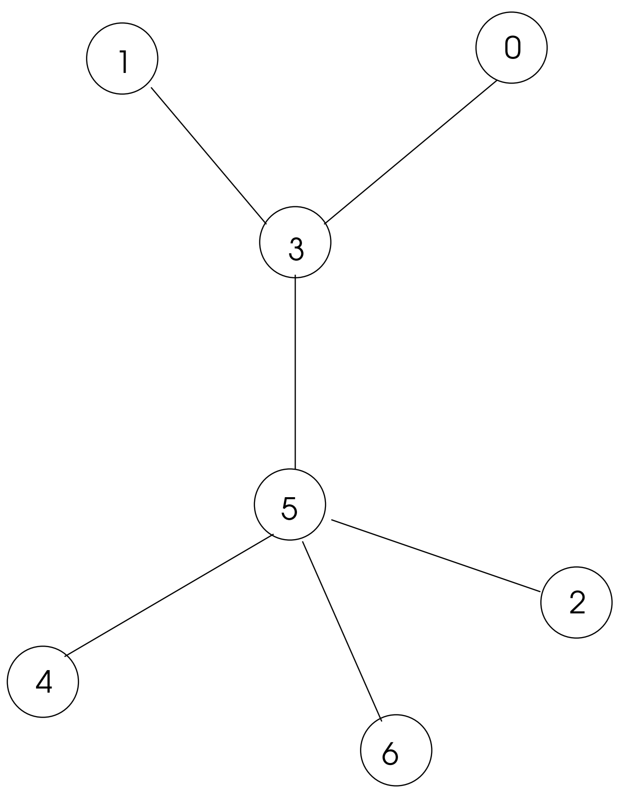
Representação da árvore e do grafo direcionado

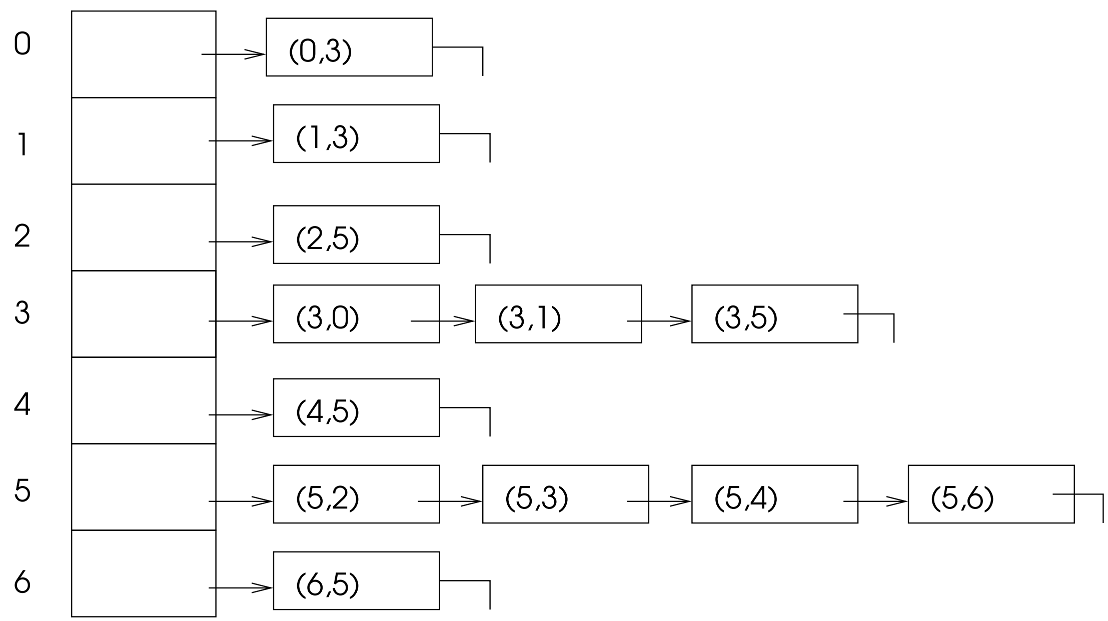
Saída:

$\triangleright$ $proxCircuito[(i,j)]$: ponteiro para a aresta seguinte a aresta (i,j), no circuito de Euler

Representação da árvore e do grafo direcionado

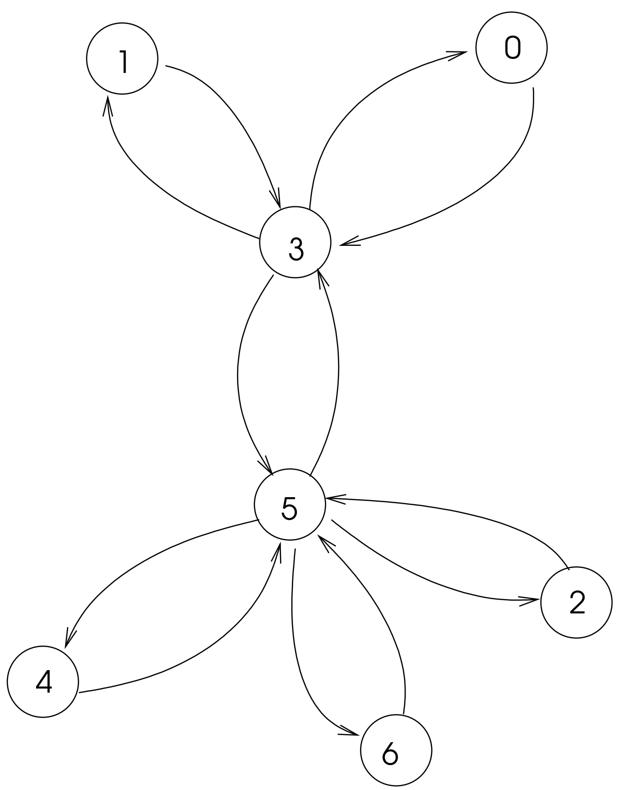
Exemplo: Árvore T, $n = 7$

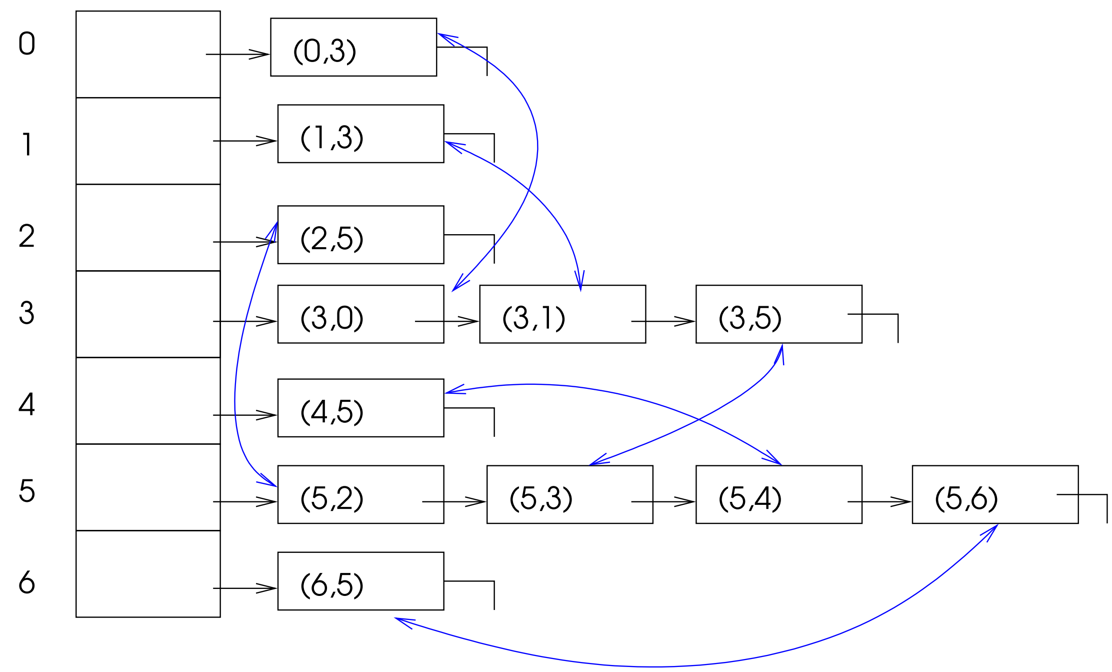




Representação da árvore e do grafo direcionado

Exemplo: Grafo Direcionado D, $n = 7$ e o Reverso





Obtenção de circuitos de Euler em árvores

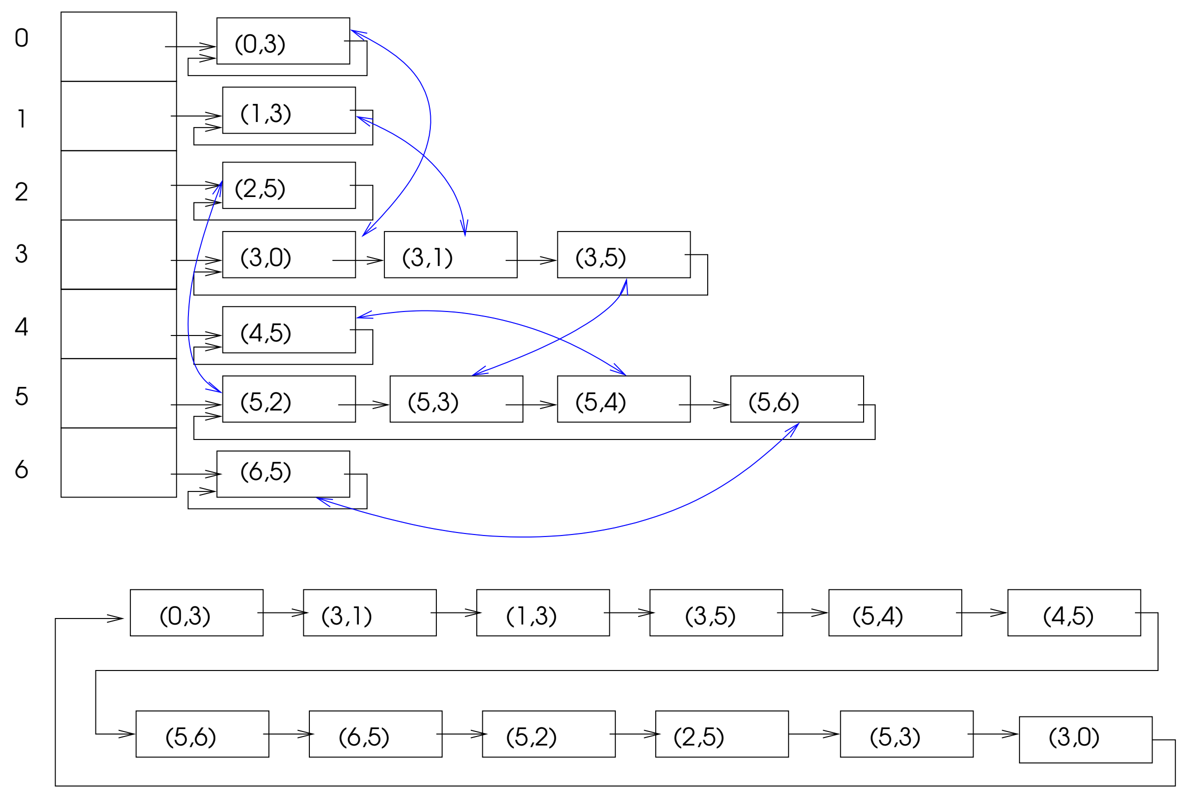
Algoritmo

para cada aresta direcionada (i, j) **faça em paralelo**
  se $prox[(i, j)] = nil$ **então**
    $prox[(i, j)] := inicio[i]$

para cada aresta direcionada (i, j) **faça em paralelo**
  $proxCircuito[(i, j)] = prox[reverso(i, j)]$

Obtenção de circuitos de Euler em árvores

$proxCircuito$



Obtenção de circuitos de Euler em árvores

Submodelo e complexidades:

Submodelo: EREW

Complexidades

- ▷ Tempo: $O(1)$
- ▷ Processador: número de arestas do grafo direcionado, $O(n)$
- ▷ É eficiente.

Obs.: Assumimos que o reverso é dado.

Obtenção de circuitos de Euler em árvores

▷ O algoritmo contrói uma lista circular representando o circuito de Euler. Quebrando esta lista em qualquer posição, obtemos uma ordenação das arestas dentro do circuito.

▷ O circuito de Euler fornece o percurso de uma busca em profundidade na árvore.

▷ Dado o circuito, conseguimos realizar várias operações sobre a árvore.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Orientação de uma árvore (determinar o vértice pai de cada vértice)

$\triangleright$ Dada uma árvore T, e um vértice escolhido para ser raiz de T, deseja-se orientar T das folhas para a raiz (transformá-la em uma "*in-tree*". Para isso, determinamos o vértice pai de cada vértice de T.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Idéia:

a) Obter circuito de Euler de T

b) Quebrar o circuito na raiz

c) Numerar as arestas dos circuitos (usando duplicação recursiva)

d) Determinar pai usando esta numeração.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Entrada:

$\triangleright$ $inicio[i]$: ponteiro para o início da lista de arestas adjacentes ao vértice.

$\triangleright$ $prox[(i,j)]$: ponteiro para a aresta seguinte à aresta (i,j) na lista de arestas adjacentes ao vértice i. Se (i,j) é a última aresta da lista, $prox[(i,j)] = nil$.

$\triangleright$ $reverso[(i,j)]$: ponteiro para (j,i), a aresta reversa de (i,j), na lista de arestas adjacentes ao vértice j.

r: vértice escolhido para ser a raiz de T.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Estruturas auxiliares:

▷ $proxCircuito[(i,j)]$: ponteiro para a aresta seguinte a aresta (i,j), no circuito de Euler

▷ $p[(i,j)]$: inicialmente terá cópia de $proxCircuito[(i,j)]$

▷ $dist[(i,j)]$: numeração da aresta (i,j) na lista $proxCircuito$.

Saída:

▷ $pai[i]$: o pai de cada vértice de T

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Passo (a): já visto

para cada aresta direcionada (i, j) **faça em paralelo**
se $prox[(i, j)] = nil$ **então**
$prox[(i, j)] := inicio[i]$

para cada aresta direcionada (i, j) **faça em paralelo**
$proxCircuito[(i, j)] := prox[reverso(i, j)]$

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Passo (b):

> **para** cada aresta direcionada (i, j) **faça em paralelo**
>   **se** $prox[(i, j)] = inicio[r]$ **então**
> $proxCircuito[reverso(i, j)] := nil$

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Obs: Agora $proxCircuito$ forma uma lista encadeada aberta de aresta, sendo que a primeira aresta é da forma $(r, -)$. Esta lista representa o percurso de uma busca em profundidade em T, partindo de r.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Passo (c):

para cada aresta direcionada (i, j) **faça em paralelo**

$dist[(i, j)] := 1$

$p[(i, j)] := proxCircuito[(i, j)]$

para cada aresta direcionada (i, j) **faça em paralelo**

enquanto $p[(i, j)] \neq nil$ **faça**

$dist[(i, j)] := dist[(i, j)] + dist[p(i, j)]$

$p[(i, j)] := p[p(i, j)]$

$dist[(i, j)] := E(D) - dist[(i, j)] + 1$

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Obs.: $E(D) = 2(n-1)$

$\triangleright$ Número de arestas na lista $proxCircuito$ é $E(D)$.

$\triangleright$ Estamos numerando as arestas na lista $proxCircuito$ de 1 a $E(D)$, do início para o fim da lista.

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Passo (d):

para cada aresta direcionada (i, j) **faça em paralelo**
  se $dist[(i, j)] < dist[reverso[(i, j)]]$ **então**
    $pai[j] := i$

  $pai[r] := -1$

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

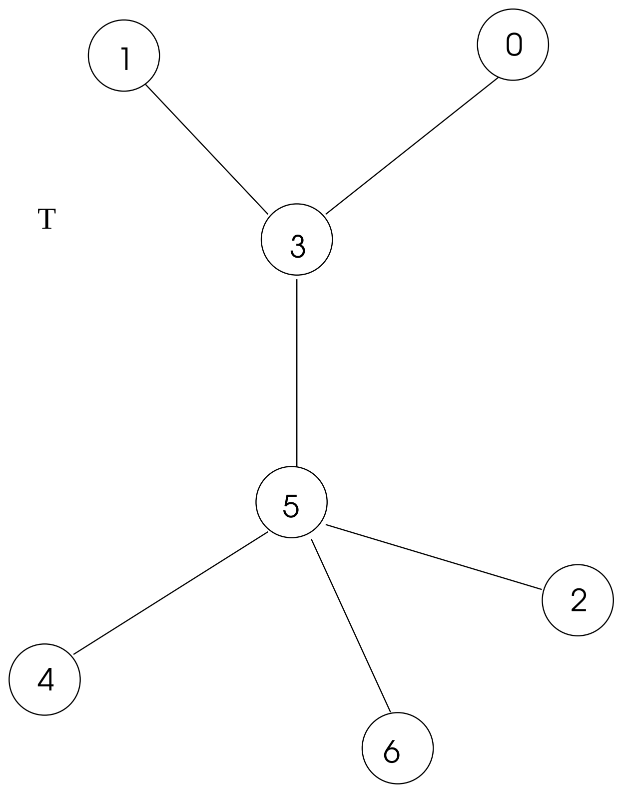
Obs.: Usando o percurso da busca em profundidade, determinamos se cada aresta é de avanço ou recuo, baseado na numeração. No percurso passamos sempre na aresta de avanço antes de passar na aresta de recuo reversa. Logo a aresta de avanço terá uma numeração menor do que a de recuo.

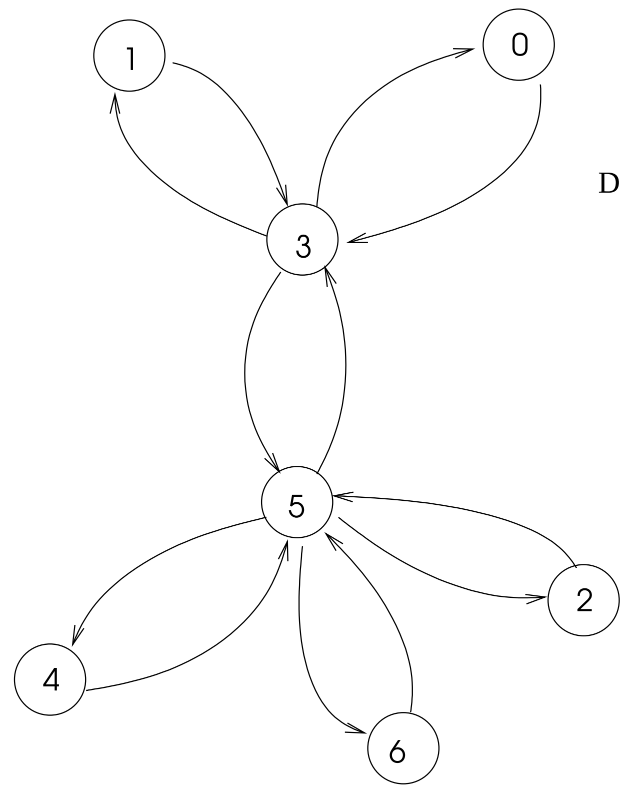
Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Obs.:

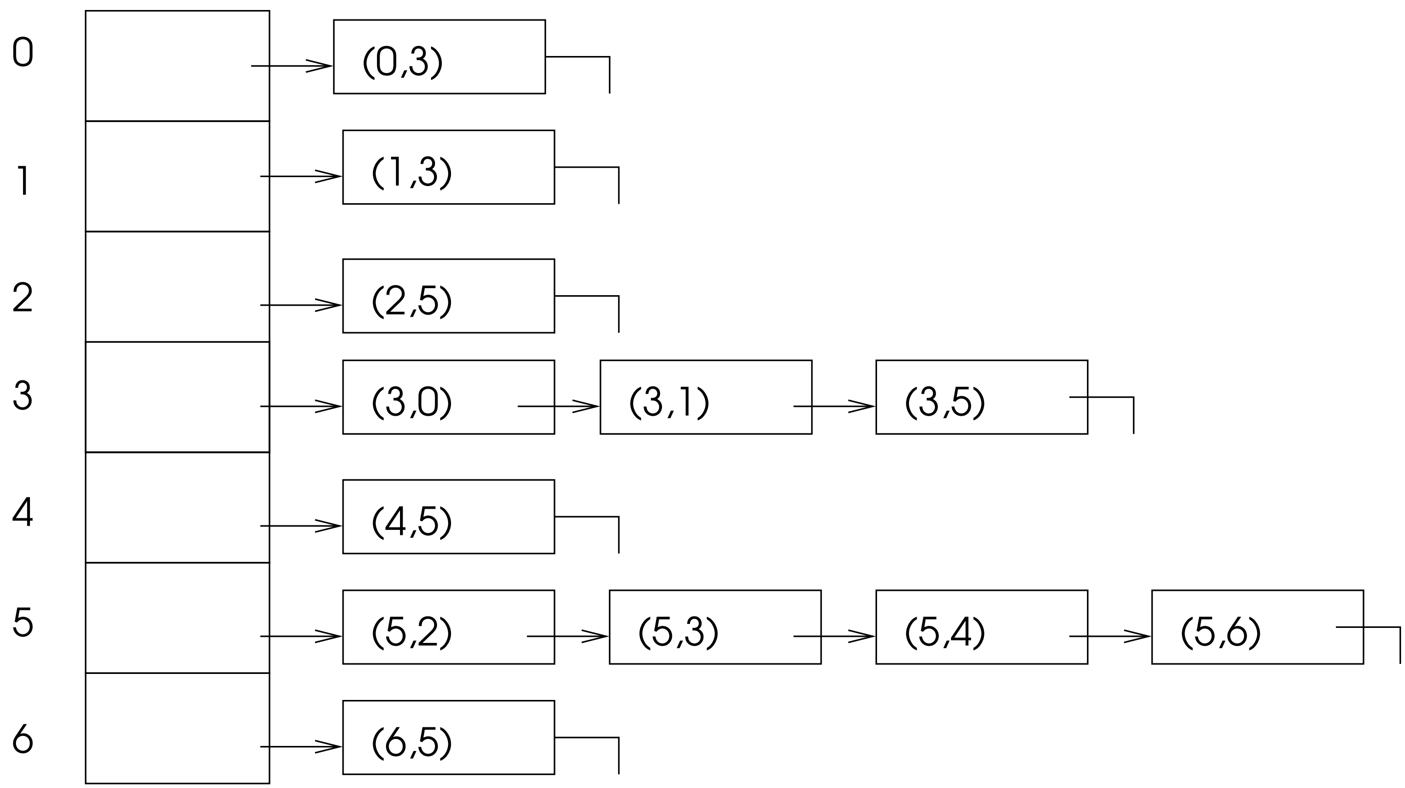
se $dist[(i,j)] < dist[reverso[(i,j)]]$ **então**
    $aresta(i,j)$ é de avanço
senão
    $aresta(i,j)$ é de recuo

Ex.: $n = 7$, $r = 0$

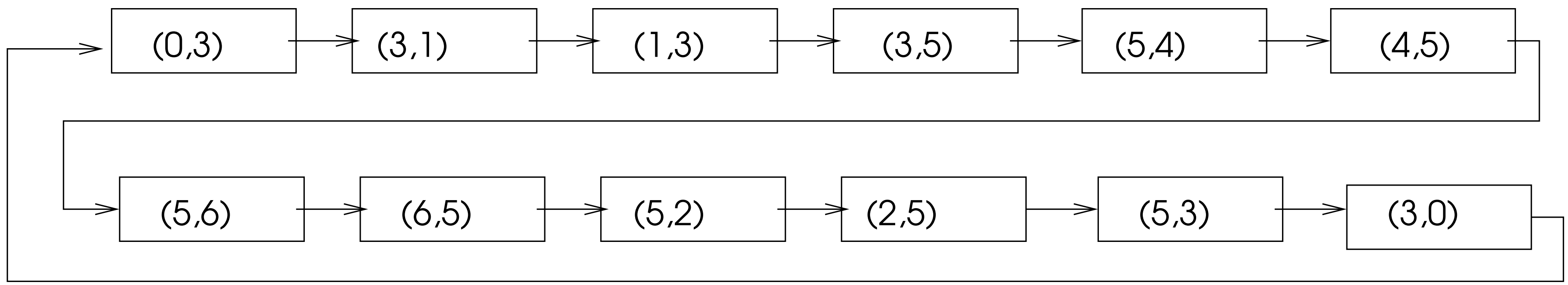




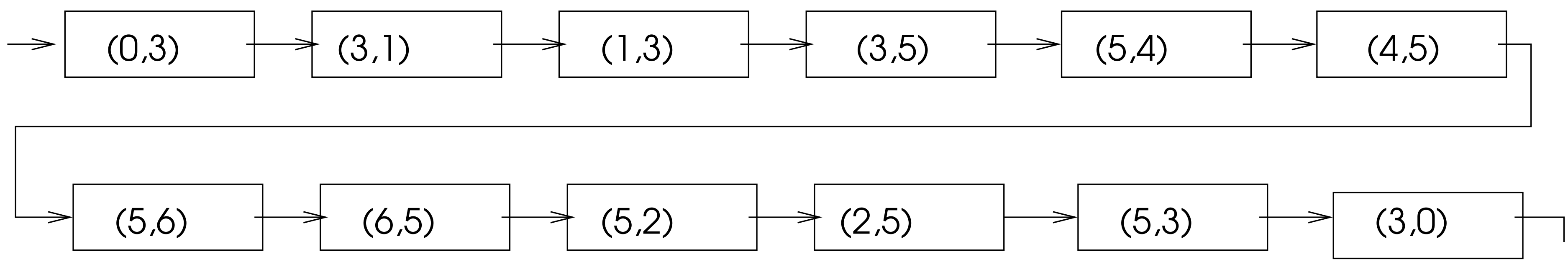
Ex.: Início



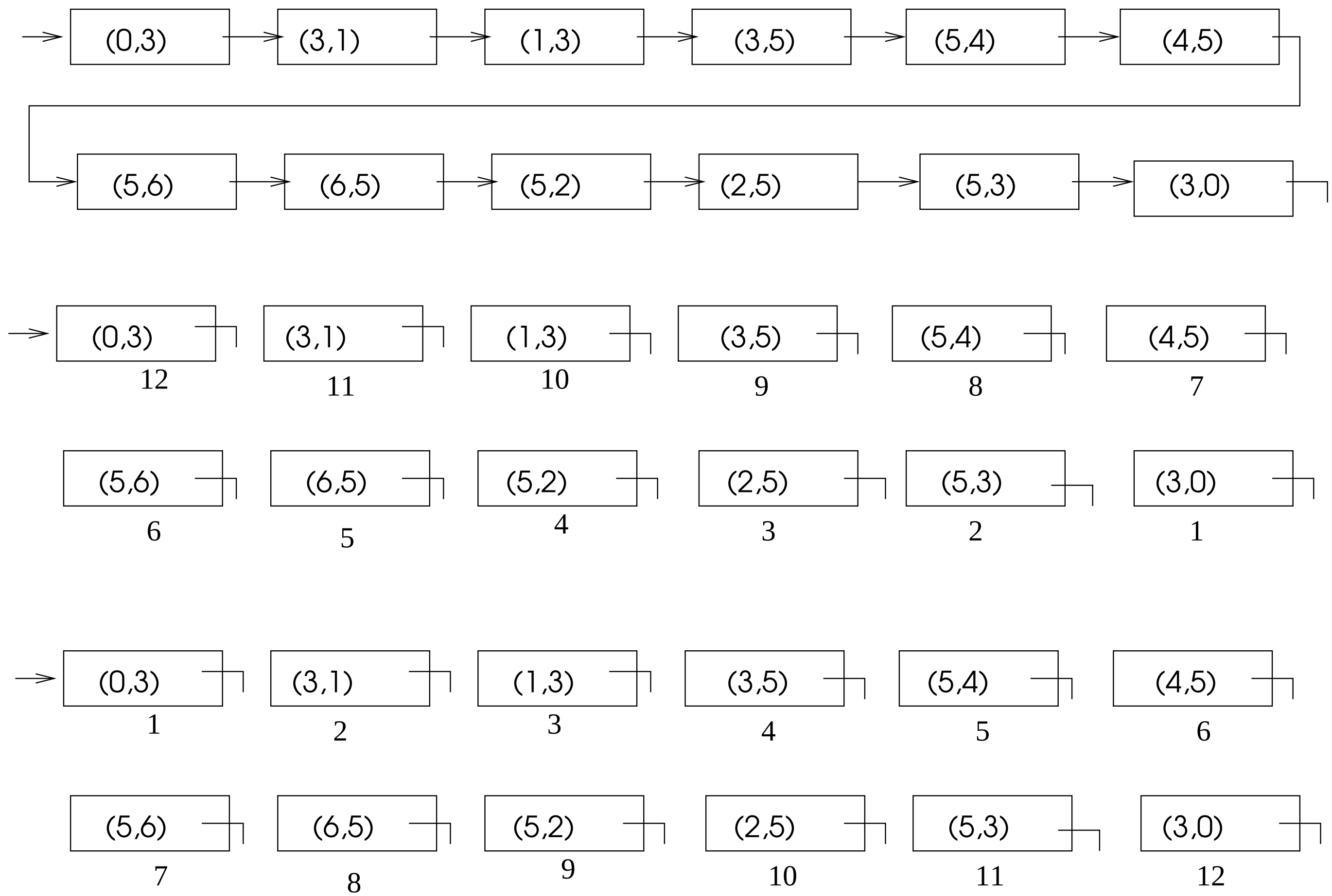
Passo (a)



Passo (b)

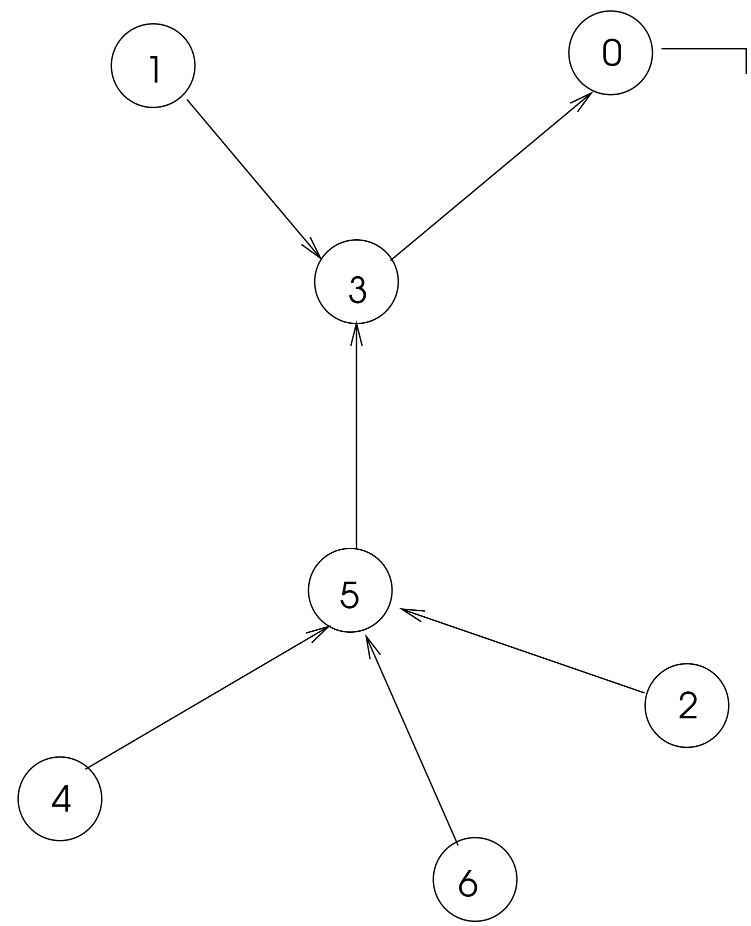


Passo (c)


(0,3)
(3,1)
(1,3)
(3,5)
(5,4)
(4,5)
(5,6)
(6,5)
(5,2)
(2,5)
(5,3)
(3,0)
(0,3) 12
(3,1) 11
(1,3) 10
(3,5) 9
(5,4) 8
(4,5) 7
(5,6) 6
(6,5) 5
(5,2) 4
(2,5) 3
(5,3) 2
(3,0) 1
(0,3) 1
(3,1) 2
(1,3) 3
(3,5) 4
(5,4) 5
(4,5) 6
(5,6) 7
(6,5) 8
(5,2) 9
(2,5) 10
(5,3) 11
(3,0) 12

Passo (d)

Pai	−1	3	5	0	5	3	5
	0	1	2	3	4	5	6



Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

Submodelo e complexidades:

Submodelo: CREW (leitura concorrente em n no passo (c))

Complexidades

- ▷ Passo (a): EREW, $t = O(1)$, $p = O(n)$
- ▷ Passo (b): EREW, $t = O(1)$, $p = O(n)$
- ▷ Passo (c): CREW, $t = O(\log n)$, $p = O(n)$
- ▷ Passo (d): EREW, $t = O(1)$, $p = O(n)$

Algoritmos em árvore usando o circuito de Euler (e duplicação recursiva)

1) Orientação de uma árvore
2) Determinar o número de descendentes de cada vértice

Determinar o número de descendentes de cada vértice

Determinar o número de descendentes de cada vértice

Dada uma árvore T enraizada, determinar para cada vértice i de T, o número de descendentes de i.

Idéia:

- ▷ a) Obter circuito de Euler de T
- ▷ b) Quebrar o circuito na raiz
- ▷ c) Numerar arestas do circuito
- ▷ d) Determinar o pai de cada vértice
- ▷ e) Determinar o número de descendentes de cada vértice, usando a numeração das arestas e pai.

Determinar o número de descendentes de cada vértice

Entrada:

$\triangleright$ n: número de vértices de T

$\triangleright$ $inicio[i]$: ponteiro para o início da lista de arestas adjacentes ao vértice.

$\triangleright$ $prox[(i,j)]$: ponteiro para a aresta seguinte à aresta (i,j) na lista de arestas adjacentes ao vértice i. Se (i,j) é a última aresta da lista, $prox[(i,j)] = nil$.

$\triangleright$ $reverso[(i,j)]$: ponteiro para (j,i), a aresta reversa de (i,j), na lista de arestas adjacentes ao vértice j.

$\triangleright$ r: vértice escolhido para ser a raiz de T.

Determinar o número de descendentes de cada vértice

Estruturas auxiliares:

$\triangleright$ $proxCircuito[(i,j)]$: ponteiro para a aresta seguinte a aresta (i,j), no circuito de Euler

$\triangleright$ $p[(i,j)]$: inicialmente terá cópia de $proxCircuito[(i,j)]$

$\triangleright$ $dist[(i,j)]$: numeração da aresta (i,j) na lista $proxCircuito$.

$\triangleright$ $pai[i]$: o pai de cada vértice de T

Saída:

$\triangleright$ $ND[i]$: número de descendentes do vértice i

Determinar o número de descendentes de cada vértice

Passos (a), (b), (c), (d), já vimos

Passo (e)

para cada aresta direcionada (i, j) **faça em paralelo**
  se $j = pai[i]$ **então**
    $\triangleright$ (i, j) é aresta de recuo e (j, i) é aresta de avanço

$$ND[i] := \frac{(dist[(i, j)] - dist[reverso[(i, j)]] - 1)}{2}$$

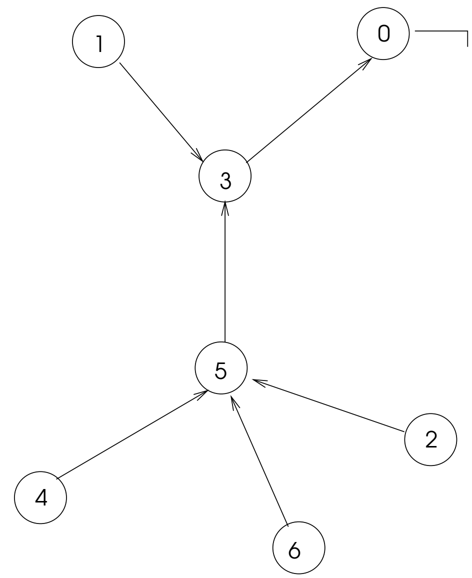
$ND[r] := n - 1$

Pai

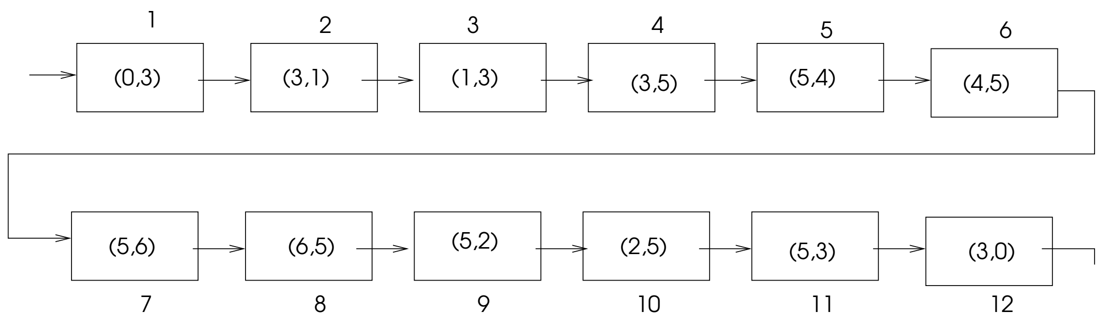
Pai	−1	3	5	0	5	3	5
	0	1	2	3	4	5	6

ND

ND	6	0	0	5	0	3	0
	0	1	2	3	4	5	6



Circuito:



Determinar o número de descendentes de cada vértice

Submodelo: CREW

Complexidades:

- ▷ Tempo: $O(\log n)$
- ▷ Processadores: $O(n)$

Obter numeração pré-ordem dos vértices

Obter numeração pré-ordem dos vértices

Dada uma árvore T, determinar para cada vértice i de T, a ordem de i no percurso em pré-ordem (ou pós-ordem) de T, enraizada.

Numeração pré-ordem:

- ▷ numera raiz
- ▷ numera as subárvores em pré-ordem

Obter numeração pré-ordem dos vértices

Idéia

▷ Passos (a), (b), (c), já vimos

▷ (d) Classificar arestas em avanço e recuo, usando a numeração obtida em (c)

▷ (e) Numerar as arestas de avanço do circuito, usando a duplicação recursiva

▷ (f) Determinar numeração pré-ordem dos vértices usando numeração obtida em (e)

Obter numeração pré-ordem dos vértices

Entrada:

$\triangleright$ n: número de vértices de T

$\triangleright$ $inicio[i]$: ponteiro para o início da lista de arestas adjacentes ao vértice.

$\triangleright$ $prox[(i,j)]$: ponteiro para a aresta seguinte à aresta (i,j) na lista de arestas adjacentes ao vértice i. Se (i,j) é a última aresta da lista, $prox[(i,j)] = nil$.

$\triangleright$ $reverso[(i,j)]$: ponteiro para (j,i), a aresta reversa de (i,j), na lista de arestas adjacentes ao vértice j.

$\triangleright$ r: vértice escolhido para ser a raiz de T.

Obter numeração pré-ordem dos vértices

Estruturas auxiliares:

$\triangleright$ $proxCircuito[(i,j)]$: ponteiro para a aresta seguinte a aresta (i,j), no circuito de Euler

$\triangleright$ $p[(i,j)]$: inicialmente terá cópia de $proxCircuito[(i,j)]$

$\triangleright$ $dist[(i,j)]$: numeração da aresta (i,j) na lista $proxCircuito$.

$\triangleright$ $avanco[(i,j)]$: Vetor de valores lógicos que indicará se a aresta é de avanço ou de recuo

Saída:

$\triangleright$ $preOrdem[i]$: numeração pré-ordem do vértice i

Obter numeração pré-ordem dos vértices

Passos (a), (b), (c), já vimos

Passo (d)

para cada aresta direcionada (i, j) **faça em paralelo**
  se $dist[(i, j)] < dist[reverso[(i, j)]]$ **então**
    $avanco[(i, j)] := true$
  senão
    $avanco[(i, j)] := false$

Passo (e)

para cada aresta direcionada (i, j) **faça em paralelo**
  se $avanco[(i, j)]$ **então**
    $dist[(i, j)] := 1$
  senão
    $dist[(i, j)] := 0$

  $p[(i, j)] := proxCircuito[(i, j)]$
  enquanto $p[(i, j)] \neq nil$ **faça**
    $dist[(i, j)] := dist[(i, j)] + dist[p(i, j)]$
    $p[(i, j)] := p[p[(i, j)]]$

  $$dist[(i, j)] := \frac{E(D)}{2} - dist[(i, j)] + 1$$

Passo (f)

para cada aresta direcionada (i, j) **faça em paralelo**
  se $avanco[(i, j)]$ **então**
    $preOrdem[j] := dist[(i, j)]$

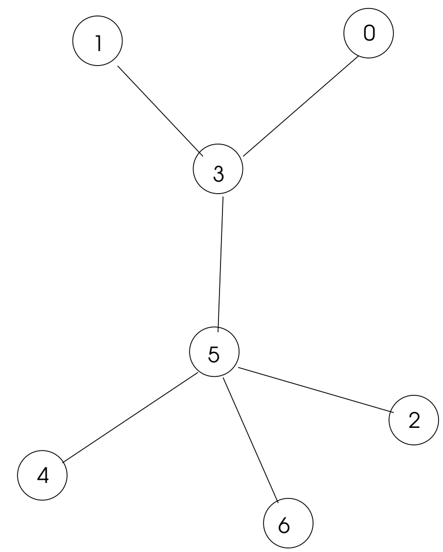
  $preOrdem[r] := 0$

avanço

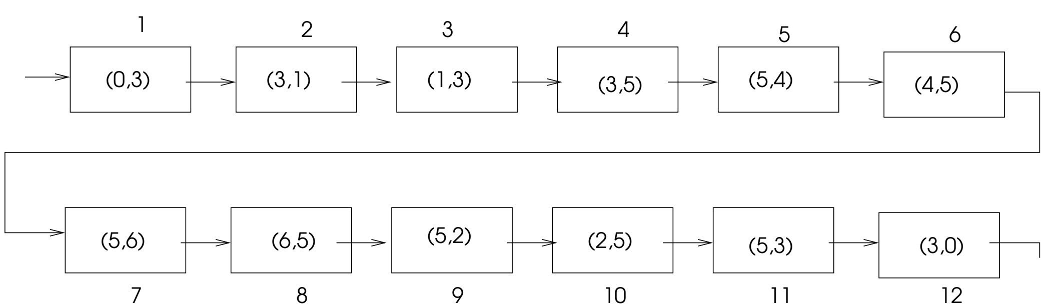
T	T	F	T	T	F	T	F	T	F	F	F
1	2	3	4	5	6	7	8	9	10	11	12

dist

1	1	0	1	1	0	1	0	1	0	0	0
1	2	3	4	5	6	7	8	9	10	11	12



Circuito:



	1	2	3	4	5	6	7	8	9	10	11	12
avanço	T	T	F	T	T	F	T	F	T	F	F	F

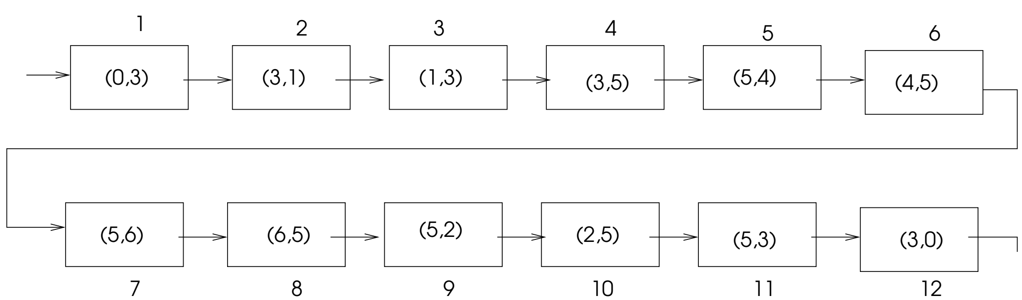
	1	2	3	4	5	6	7	8	9	10	11	12
dist	1	1	0	1	1	0	1	0	1	0	0	0

Duplicação Recursiva

	1	2	3	4	5	6	7	8	9	10	11	12
dist	6	5	4	4	3	2	2	1	1	0	0	0

	1	2	3	4	5	6	7	8	9	10	11	12
dist	1	2	3	3	4	5	5	6	6	7	7	7

Circuito:

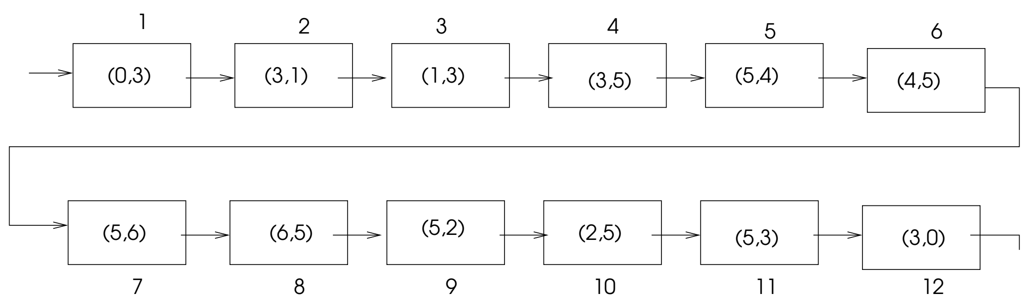


avanço	T	T	F	T	T	F	T	F	T	F	F	F
	1	2	3	4	5	6	7	8	9	10	11	12

dist	1	2	3	3	4	5	5	6	6	7	7	7
	1	2	3	4	5	6	7	8	9	10	11	12

preOrdem	0	2	6	1	4	3	5
	0	1	2	3	4	5	6

Circuito:



Obter numeração pré-ordem dos vértices

Submodelo: CREW

Complexidades:

- ▷ Tempo: $O(\log n)$
- ▷ Processadores: $O(n)$

Numeração pós-Ordem dos vértices

Observação:

Para obter numeração pós-ordem

- ▷ Marcar arestas de recuo
- ▷ Numerar arestas de recuo
- ▷ Obter numeração pós-Ordem

Numeração pós-Ordem dos vértices

Passo (f)

para cada aresta direcionada (i, j) **faça em paralelo**
  se $recuo[(i, j)]$ **então**
    $posOrdem[j] := dist[(i, j)] - 1$

$posOrdem[r] := n - 1$

Obter a relação é descendente entre vértices

Obter a relação é descendente entre vértices

Dada uma árvore T enraizada, determinar para cada par de vértices i, j de T, se i é descendente de j

- Idéia:
  - Determinar números de descendentes de cada vértice.
  - Determinar a numeração pré-ordem de cada vértice.
  - Determinar a relação é descendente para cada par de vértices, usando número de descendentes e pré-ordem.

Obter a relação é descendente entre vértices

Entrada:

$\triangleright$ n: número de vértices de T

$\triangleright$ $inicio[i]$: ponteiro para o início da lista de arestas adjacentes ao vértice.

$\triangleright$ $prox[(i,j)]$: ponteiro para a aresta seguinte à aresta (i,j) na lista de arestas adjacentes ao vértice i. Se (i,j) é a última aresta da lista, $prox[(i,j)] = nil$.

$\triangleright$ $reverso[(i,j)]$: ponteiro para (j,i), a aresta reversa de (i,j), na lista de arestas adjacentes ao vértice j.

$\triangleright$ r: vértice escolhido para ser a raiz de T.

Obter a relação é descendente entre vértices

Estruturas auxiliares:

$\triangleright$ $proxCircuito[(i,j)]$: ponteiro para a aresta seguinte a aresta (i,j), no circuito de Euler

$\triangleright$ $p[(i,j)]$: inicialmente terá cópia de $proxCircuito[(i,j)]$

$\triangleright$ $dist[(i,j)]$: numeração da aresta (i,j) na lista $proxCircuito$.

$\triangleright$ $avanco[(i,j)]$: Vetor de valores lógicos que indicará se a aresta é de avanço ou de recuo

$\triangleright$ $ND[i]$: número de descendentes do vértice vértices i

$\triangleright$ $preOrdem[i]$: numeração pré-ordem do vértice i

Saída:

$\triangleright$ $ehDescendente[i,j]$: será 1 se i for descendente de j. Senão será 0.

Obter a relação é descendente entre vértices

1) Obtem número de descendentes: já visto

2) Obtem a pré-ordem: já visto

3) Obter a relação é descendente

para $0 \leq i, j \leq n-1$ **faça em paralelo**
  se $preOrdem[j] < preOrdem[i]$ e
    $preOrdem[i] \leq preOrdem[j] + ND[j]$ **então**
      $ehDescendente[i, j] = 1$
  senão
      $ehDescendente[i, j] = 0$

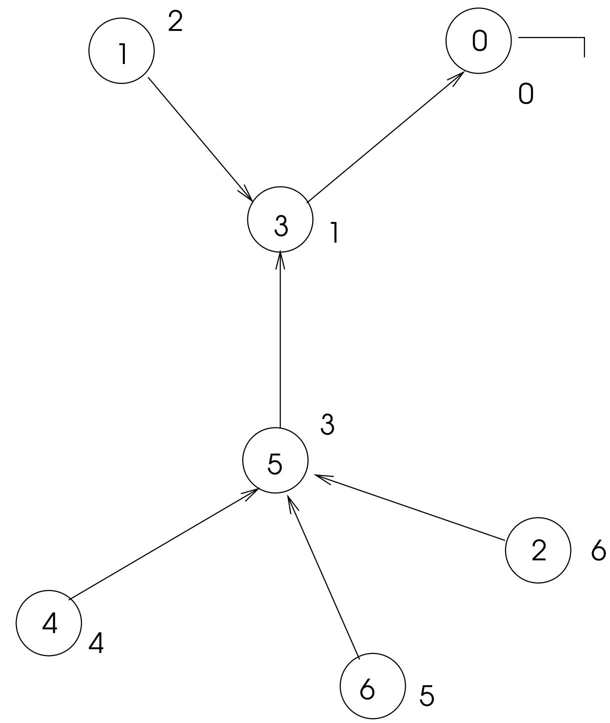
PreOrdem

0	1	2	3	4	5	6
0	2	6	1	4	3	5

ND

0	1	2	3	4	5	6
6	0	0	5	0	3	0

i \ j	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	1	0	0				
3				0			
4		0			0		
5						0	
6							0



Obter a relação é descendente entre vértices

Submodelo: CREW

Complexidades:

- ▷ Tempo: $O(\log n)$
- ▷ Processadores: $O(n^2)$

Fim