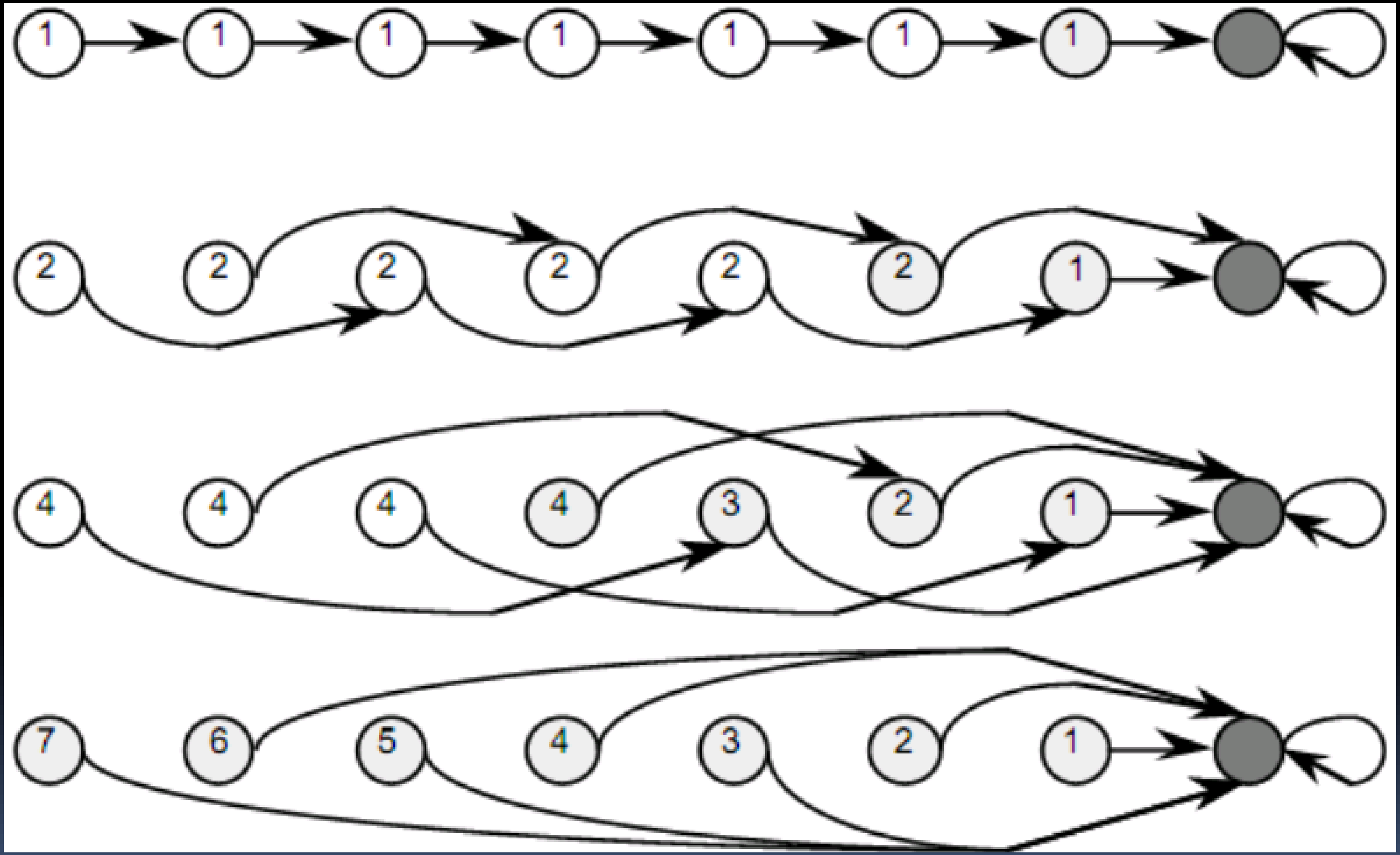
BÀI 8. LIST & TREE

NỘI DUNG BÀI HỌC

- Kỹ thuật con trỏ nhảy:
- Chu trình Euler
- Các bài toán trên cây

1. KỸ THUẬT CON TRỎ NHẢY

KỸ THUẬT CON TRỎ NHẢY



KỸ THUẬT CON TRỎ NHẢY

- Danh sách được định nghĩa bởi 2 mảng:
  - Mảng giá trị A[1..n]
  - Mảng địa chỉ L[1..n], L[i] là chỉ số phần tử kế tiếp mà A[i] trỏ đến.
  - Phần tử cuối là phần tử trỏ đến chính nó

```
for k = 1 to log2n do
      // forall nodes i do in parallel
      for i = 1 to n do in parallel
            do_something;
            L[i] = L[L[i]];
      end parallel
end for.
```

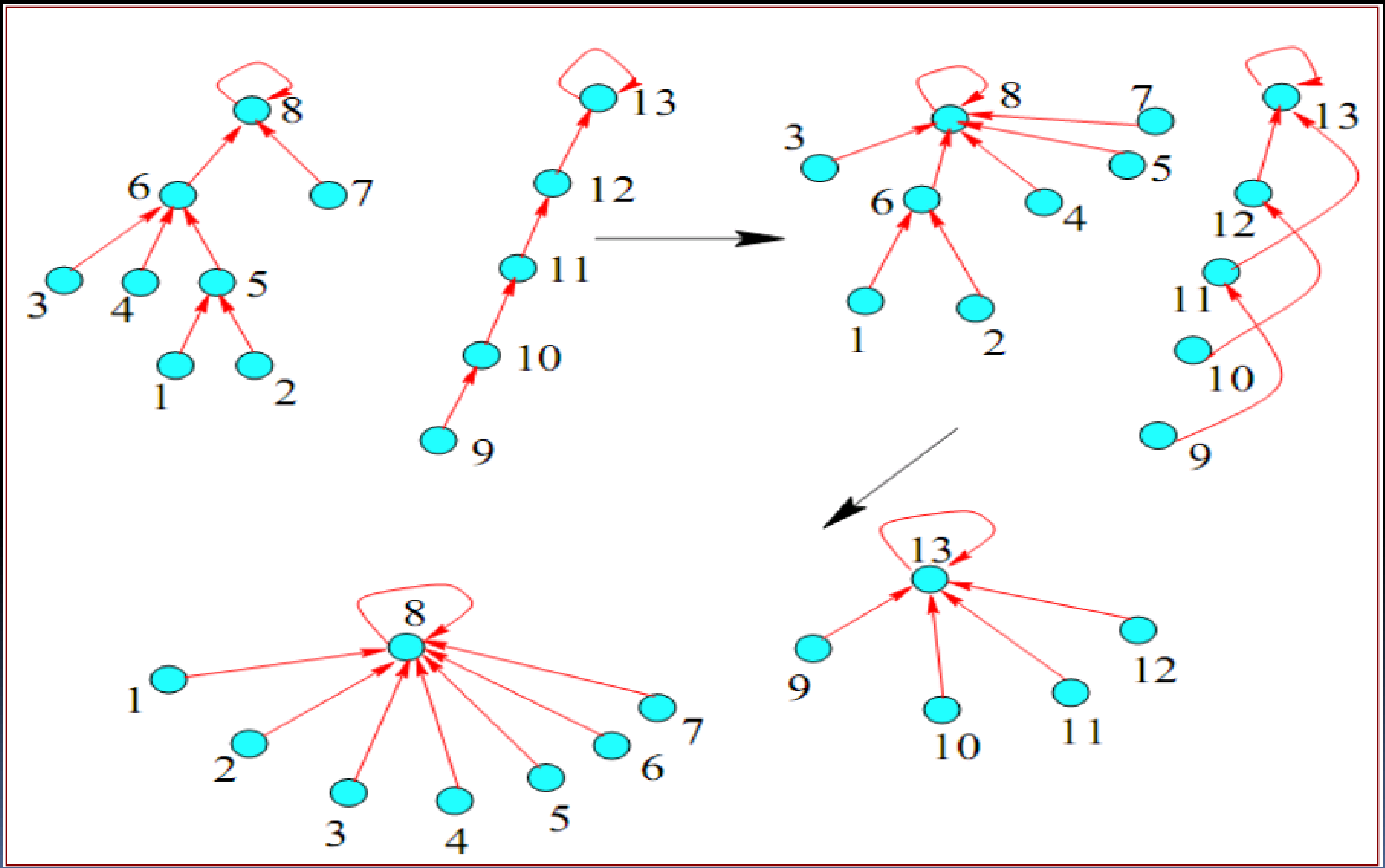
ROOTED-DIRECTED TREE

- Định nghĩa: Rooted-directed Tree (cây hướng gốc) T là một đồ thị có hướng với 1 nút đặc biệt r thỏa mãn:
  - outdegree(r)=0
  - outdegree(v)=1, $\forall v \in V \setminus \{r\}$
  - $\forall v \in V \setminus \{r\}$: ∃ 1 đường đi từ v tới r
- r được gọi là gốc của cây
- Biểu diễn cây T theo mảng P[1..n]
  - P[i] = j nếu j là cha của i trên cây
  - Gốc là nút trỏ đến chính nó: P[r]=r

XÁC ĐỊNH GỐC CÂY TRONG RỪNG

- Phát biểu bài toán:
  - Gọi F là rừng các cây hướng gốc
  - F được biểu diễn thông qua mảng P[1..n]
  - Với mỗi nút i trong rừng, hãy xác định gốc của cây chứa nút i, ta gọi là S[i]
- Cách tiếp cận:
  - Sử dụng kỹ thuật con trỏ nhảy

VÍ DỤ GỐC CÂY TRONG RỪNG



XÁC ĐỊNH GỐC CÂY TRONG RỪNG

```
input   : rừng F xác định bởi P[1..n]
output : S[1..n], S[i] -- gốc của cây con chứa nút i
begin
      for i = 1 to n do in parallel
            S[i] = P[i];
            while S[i] <> S[S[i]] do
                  S[i]  = S[S[i]];
            end while.
      end for.
end.
```

pointer jumping

BÀI TOÁN SUFFIX SUM TRÊN CÂY

- Phát biểu bài toán:
  - Rừng F biểu diễn bởi mảng P[1..n].
  - Các nút trên cây có trọng số là W[1..n]
  - Nút gốc cây có trọng số bằng 0.
  - Hãy xác định tổng trọng số đi từ nút v bất kỳ tới gốc ra của cây con chứa v.
- Cách tiếp cân:
  - Kỹ thuật con trỏ nhảy

BÀI TOÁN SUFFIX SUM TRÊN CÂY

```
input   : rừng F xác định bởi P[1..n], W[1..n]
output : R[1..n], R[i] -- trọng số đi từ i tới S[i]
begin
      for i = 1 to n do in parallel
            S[i] = P[i];
            while S[i] <> S[S[i]] do
                  W[i] = W[i] + W[S[i]];
                  S[i]  = S[S[i]];
            end while.
      end for.
end.
```

Mô hình CREW

2. CHU TRÌNH EULER

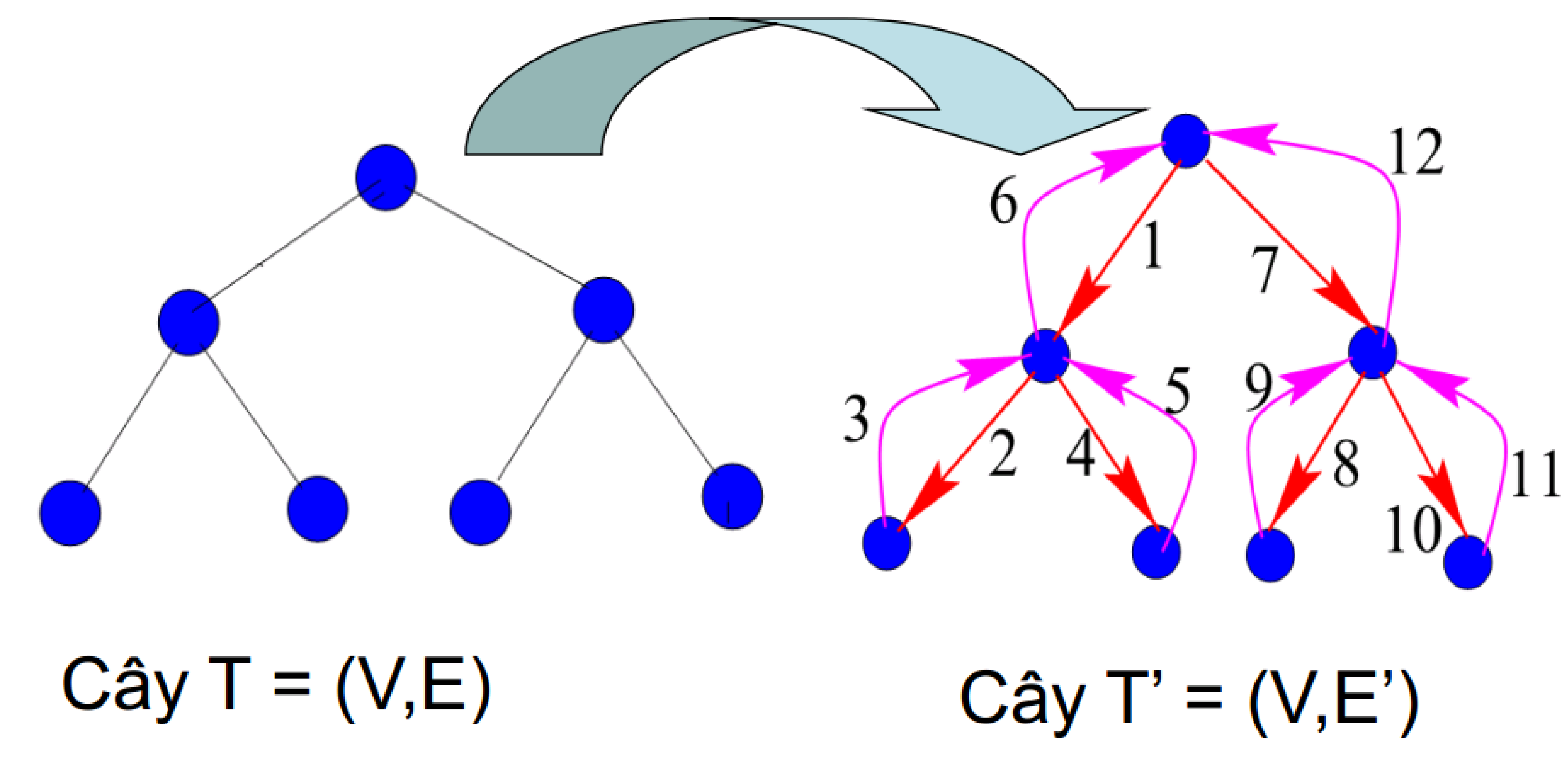
CHU TRÌNH EULER

- Nhắc lại khái niệm:
  - Chu trình Euler đi qua tất cả các cạnh của đồ thị 1 lần
  - Đối với đồ thị có hướng thì được gọi là chu trình có hướng
  - Những đồ thị thỏa mãn điều kiện trên được gọi là đồ thị Euler

CHU TRÌNH EULER

- Xét T=(V,E) là một cây vô hướng Mỗi cạnh vô hướng (u,v) thuộc E được chuyển thành 2 cạnh có hướng <u,v> và <v,u>
- Khi đó:
  - T’ = (V,E’) là 1 đồ thị có hướng
  - Outdegree(v) = Indegree(v)
  - Xác định chu trình Euler trên đồ thị T’

CHU TRÌNH EULER

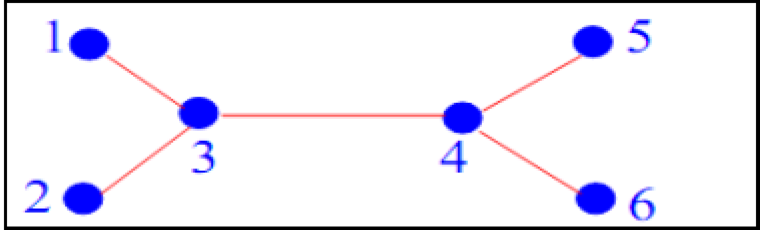


XÂY DỰNG CHU TRÌNH EULER

- Xây dựng hàm successor s: $E' \rightarrow E$
- Xác định danh sách kề với v; adj(v) = $<u_0, u_1, .. u_{d-1}>$ với d là bậc của v trong T.
- Hàm successor:

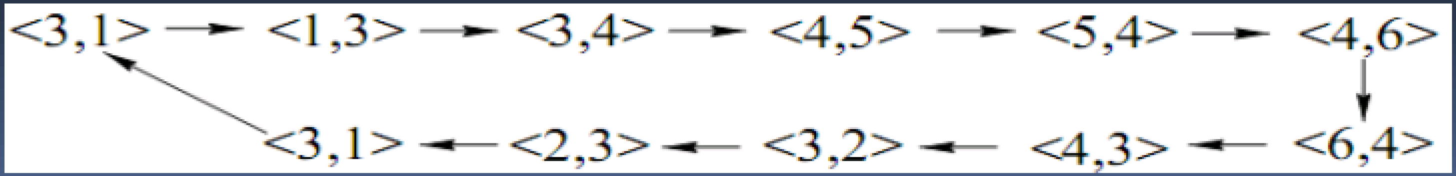
$$s(<u_i, v>) = <v, u_{(i+1) \bmod d}>, \forall i | 0 \leq i < d-1$$

CHU TRÌNH EULER



v	adj(v)
1	3
2	3
3	2, 1, 4
4	3, 5, 6
5	4
6	4

edge	successor
<3,1>	<1,3>
<3,2>	<2,3>
<2,3>	<3,1>
<1,3>	<3,4>
<4,3>	<3,2>
<3,4>	<4,5>
<5,4>	<4,6>
<6,4>	<4,3>
<4,5>	<5,4>
<4,6>	<6,4>



XÂY DỰNG CHU TRÌNH EULER

- Xét 1 nút u nằm trong L[v] bất kỳ :
  - Một con trỏ xác định u’ nằm sau u.
  - Một con trỏ dùng để xác định vị trí cặp cạnh ngược chiều <u,v> và <v,u> trong T’ từ 2 danh sách kề L[u] và L[v]

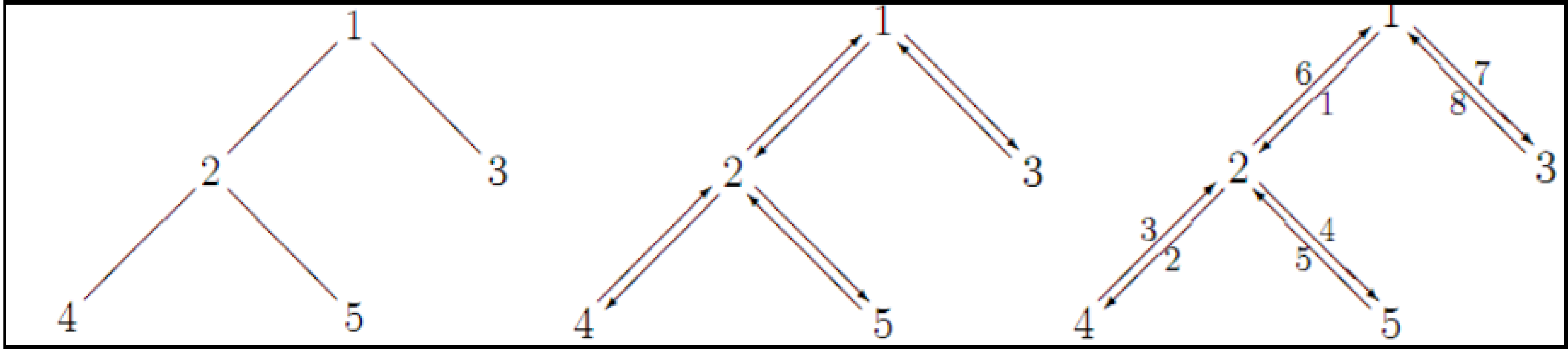
XÂY DỰNG CHU TRÌNH EULER

```
input   :  T = (V,E) biểu diễn bởi L(v).
output :  Euler tour : {next(e) vơi mọi e thuộc E}
begin
        for each edge <u,v> thuộc E
                next(<u,v>) = s(<u,v>);
        end parallel
end.
(s(e) := successor of reverse edge in adjacency list)
```

XÂY DỰNG CHU TRÌNH EULER

- Định lý: Cho cây T=(V,E):
  - Xác định bởi các danh sách kề của các đỉnh.
  - Mỗi nút của danh sách kề có 2 con trỏ
  - Chu trình Euler trong cây T’: O(1) đơn vị thời gian và O(n) thao tác thực hiện

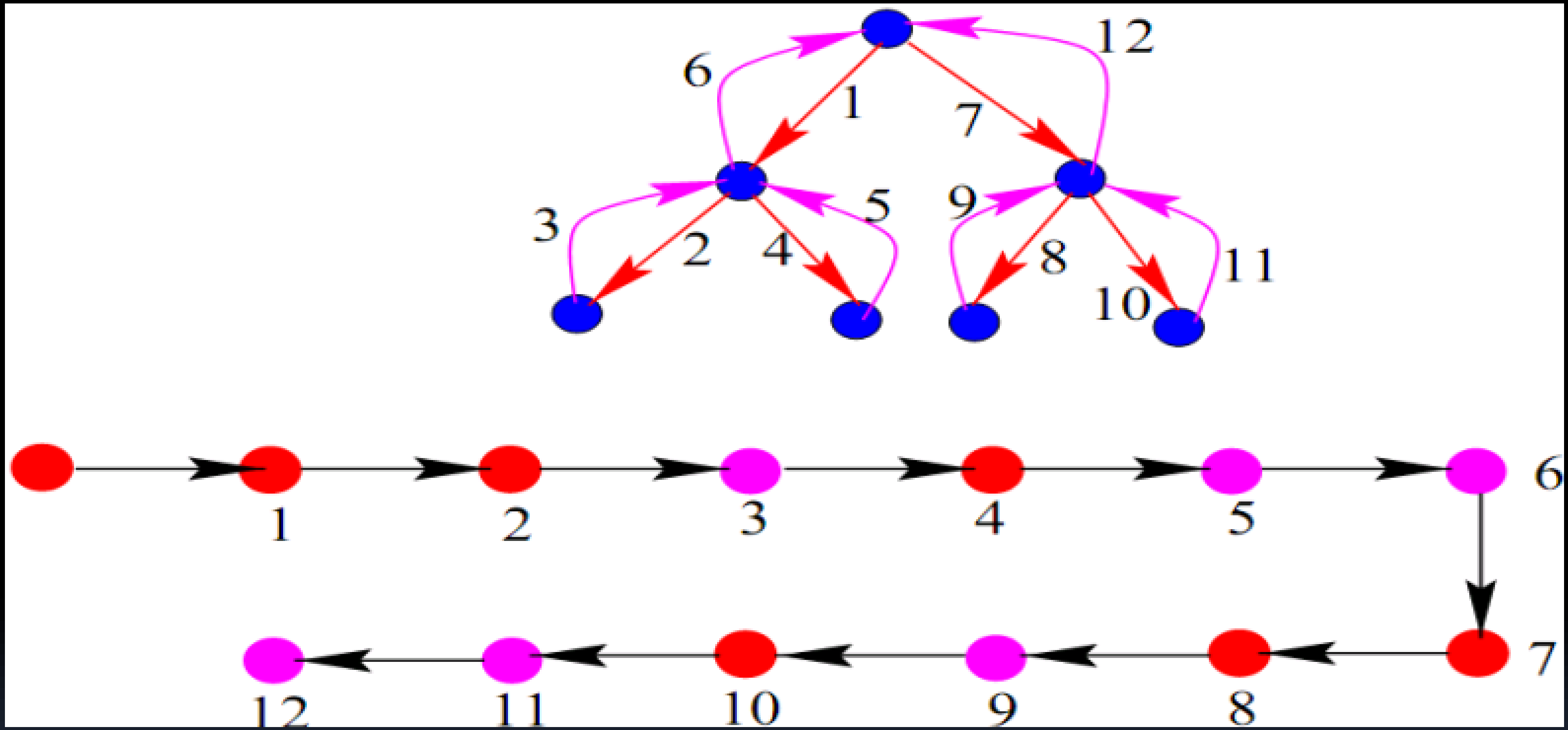
XÂY DỰNG CHU TRÌNH EULER



XÂY DỰNG CHU TRÌNH EULER

- Nếu cây có n đỉnh, ta có thể biểu diễn cây dưới dạng 1 danh sách 2n-2 nút.
- Với quy ước như sau: với mỗi đỉnh v ta gọi p(v) là cha của v. Khi đó:
  - Nút đỏ là cạnh có dạng <p(v),v>
  - Nút hồng là cạnh có dạng <v,p(v)>

XÂY DỰNG CHU TRÌNH EULER

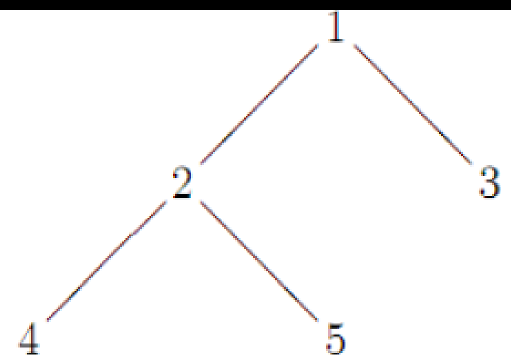


XÂY DỰNG CHU TRÌNH EULER

- Biểu diễn cây T, đồ thị T’ bởi danh sách liên kết
- Xây dựng chu trình Euler theo thuật toán đã có

- **Adjacency list:**

1	2	3	4	5
(1,2)	(2,1)	(3,1)	(4,2)	(5,2)
(1,3)	(2,4)			
	(2,5)			



- **Solution:**

edge	(1,2)	(1,3)	(2,1)	(2,4)	(2,5)	(3,1)	(4,2)	(5,2)
succ	(2,4)	(3,1)	(1,3)	(4,2)	(5,2)	(1,2)	(2,5)	(2,1)

Path: $1 \to 2 \to 4 \to 2 \to 5 \to 2 \to 1 \to 3 \to 1$

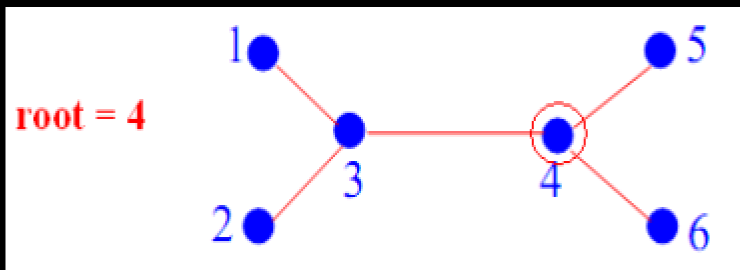
3. CÁC BÀI TOÁN TRÊN CÂY

3.1. ROOTED TREE

PHÁT BIỂU BÀI TOÁN

- Cho 1 cây T = (V,E) xác định bởi danh sách đỉnh kề và 1 nút r bất kỳ thuộc V. Hãy xây dựng cây T với gốc là r bằng cách: với mỗi nút $v \neq r$ xác định nút cha của v (đặt nút cha là p(v))
- Cách tiếp cận:
  - Thiết lập 1 chu trình Euler trên cây T’
  - Giả sử u là nút cuối cùng trong danh sách kề của nút r. Đặt s(<u.r>) = 0.
  - Đặt trọng số 1 cho các cạnh <x,y> trên T’ và thực hiện suffix_sum trên cây
  - Với mỗi <x,y> xác định x = p(y) nếu suffix_sum(<x,y>) lớn hơn suffix_sum(<y,x>)

VÍ DỤ ROOTED TREE (ROOT: 4)



v	adj(v)
1	3
2	3
3	2, 1, 4
4	3, 5, 6
5	4
6	4

edge	successor
<3,1>	<1,3>
<3,2>	<2,3>
<2,3>	<3,1>
<1,3>	<3,4>
<4,3>	<3,2>
<3,4>	<4,5>
<5,4>	<4,6>
<6,4>	null
<4,5>	<5,4>
<4,6>	<6,4>

VÍ DỤ ROOTED TREE (ROOT: 4)

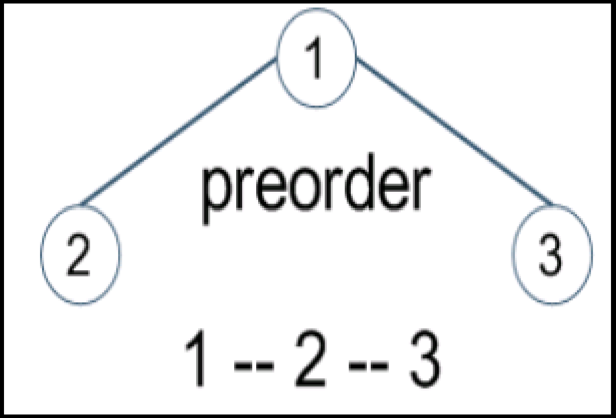
Euler Tour				Result	
Thứ tự	Cạnh	Giá trị	Suffix_sum	v	p(v)
1	<4,3>	1	10	1	3
2	<3,2>	1	9	2	3
3	<2,3>	1	8	3	4
4	<3,1>	1	7	4	4
5	<1,3>	1	6	5	4
6	<3,4>	1	5	6	4
7	<4,5>	1	4		
8	<5,4>	1	3		
9	<4,6>	1	2		
10	<6,4>	1	1		

3.2. DUYỆT CÂY

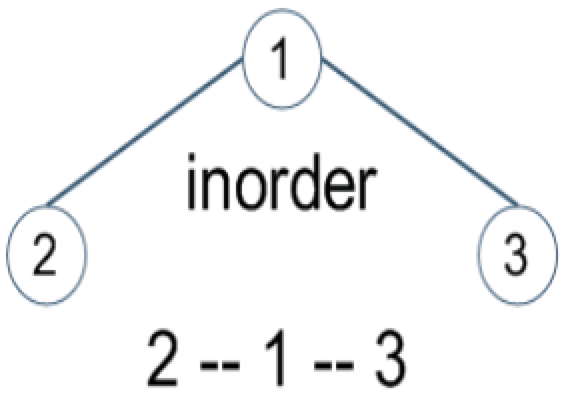
DUYỆT CÂY

- Qui tắc duyệt cây:
  - Đi qua lần lượt các nút của cây, mỗi nút 1 lần (tên nút hoặc giá trị chứa bên trong nút) theo thứ tự đi qua.
- Có 3 cách duyệt cây quan trọng;
  - Duyệt trước: Pre-Order
  - Duyệt giữa: In-Order
  - Duyệt sau: Post Order

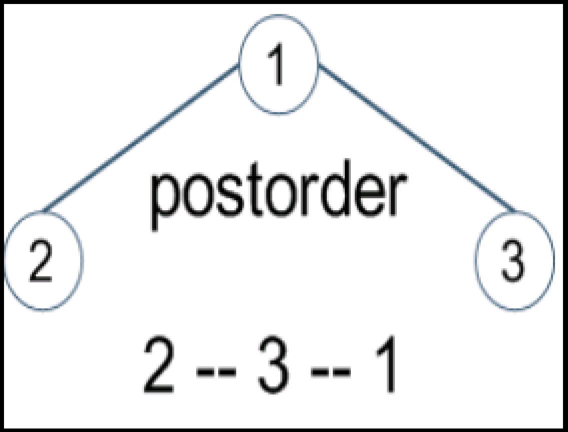
DUYỆT TRƯỚC PRE-ORDER



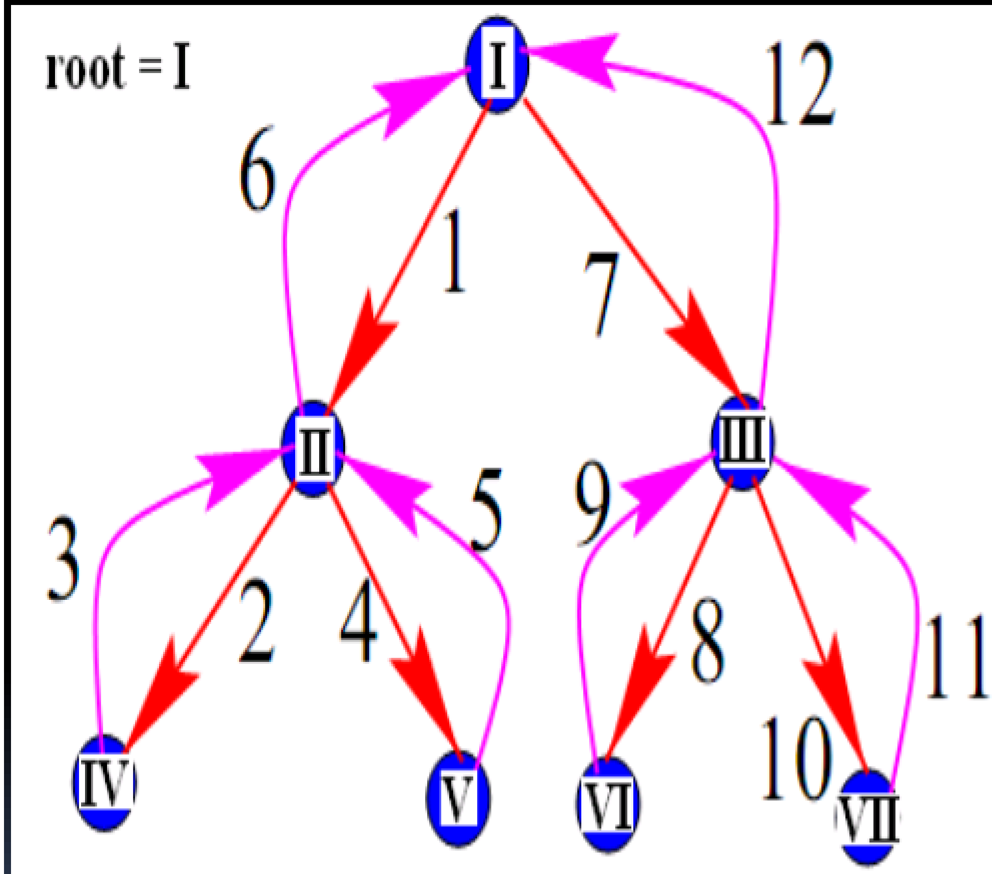
DUYỆT GIỮA IN-ORDER



DUYỆT SAU POST-ORDER



PHƯƠNG PHÁP TIẾP CẬN



Euler Tour		Tree	
Thứ tự	Cạnh	v	p(v)
1	<1,2>	1	1
2	<2,4>	2	1
3	<4,2>	3	1
4	<2,5>	4	2
5	<5,2>	5	2
6	<2,1>	6	3
7	<1,3>	7	3
8	<3,6>		
9	<6,3>		
10	<3,7>		
11	<7,3>		
12	<3,1>		

DUYỆT SAU POST-ORDER

- Thiết lập chu trình Euler trên cây T
- Với gốc r, xác định cây hướng gốc (với mọi v xác định p(v) là cha của v)
- Đặt trọng số cho các cạnh
  - w(<v,p(v)>) = 1 & w(<p(v),v>)=0.
- Với mỗi cặp <u,v>, xác định suffix_sum cho cạnh <u,v>. Ta gọi là S(<u,v>)
- Vị trí duyệt v là: |V| - S(<v,p(v)>)
- Cuối cùng là duyệt đỉnh gốc r

DUYỆT SAU POST-ORDER

Euler Tour				Tree	
Thứ tự	Cạnh	Giá trị	Suffix_sum	v	p(v)
1	<1,2>	0	6	1	1
2	<2,4>	0	6	2	1
3	<4,2>	1	6	3	1
4	<2,5>	0	5	4	2
5	<5,2>	1	5	5	2
6	<2,1>	1	4	6	3
7	<1,3>	0	3	7	3
8	<3,6>	0	3		
9	<6,3>	1	3		
10	<3,7>	0	2		
11	<7,3>	1	2		
12	<3,1>	1	1		

DUYỆT SAU POST-ORDER

- Xác định vị trí các đỉnh như sau:
  - S(<2,1>) = 4 -> Position (2) = 7 – 4 = 3
  - S(<3,1>) = 1 -> Position (3) = 7 – 1 = 6
  - S(<4,2>) = 6 -> Position (4) = 7 – 6 = 1
  - S(<5,2>) = 5 -> Position (5) = 7 – 5 = 2
  - S(<6,3>) = 3 -> Position (6) = 7 – 3 = 4
  - S(<7,3>) = 2 -> Position (7) = 7 – 2 = 5
- Thứ tự duyệt là :
  - [4->5->2->6->7->3->1]

DUYỆT TRƯỚC PRE-ORDER

- Thiết lập chu trình Euler trên cây T
- Với gốc r, xác định cây hướng gốc (với mọi v xác định p(v) là cha của v)
- Đặt trọng số cho các cạnh
  - w(<v,p(v)>) = 0 & w(<p(v),v>) = 1
- Với mỗi cặp <u,v> xác định suffix_sum cho cạnh <u,v>. Ta gọi là S(<u,v>)
- Vị trí duyệt v là |V| - S(<p(v),v>)
- Gốc duyệt đầu tiên

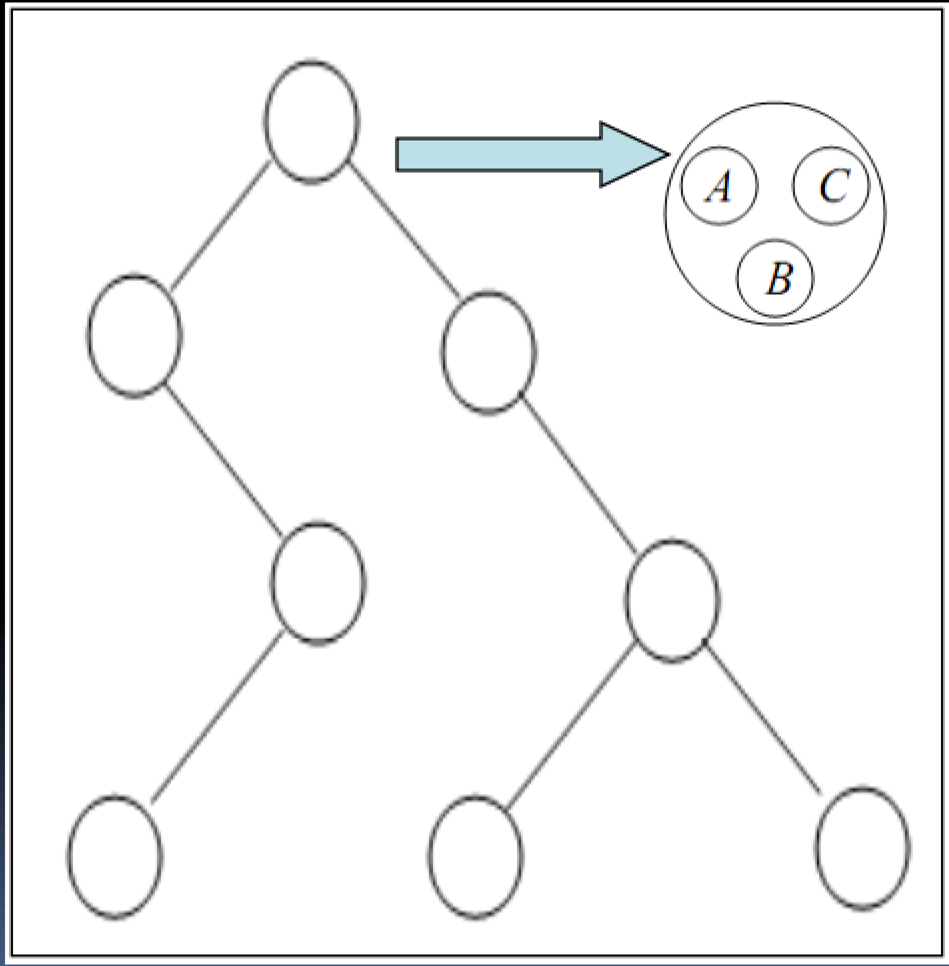
DUYỆT TRƯỚC PRE-ORDER

Euler Tour				Tree	
Thứ tự	Cạnh	Giá trị	Suffix_sum	v	p(v)
1	<1,2>	1	6	1	1
2	<2,4>	1	5	2	1
3	<4,2>	0	4	3	1
4	<2,5>	1	4	4	2
5	<5,2>	0	3	5	2
6	<2,1>	0	3	6	3
7	<1,3>	1	3	7	3
8	<3,6>	1	2		
9	<6,3>	0	1		
10	<3,7>	1	1		
11	<7,3>	0	0		
12	<3,1>	0	0		

DUYỆT TRƯỚC PRE-ORDER

- Xác định vị trí các đỉnh như sau:
  - S(<1,2>) = 6 -> Position (2) = 7 – 6 = 1
  - S(<1,3>) = 3 -> Position (3) = 7 – 3 = 4
  - S(<2,4>) = 5 -> Position (4) = 7 – 5 = 2
  - S(<2,5>) = 4 -> Position (5) = 7 – 4 = 3
  - S(<3,6>) = 2 -> Position (6) = 7 – 2 = 5
  - S(<3,7>) = 1 -> Position (7) = 7 – 1 = 6
- Thứ tự duyệt là:
  - [1 -> 2 -> 4 -> 5 -> 3 -> 6 -> 7]

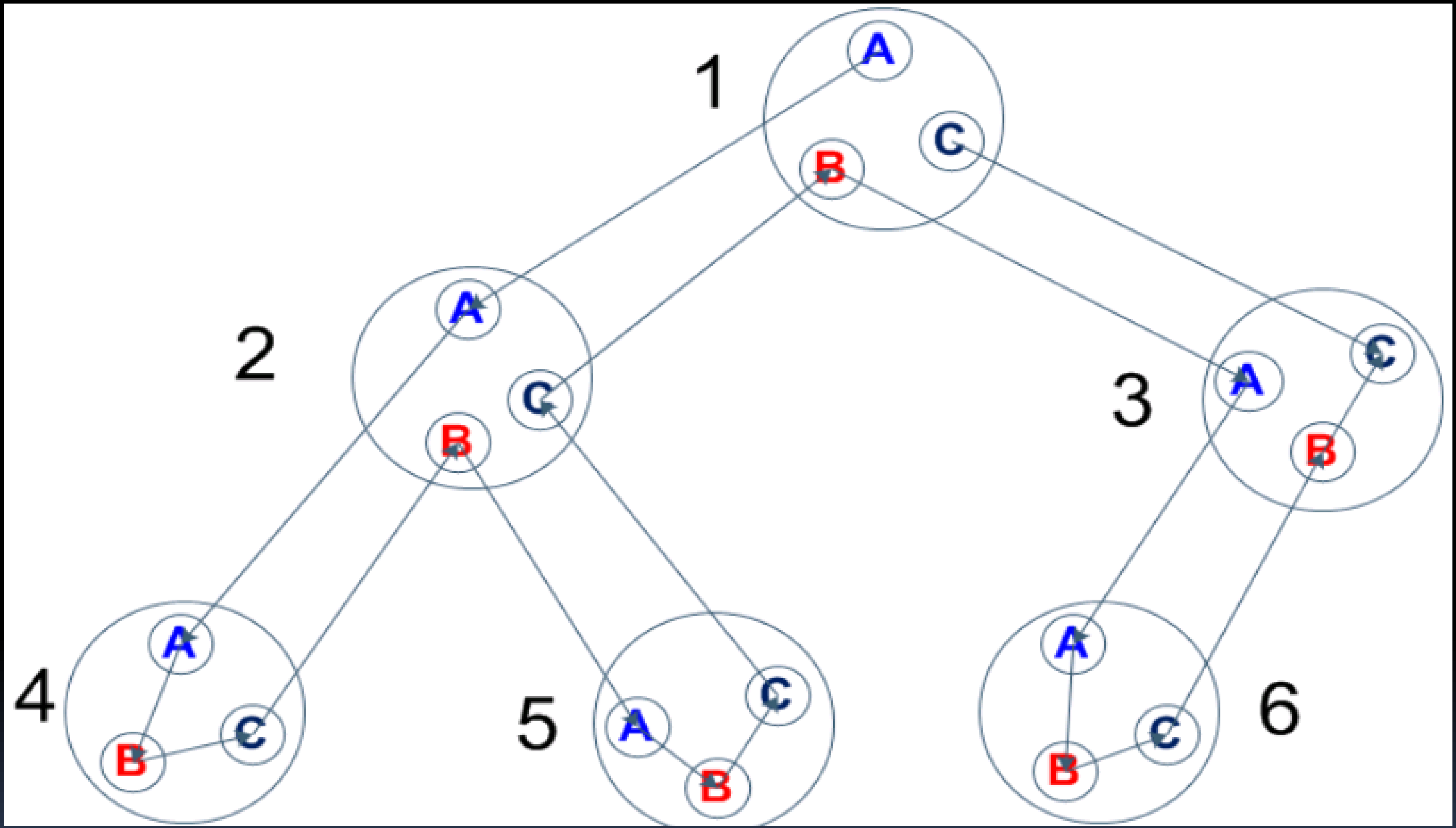
VÍ DỤ MINH HỌA



CÁCH TIẾP CẬN KHÁC

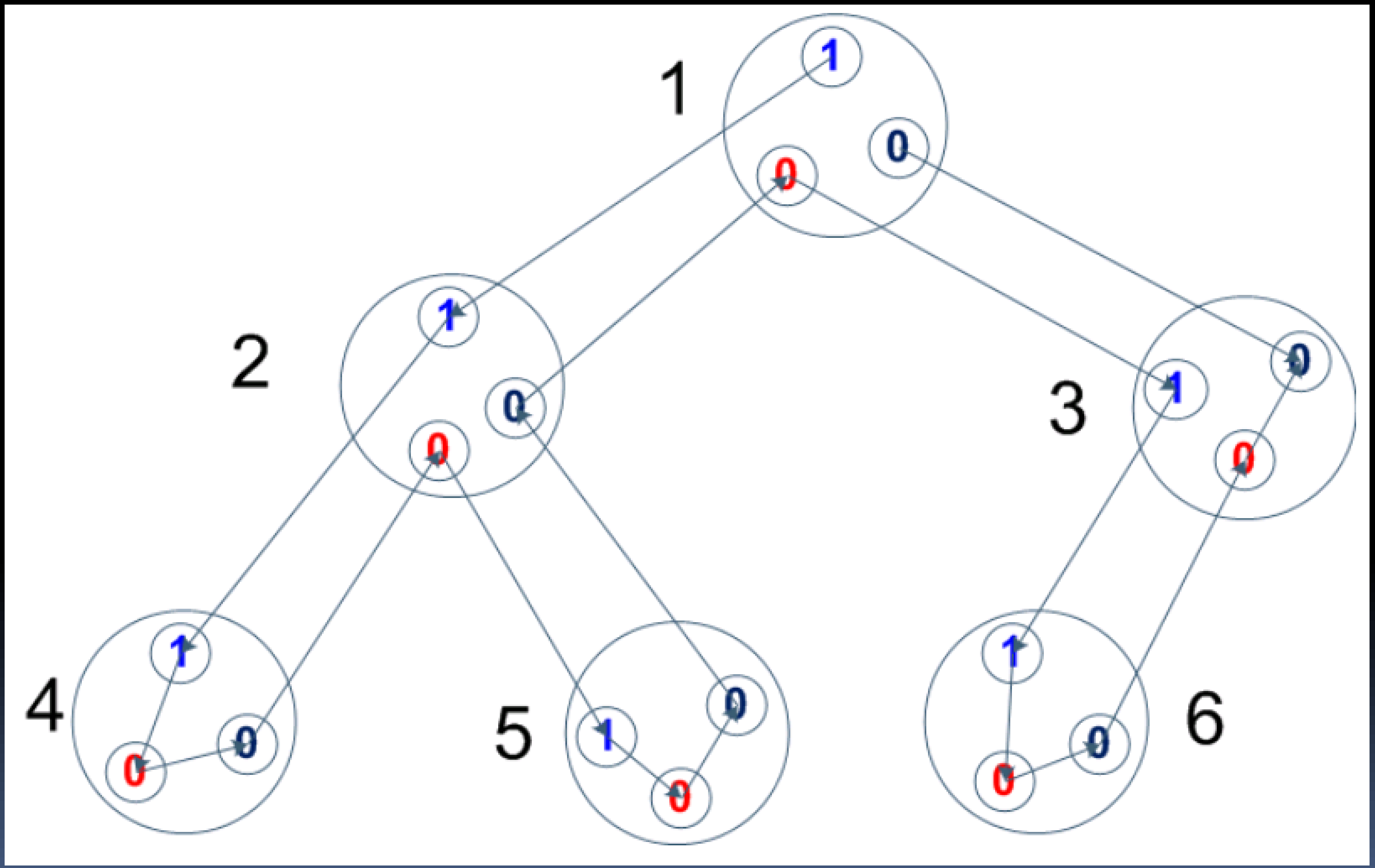
- Đối với cây nhị phân:
  - Mỗi nút v được coi là 3 nút con: v[a], v[b], v[c]
  - Quy tắc của nút [a]:
    - Nếu v có con trái là u thì: v[a] -> u[a]
    - Nếu v không có con trái thì: v[a] -> v[b]
  - Quy tắc của nút [b]:
    - Nếu v có con phải là u thì: v[b] -> u[a]
    - Nếu v không có con phải thì: v[b] -> v[c]
  - Quy tắc của nút [c]:
    - Nếu v là con trái của u thì: v[c] -> u[b]
    - Nếu v là con phải của u thì: v[c] -> u[c]
    - Nếu v là nút gốc thì: v[c] -> NULL

CHU TRÌNH EULER VÀ LINKED LIST

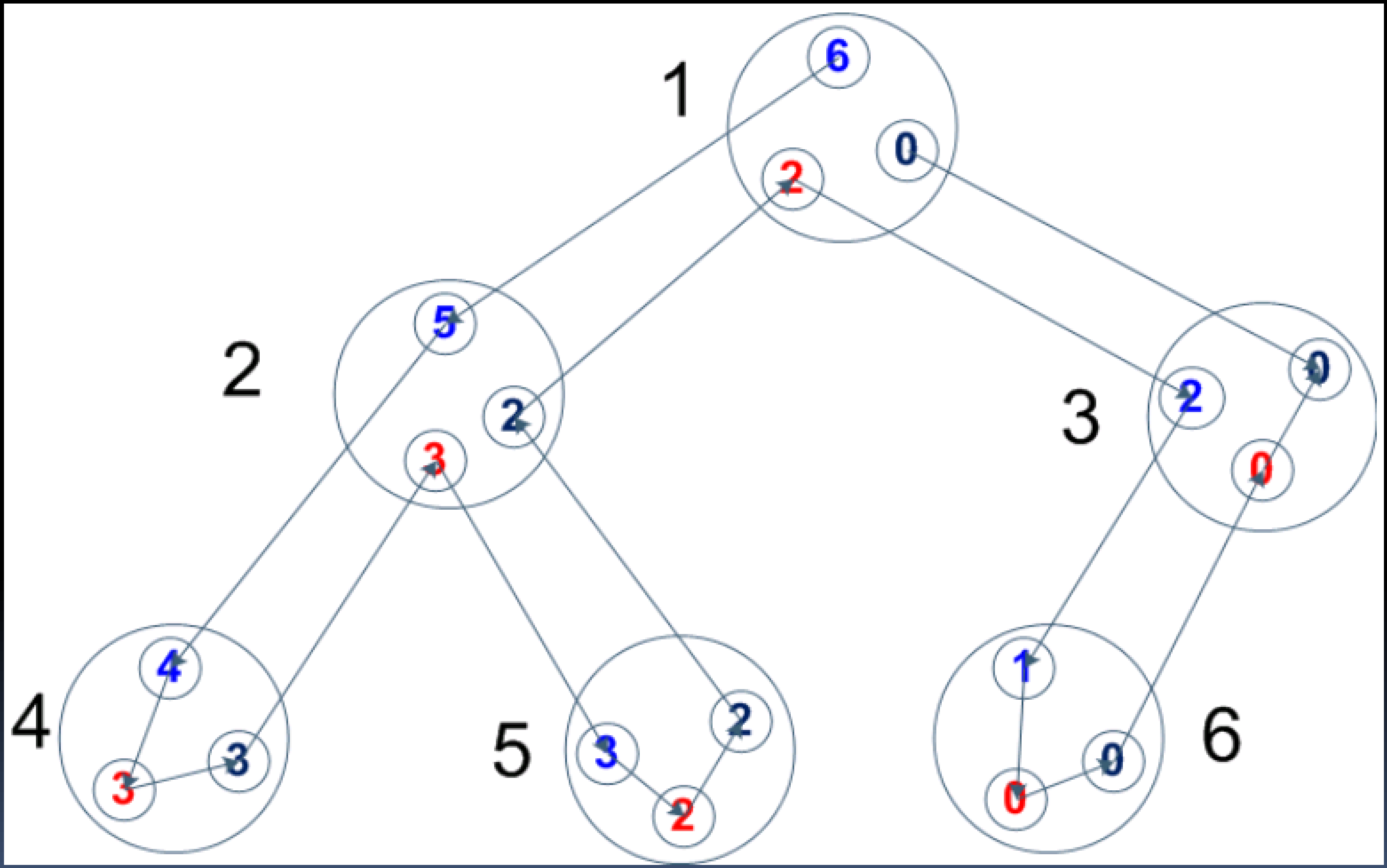


1[A] → 2[A] → 4[A] → 4[B] → 4[C] → 2[B] → 5[A] → 5[B] →5[C] → 2[C] → 1[B] → 3[A] → 6[A] → 6[B] → 6[C] → 3[B] → 3[C] → 1[C] → (NULL)

PREORDER: A=1, B=0, C=0



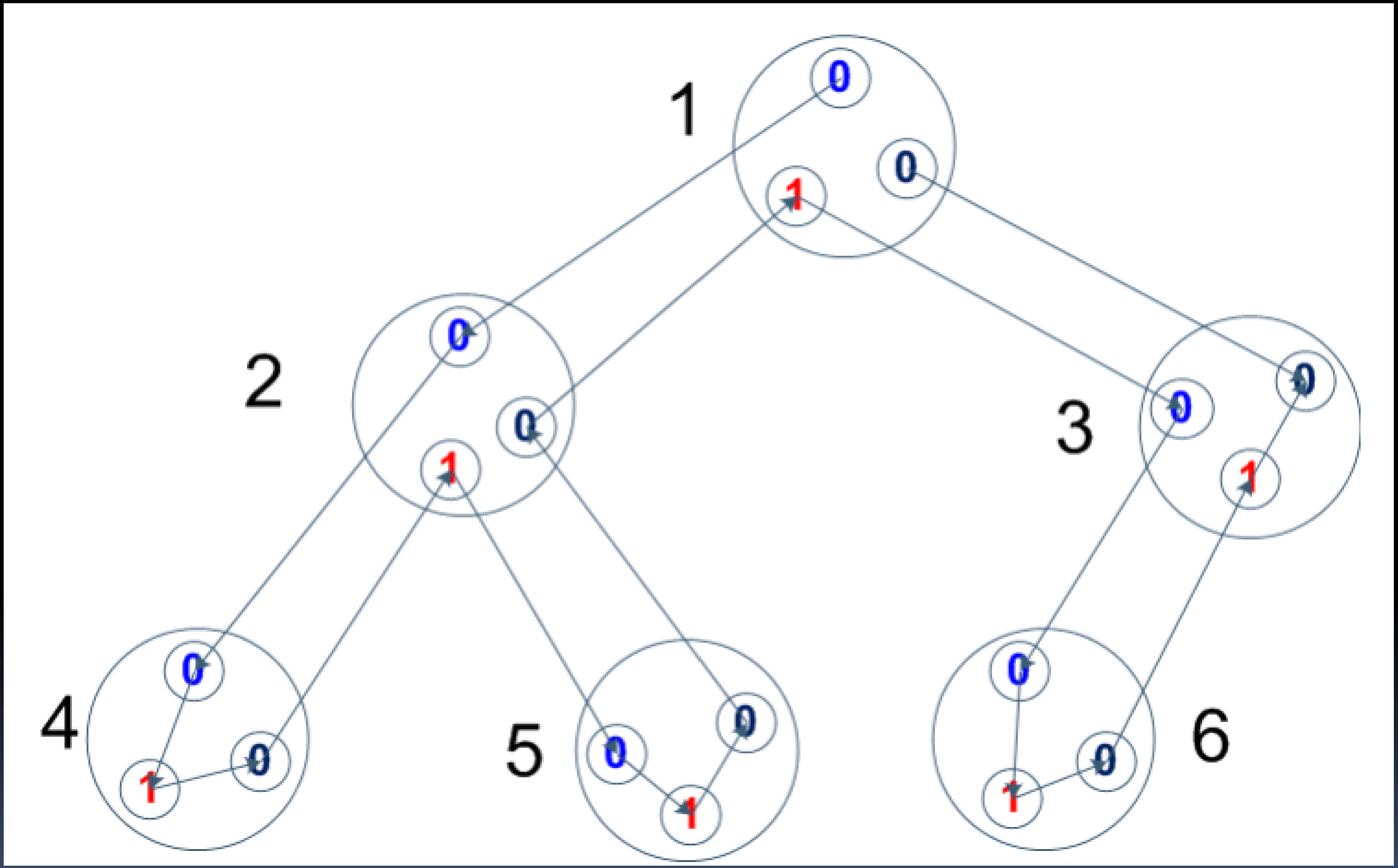
TÍNH SUFFIX SUM TRÊN DANH SÁCH



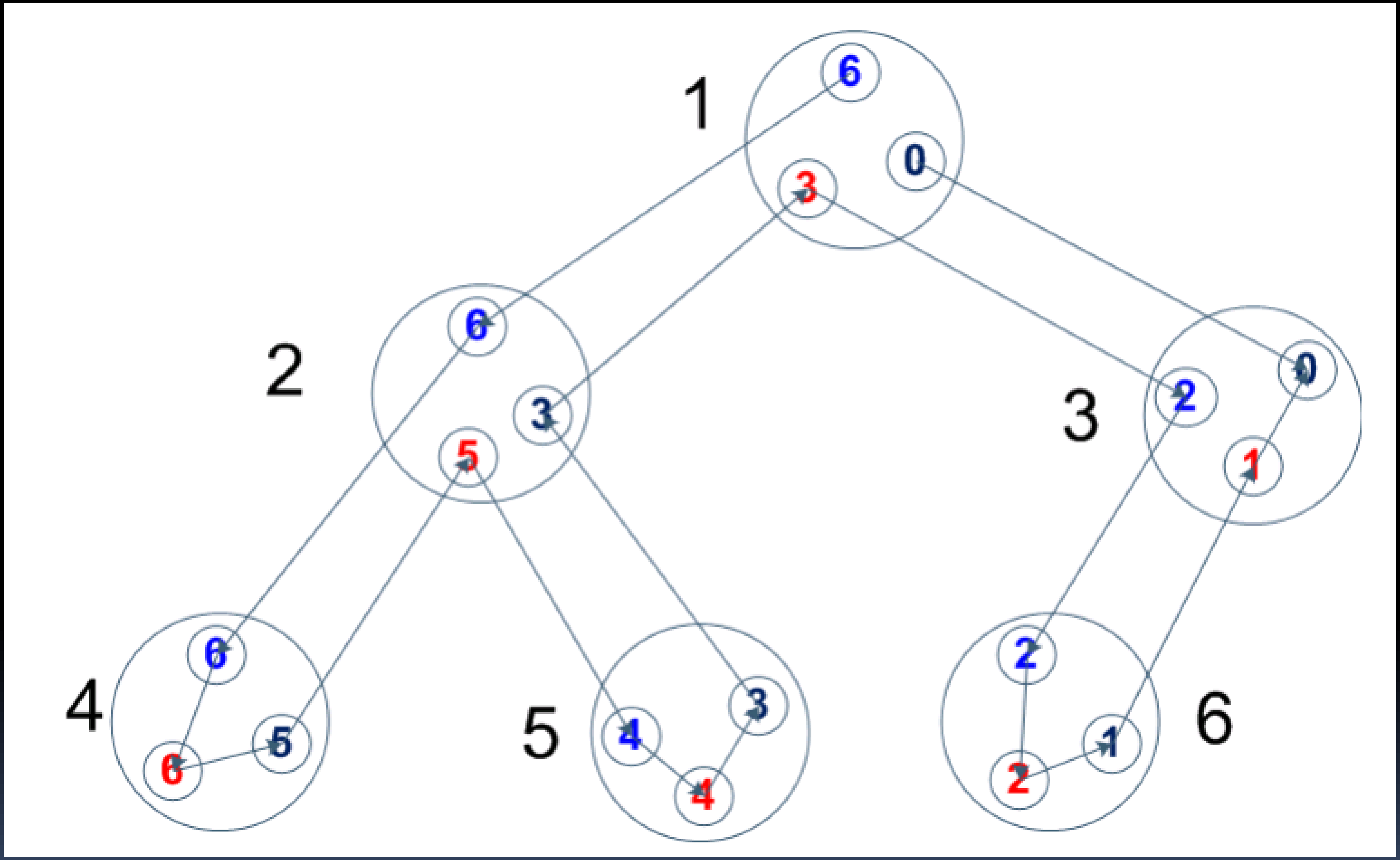
THỨ TỰ DUYỆT PREORDER

- Thứ tự duyệt đỉnh v: |V| - v[A] + 1
  - 1[A] = 6 -> Position (1) = 6 – 6 + 1 = 1
  - 2[A] = 5 -> Position (2) = 6 – 5 + 1 = 2
  - 3[A] = 2 -> Position (3) = 6 – 2 + 1 = 5
  - 4[A] = 4 -> Position (4) = 6 – 4 + 1 = 3
  - 5[A] = 3 -> Position (5) = 6 – 3 + 1 = 4
  - 6[A] = 1 -> Position (6) = 6 – 1 + 1 = 6
- Thứ tự duyệt là [1 -> 2 -> 4 -> 5 -> 3 -> 6]

INORDER: A= 0, B=1, C=0



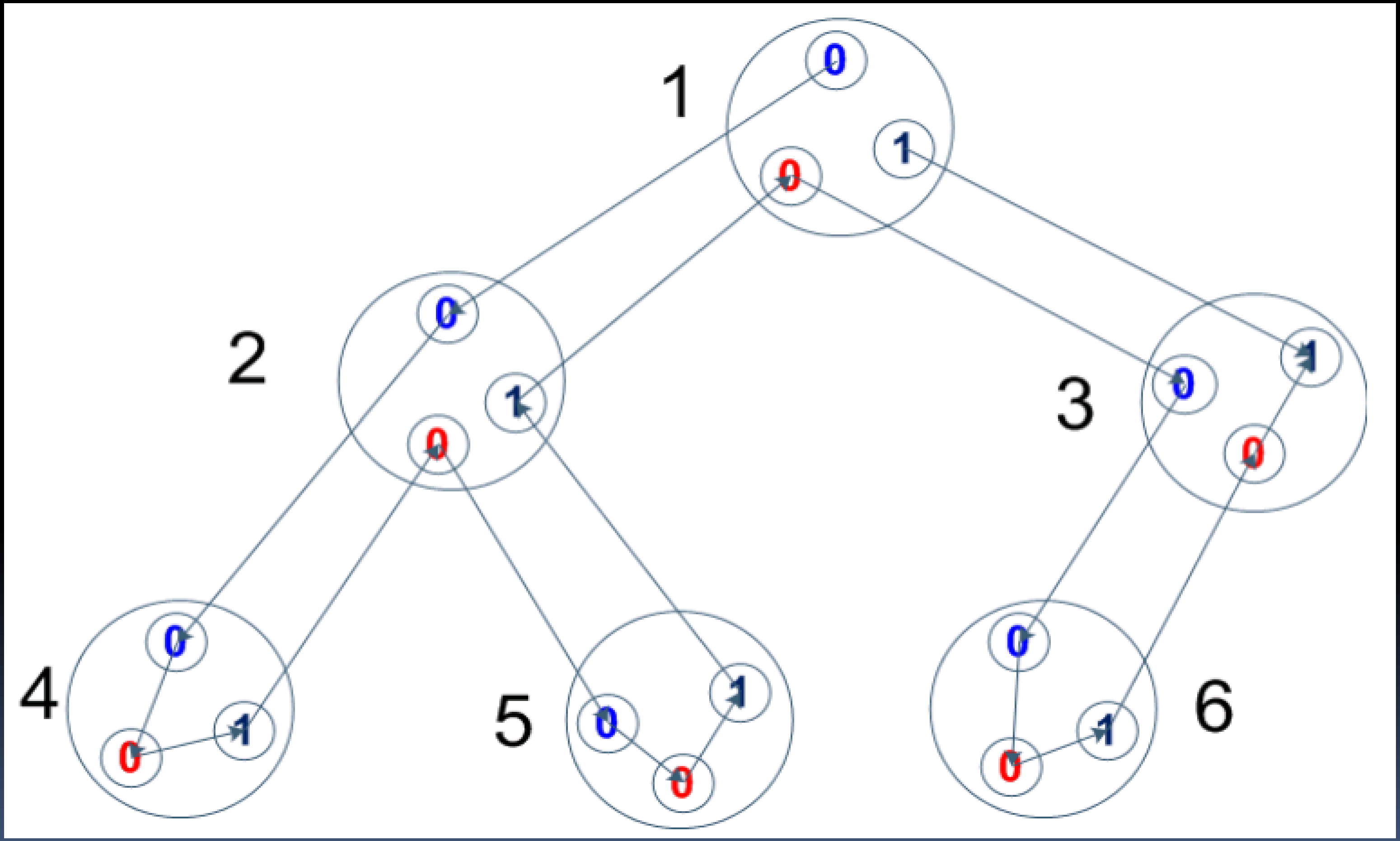
TÍNH SUFFIX SUM TRÊN DANH SÁCH



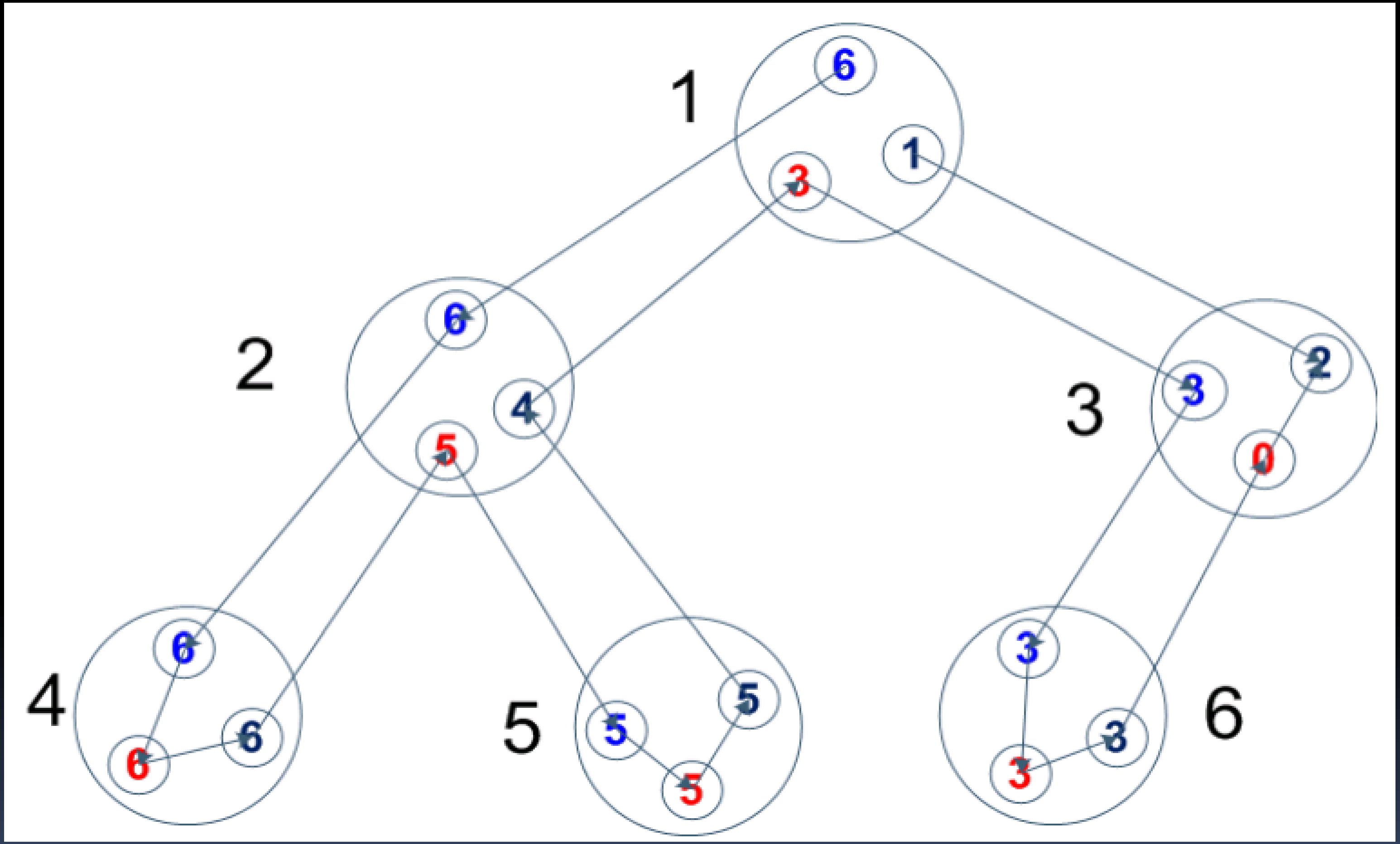
THỨ TỰ DUYỆT INORDER

- Thứ tự duyệt đỉnh v: |V| - v[B] + 1
  - 1[B] = 3 -> Position (1) = 6 – 3 + 1 = 4
  - 2[B] = 5 -> Position (2) = 6 – 5 + 1 = 2
  - 3[B] = 1 -> Position (3) = 6 – 1 + 1 = 6
  - 4[B] = 6 -> Position (4) = 6 – 6 + 1 = 1
  - 5[B] = 4 -> Position (5) = 6 – 4 + 1 = 3
  - 6[B] = 2 -> Position (6) = 6 – 2 + 1 = 5
- Thứ tự duyệt là [4 -> 2 -> 5 -> 1 -> 6 -> 3]

POSTORDER: A= 0, B=0, C=1



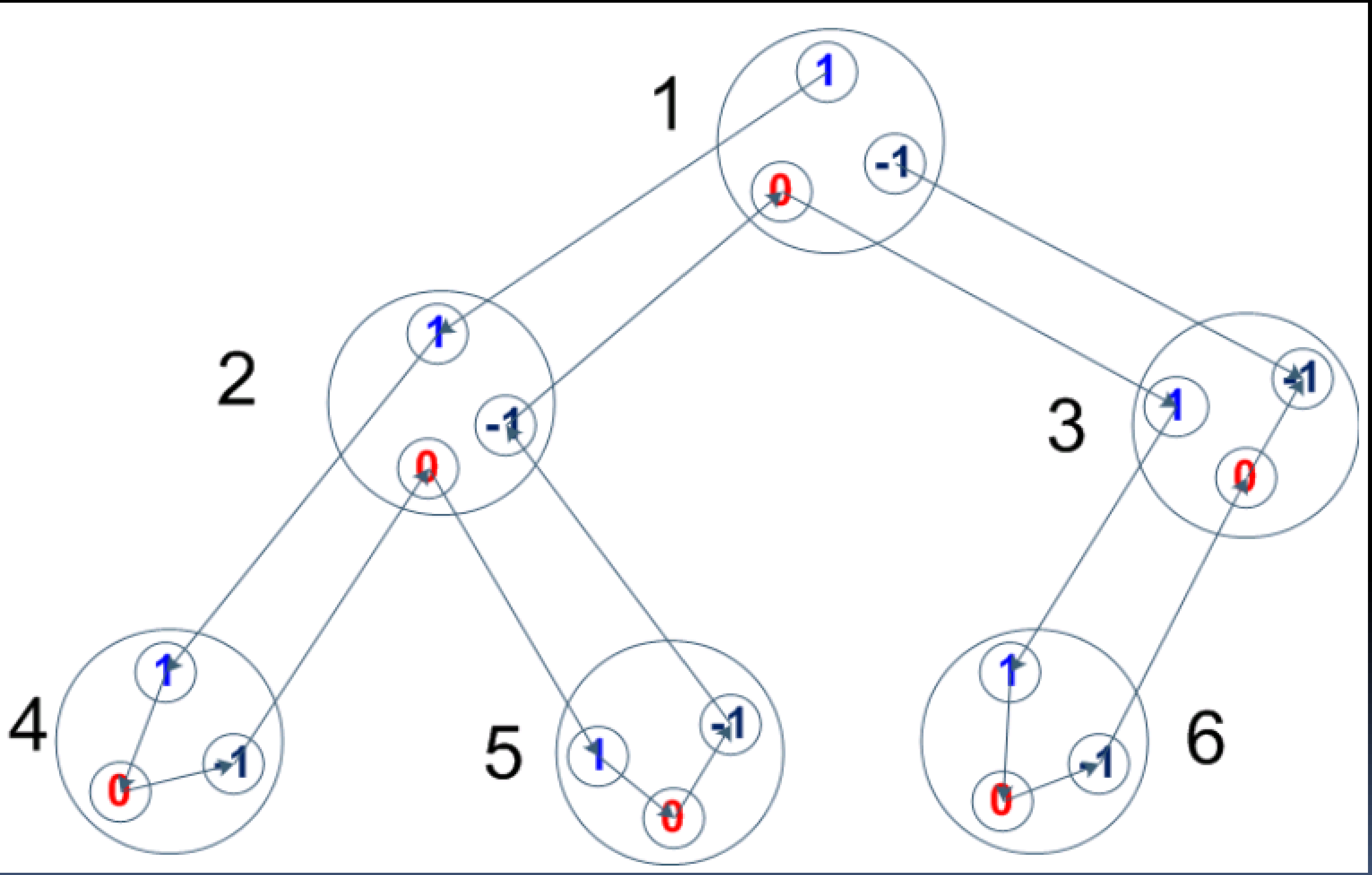
TÍNH SUFFIX SUM TRÊN DANH SÁCH



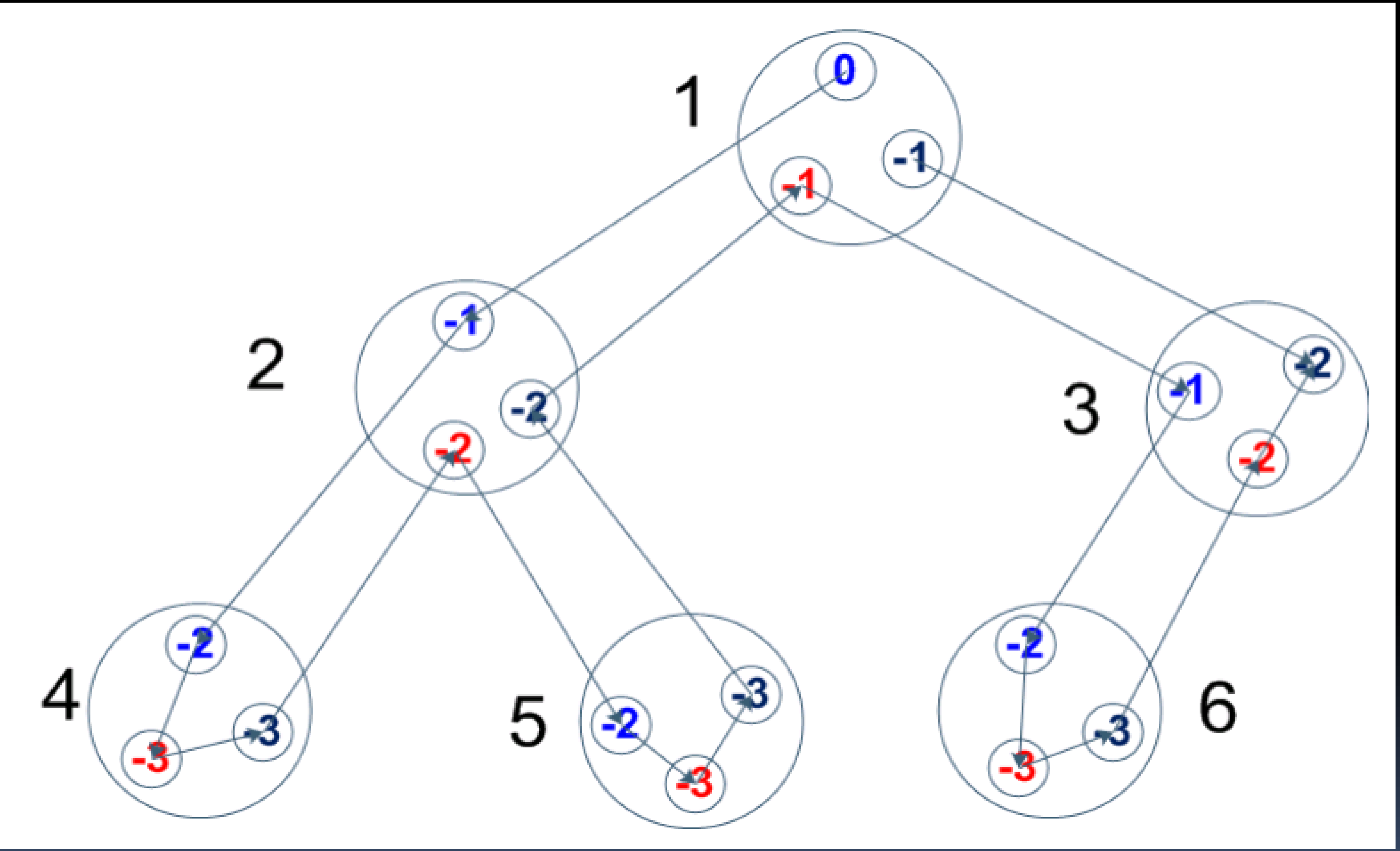
THỨ TỰ DUYỆT POSTORDER

- Thứ tự duyệt đỉnh v: |V| - v[C] + 1
  - 1[C] = 1 -> Position (1) = 6 – 1 + 1 = 6
  - 2[C] = 4 -> Position (2) = 6 – 4 + 1 = 3
  - 3[C] = 2 -> Position (3) = 6 – 2 + 1 = 5
  - 4[C] = 6 -> Position (4) = 6 – 6 + 1 = 1
  - 5[C] = 5 -> Position (5) = 6 – 5 + 1 = 2
  - 6[C] = 3 -> Position (6) = 6 – 3 + 1 = 4
- Thứ tự duyệt là [4 -> 5 -> 2 -> 6 -> 3 -> 1]

DEPTH(V): A=1, B=0, C=-1



TÍNH SUFFIX SUM TRÊN DANH SÁCH

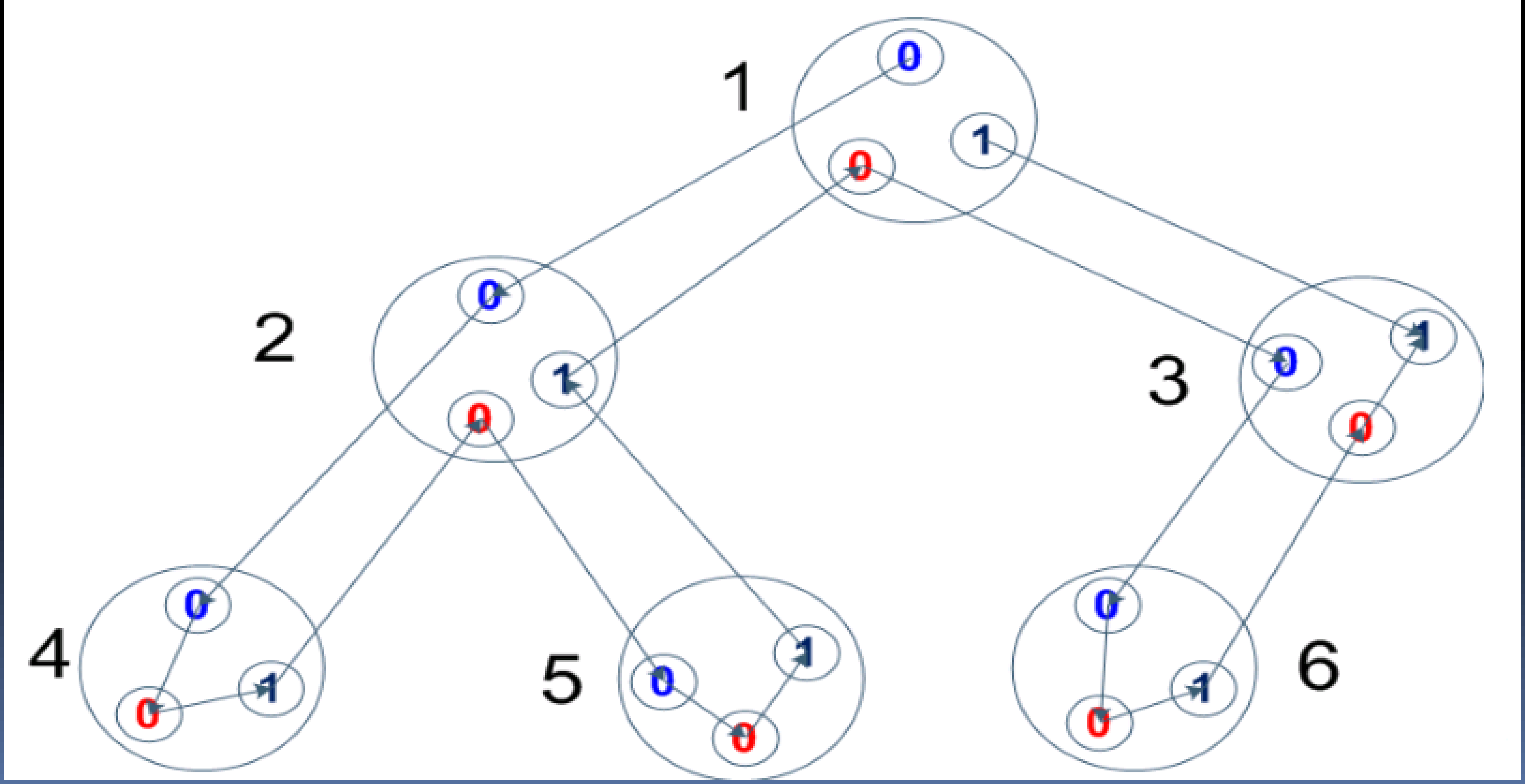


XÁC ĐỊNH ĐỘ SÂU CÁC NÚT

- Độ cao của đỉnh v: abs(v[A])
  - 1[A] = 0 -> Position (1) = 6 – 1 + 1 = 0
  - 2[A] = -1 -> Position (2) = 6 – 4 + 1 = 1
  - 3[A] = -1 -> Position (3) = 6 – 2 + 1 = 1
  - 4[A] = -2 -> Position (4) = 6 – 6 + 1 = 2
  - 5[A] = -2 -> Position (5) = 6 – 5 + 1 = 2
  - 6[A] = -2 -> Position (6) = 6 – 3 + 1 = 2
- Độ cao: Height(v) = H – Depth(v). Trong đó H = max {Depth(v)}

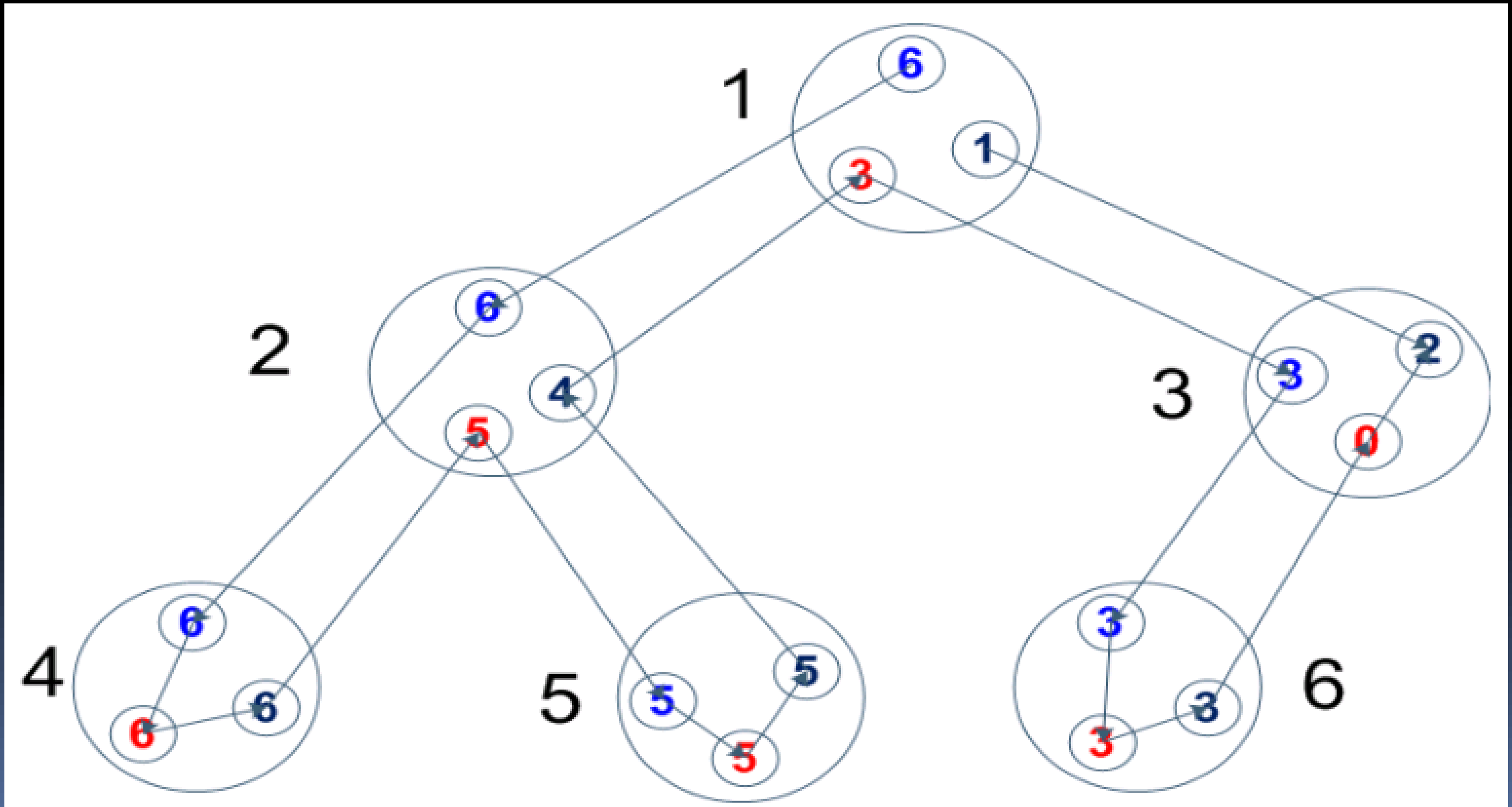
XÁC ĐỊNH KÍCH THƯỚC CÂY CÓ GỐC V

- Với mọi v, xác định số nút trong cây con có v là gốc. Đặt A = 0, B = 0, C = 1



XÁC ĐỊNH KÍCH THƯỚC CÂY CÓ GỐC V

- Tính Suffix Sum trên danh sách dựa theo chu trình Euler



XÁC ĐỊNH KÍCH THƯỚC CÂY CÓ GỐC V

- Size (v) = v[A] – v[C] + 1
  - Size (1) = 6 – 1 + 1 = 6
  - Size (2) = 6 – 4 + 1 = 3
  - Size (3) = 3 – 2 + 1 = 2
  - Size (4) = 6 – 6 + 1 = 1
  - Size (5) = 5 – 5 + 1 = 1
  - Size (6) = 3 – 3 + 1 = 1

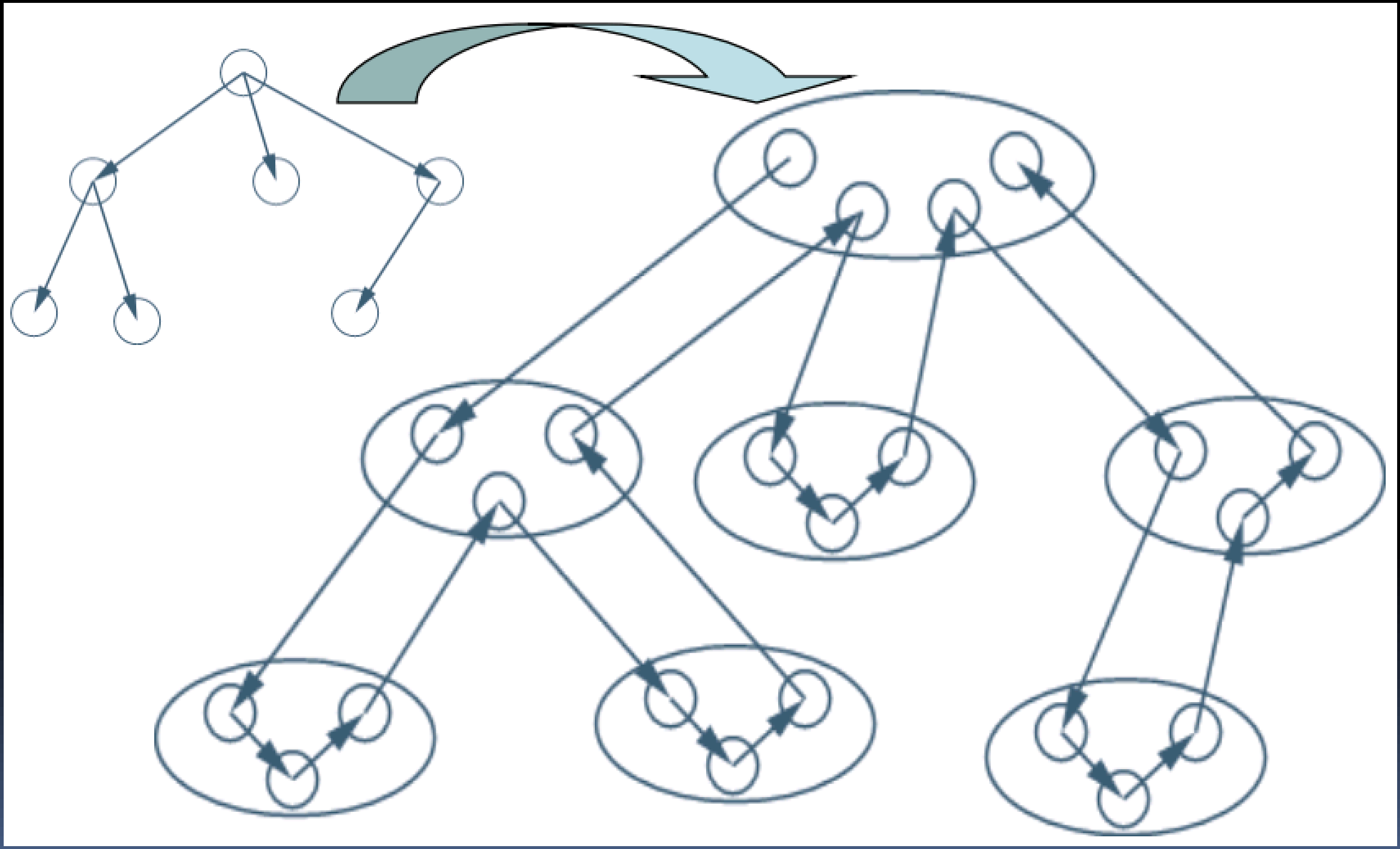
CHU TRÌNH EULER CHO CÂY TỔNG QUÁT

- Xét 1 nút v bất kỳ. Giả sử $\{v_1, v_2, \ldots v_m\}$ là các con của v từ trái qua phải.
- Mô phỏng v bởi m+1 nút con:
  - v[A]: Điểm vào của v trong chu trình Euler
  - v[C]: Điểm ra của v trong chu trình Euler
  - $v[B_k]$: nối với các nút con của v_{k+1}. (k=1..m-1)
- Nếu v là nút lá hoặc chỉ có 1 con thì vẫn biểu diễn v bởi v[A], v[B], v[C]

QUY TẮC NỐI ĐỈNH

- Quy tắc đối với A:
  - Nếu v có con trái ngoài cùng là v_1 thì v[A] nối với v_1[A]
  - Nếu v không có con thì v[A] nối với v[B]
- Quy tắc đối với B:
  - Nếu v là nút thì v[B] nối với v[C]
  - Nếu v có các nút con {v_1, v_2,…, v_m} thì v[B_k] nối với v_{k+1}[A]. Với k = 1.. m-1
- Quy tắc đối với C:
  - Nếu v là con ngoài cùng bên phải của u thì v[C] nối với u[C]
  - Nếu v là con thứ k của u thì v[C] nối với u[B_k]
  - Nếu v là gốc thì v[C] nối với NULL

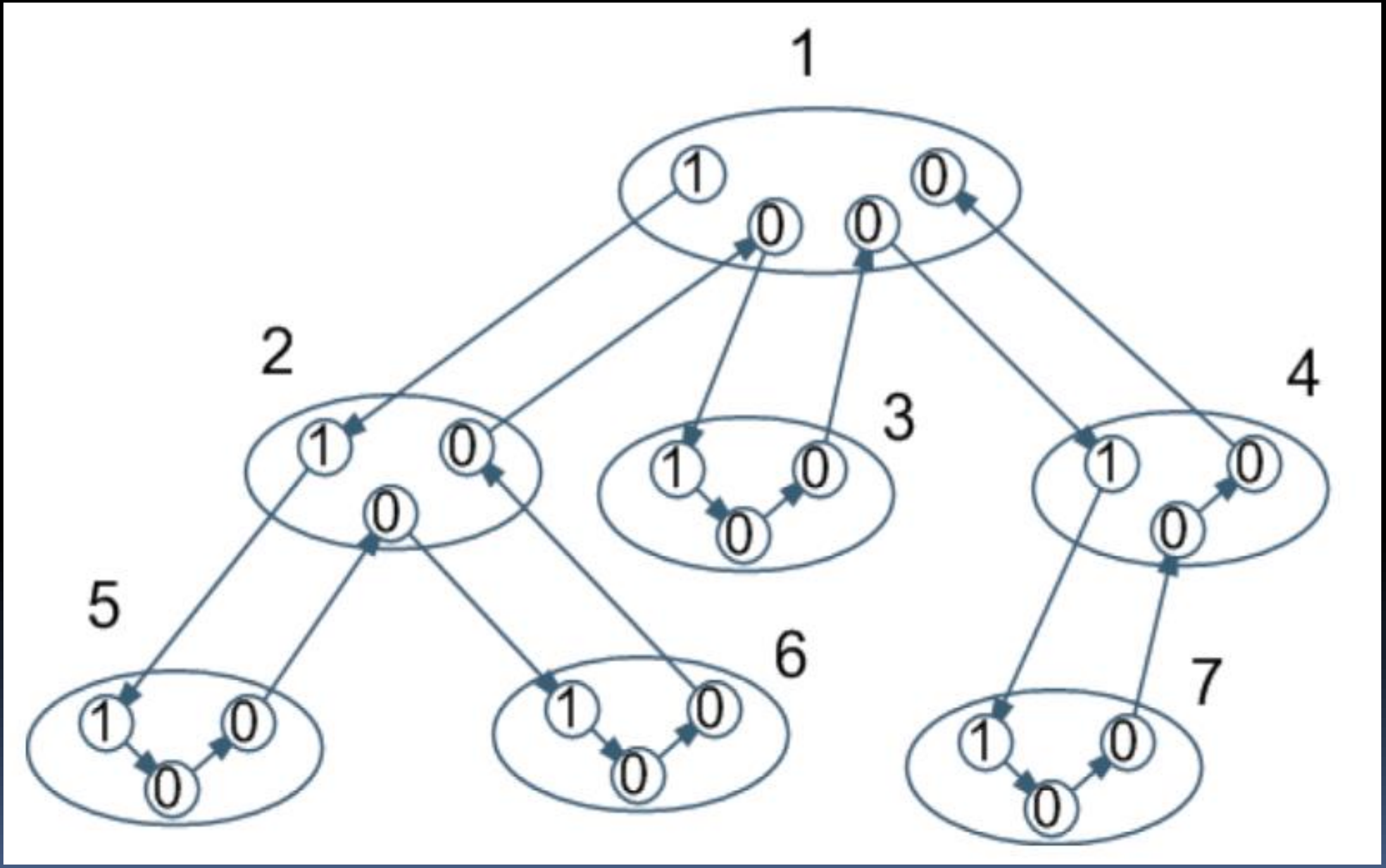
VÍ DỤ MINH HỌA



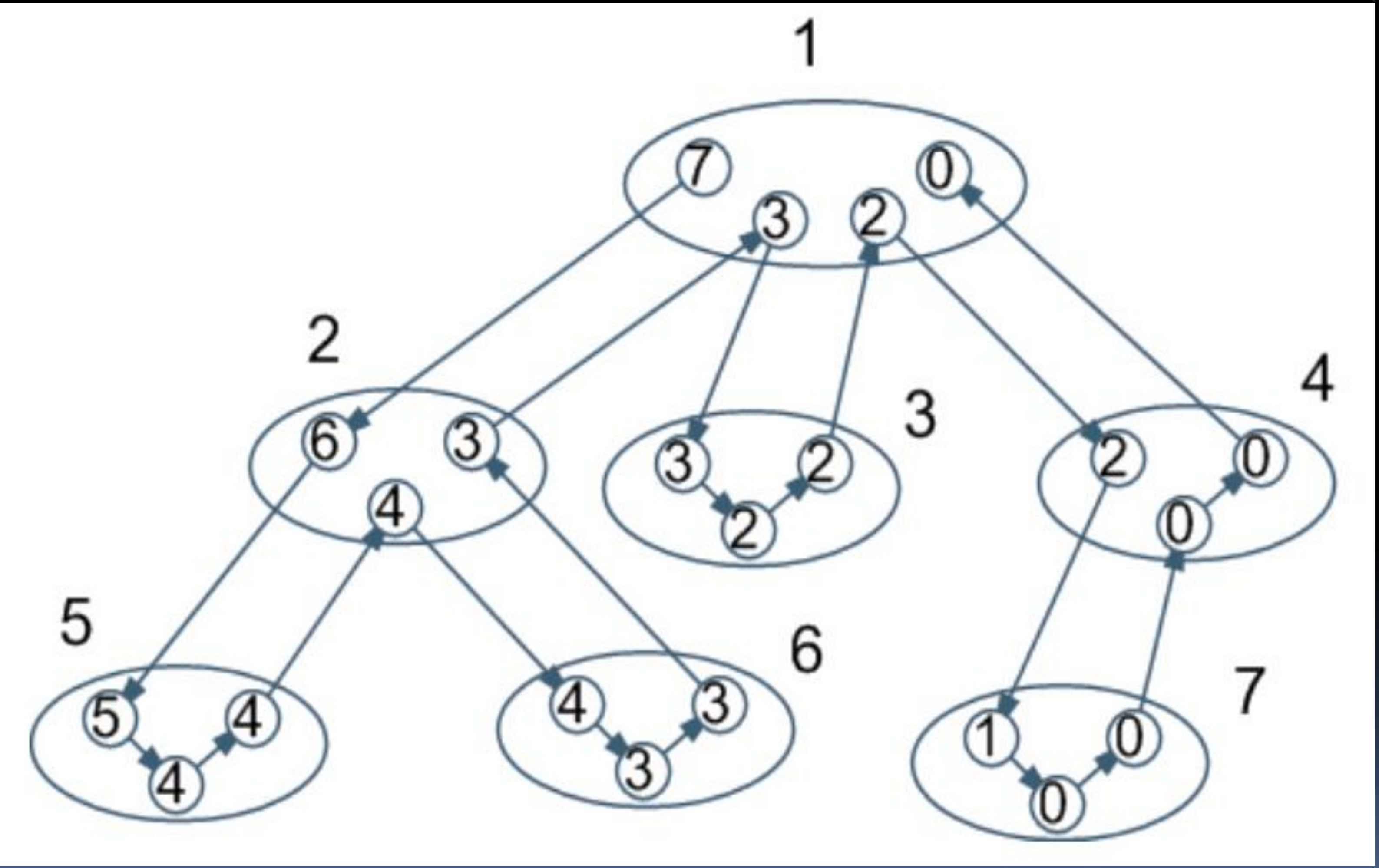
CÁC BÀI TOÁN VỚI CÂY TỔNG QUÁT

- Bài toán duyệt:
  - Không có khái niệm InOrder
  - PreOrder và PostOrder duyệt giữa trên đỉnh đầu vào A và đỉnh ra C. Các giá trị $B_k = 0$
    - PreOrder: A = 1; C = 0
    - PostOrder: A = 0; C = 1
  - Bài toán về độ sâu, độ cao: A = 1; C = -1.
  - Bài toán kích thước cây con; A = 0; C = 1

PREORDER: A = 1, B_k = 0, C = 0



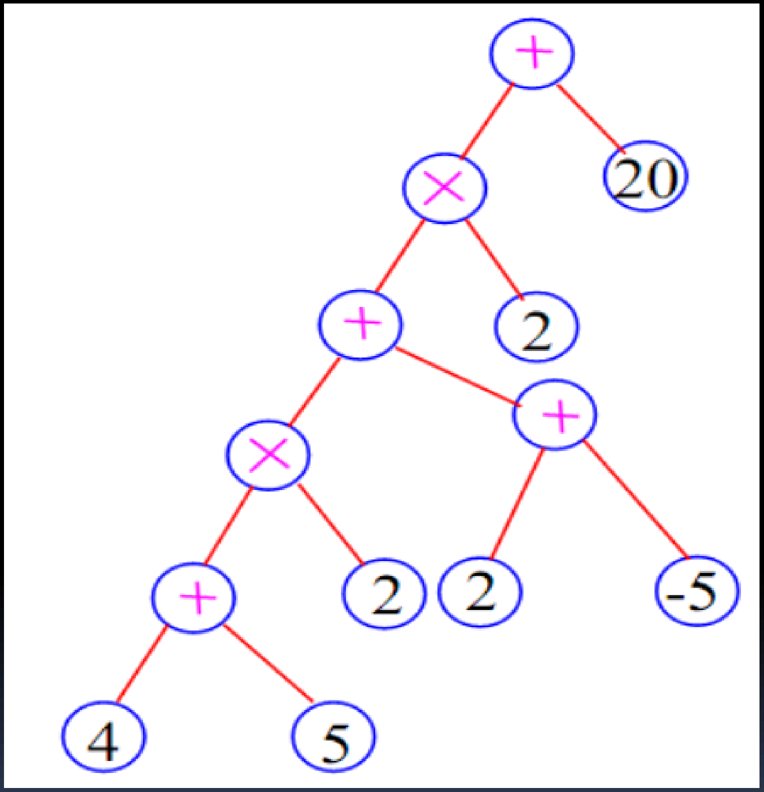
PREORDER: A = 1, B_k = 0, C = 0



3.3. TREE CONTRACTION

PHÁT BIỂU BÀI TOÁN

- Cho một cây nhị phân. Hãy thực hiện việc rút gọn cây lại thành 1 cây bao gồm 1 gốc và 2 nút con.

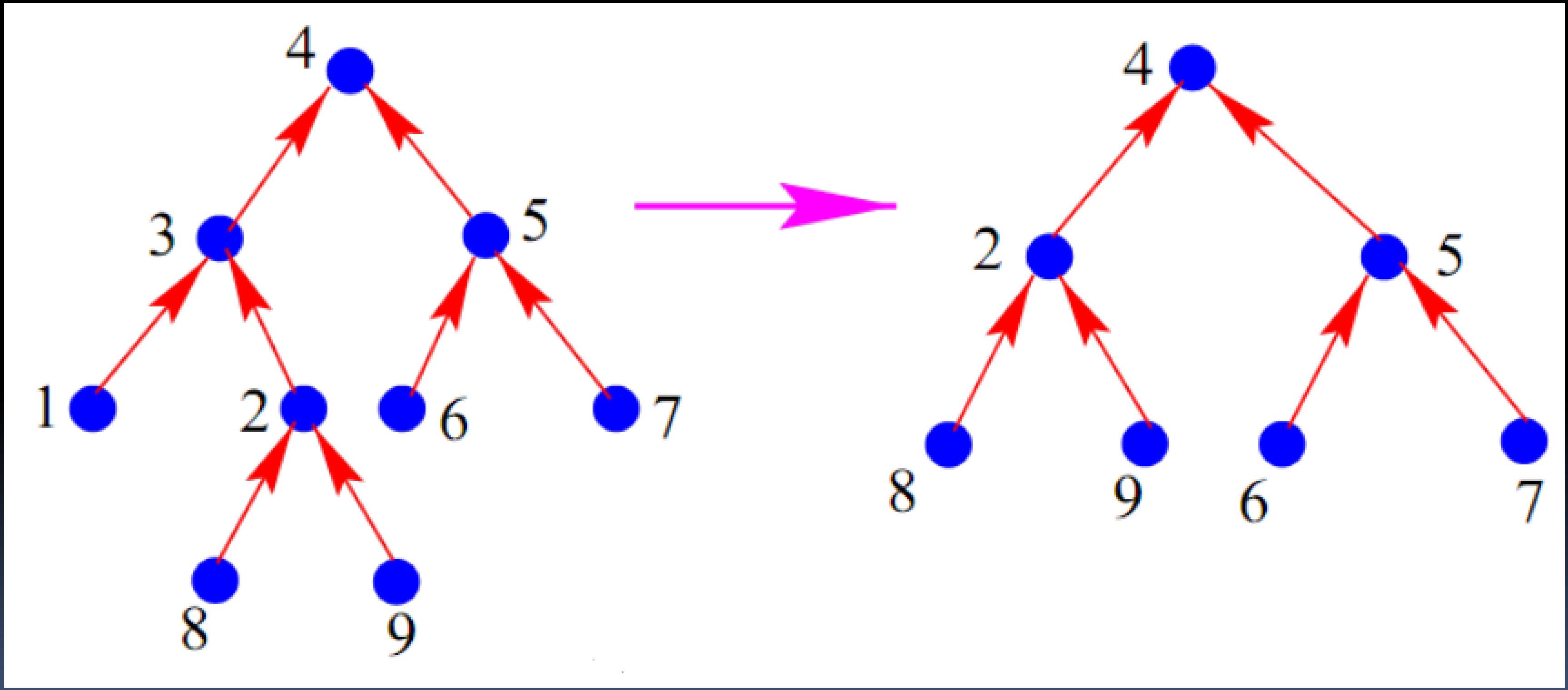


PHƯƠNG PHÁP TIẾP CẬN

- Cây T = (V,E) là 1 cây nhị phân gốc r:
  - p(v) là nút cha của v trên cây T
  - sib(v) là nút anh em của v: là con của cùng 1 nút cha.
- Thao tác RAKE cho nút lá v: p(v) <> r
  - Xóa các nút v, p(v) trên cây T.
  - Nối sib(v) với p(p(v)) trên cây T

PHƯƠNG PHÁP TIẾP CẬN

- Toán tử RAKE – rút gọn các nút lá:



PHƯƠNG PHÁP TIẾP CẬN

- Vấn đề nảy sinh
  - Không thể thực hiện toán tử RAKE với nút lá nối liền với gốc
  - Chỉ thực hiện song song trên các lá mà cha của chúng không kề nhau
  - Ví dụ: nút lá 1,8,9 không thể cùng thực hiện

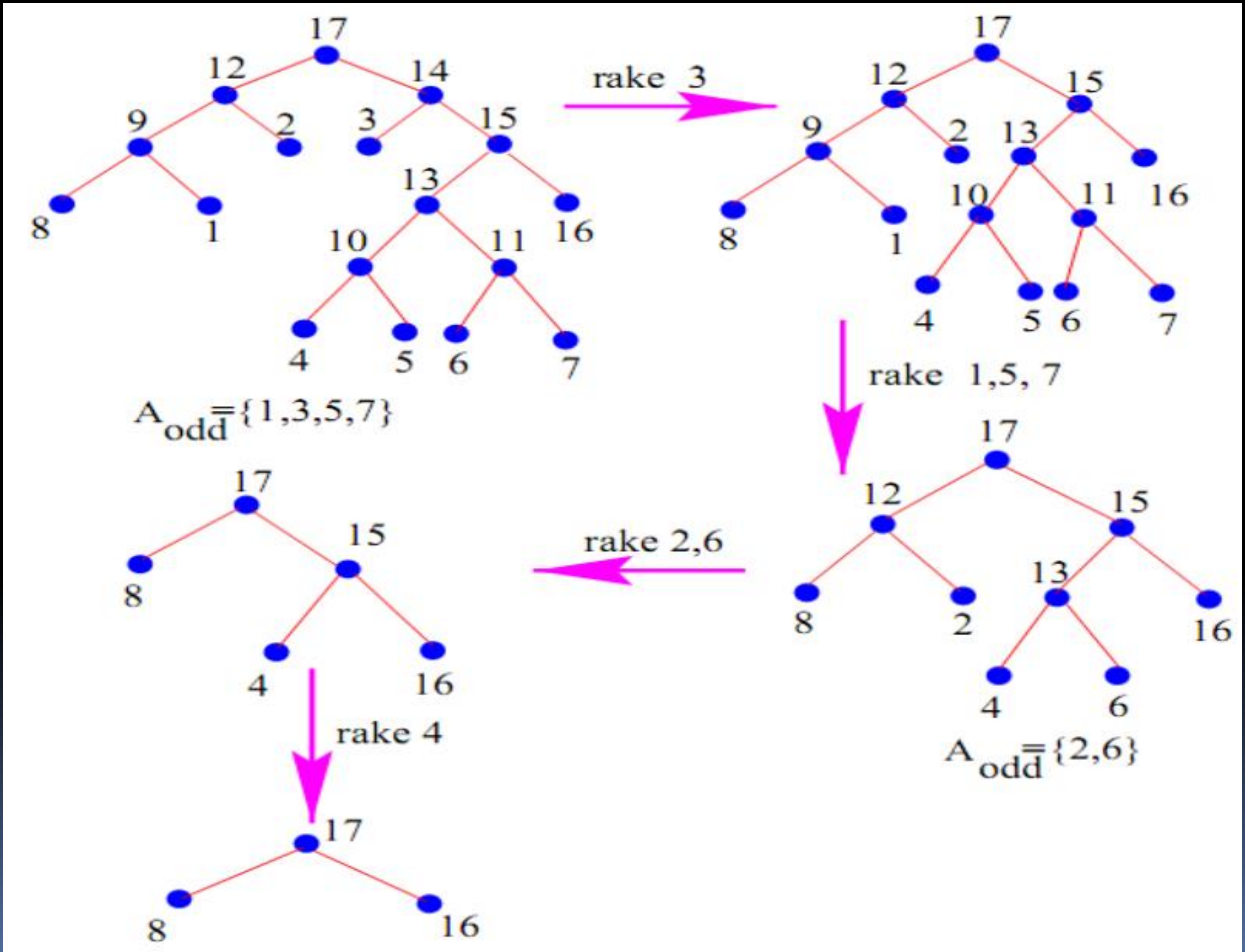
PHƯƠNG PHÁP TIẾP CẬN

- Cách giải quyết:
  - Mỗi nút cha phải lưu trữ thông tin về con trái và con phải của mình
  - Đánh dấu các nút là theo thứ tự từ 1,2,…
  - Xét các nút có chỉ số lẻ:
    - Các nút là con trái thì cha của chúng sẽ không kề nhau. Ta gọi là nhóm 1
    - Tương tự với các nút con phải lẻ. Ta gọi là nhóm 2
  - Thực hiện song song trên từng nhóm lần lượt sẽ đảm bảo không bị vi phạm điều kiện của RAKE

CÁC BƯỚC THUẬT TOÁN

- B1: Đánh dấu các nút lá theo thứ tự từ 1..n để lưu vào mảng Z, (ngoại trừ 2 nút lá nằm bên trái, phải ngoài cùng)
- Lặp:
  - B2. Thực hiện RAKE với các nút Z[k] với k lẻ và là con bên trái
  - B3. Thực hiện RAKE với các nút Z[k] với k lẻ còn lại
  - B4. Gán Z = tập các Z[k] với k chẵn
- Cho đến khi còn lại 3 nút thì dừng lại

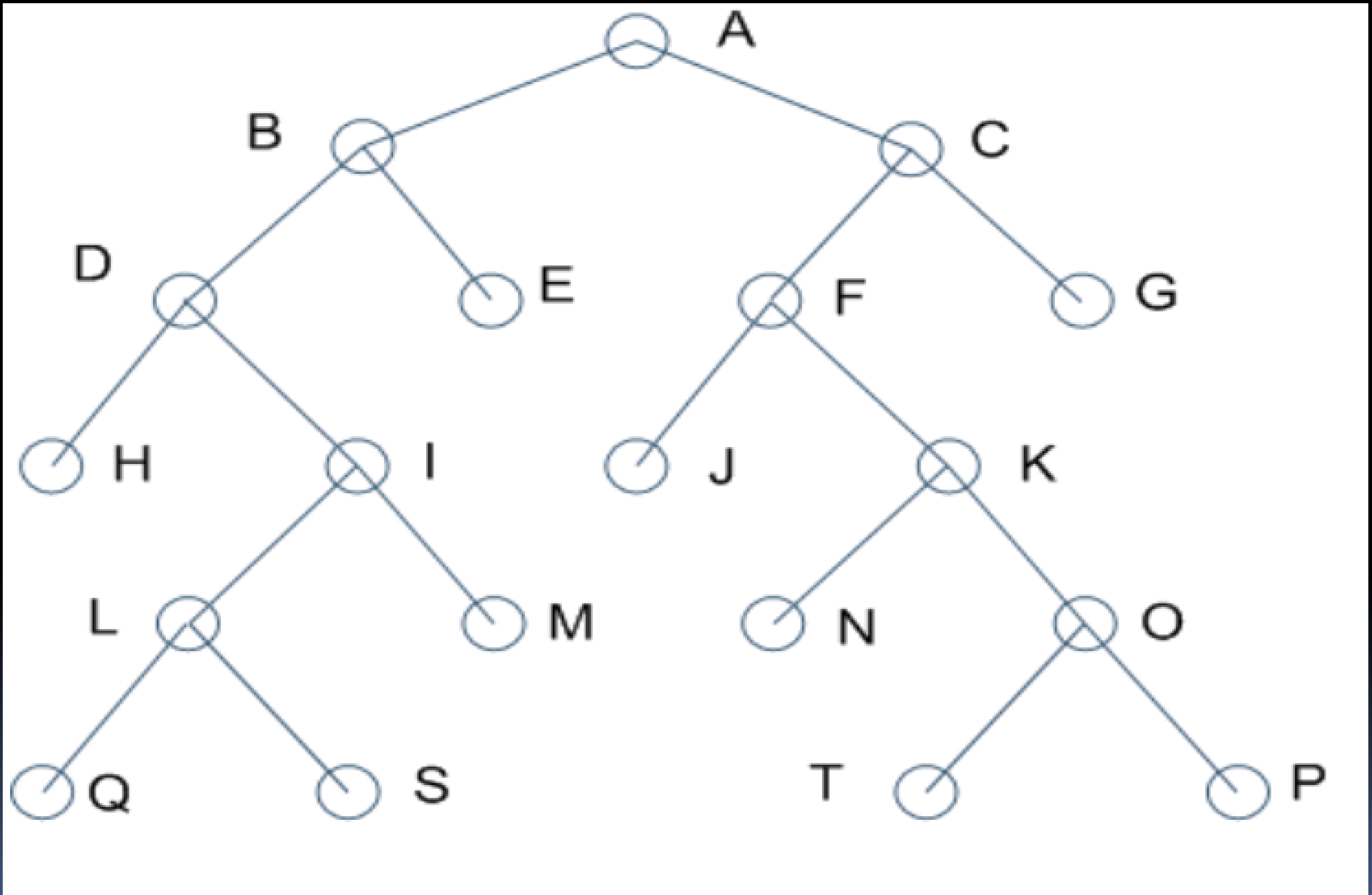
VÍ DỤ


17
12
14
9
2
3
15
8
1
13
16
10
11
4
5
6
7
A odd={1,3,5,7}
rake 3
17
12
15
9
2
13
8
1
10
11
16
4
5
6
7
rake 1,5, 7
17
12
15
8
2
13
16
4
6
A odd={2,6}
rake 2,6
17
8
15
4
16
rake 4
17
8
16

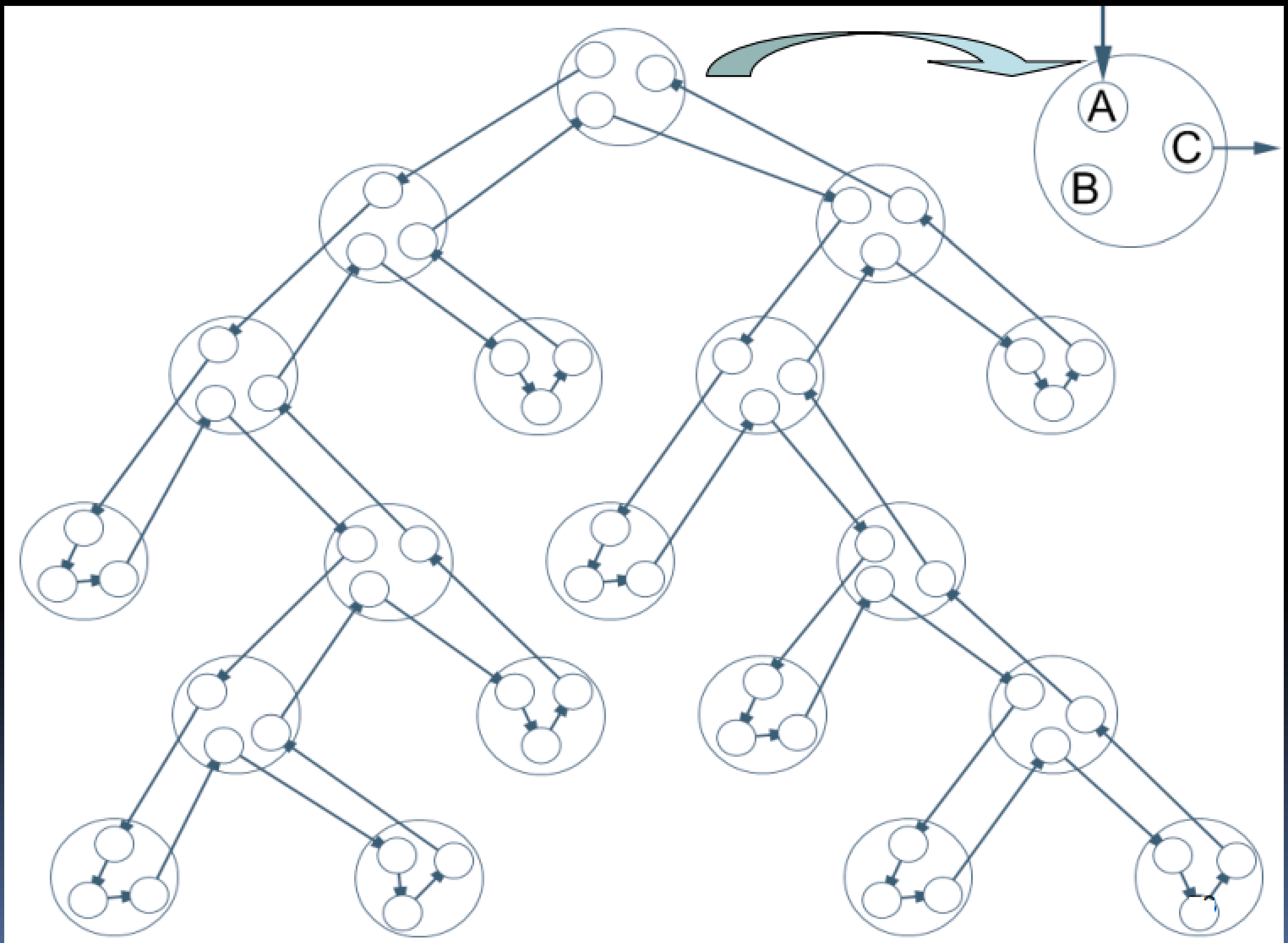
CÁC BƯỚC CHI TIẾT

- Giải bước 1 trong thuật toán:
  - Cho cây T = (V,E)
  - Hãy đánh số các lá từ trái qua phải (ngoại trừ 2 nút trái, phải ngoài cùng) theo thứ tự từ 1..n
- Cách giải quyết:
  - Sử dụng chu trình Euler
- Ví dụ minh họa với cây nhị phân (mỗi nút có đúng 2 nút con)

ĐÁNH THỨ TỰ CÁC LÁ TỪ TRÁI – PHẢI



CÁC BƯỚC CHI TIẾT

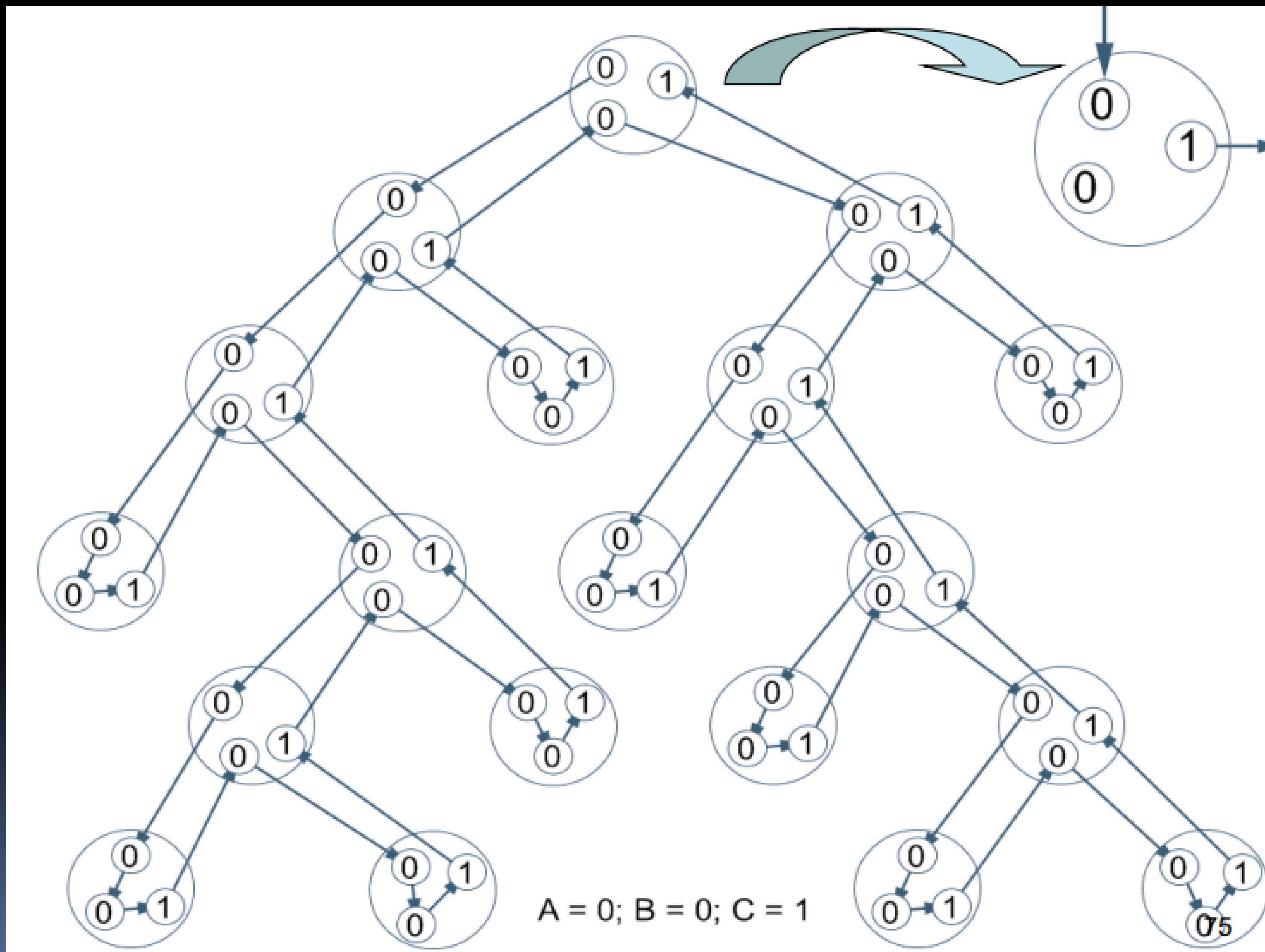


XÁC ĐỊNH CÁC NÚT LÁ

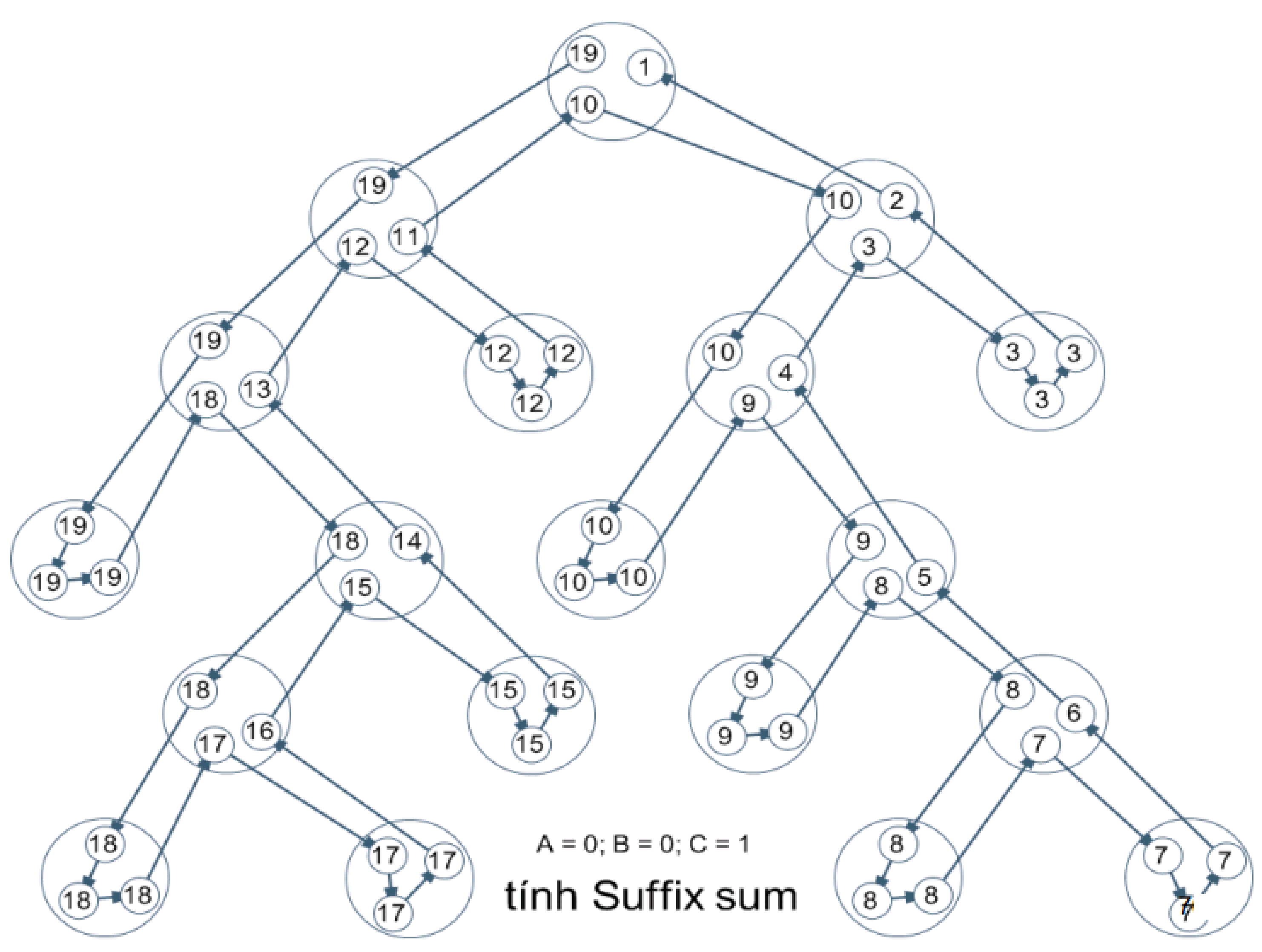
- Xây dựng chu trình Euler trên cây
- Tại mỗi nút v: v[A]=0; v[B]=0; v[C]=1
- Tính Suffix Sum đối với các nút trên danh sách sinh ra từ chu trình Euler
- Nút lá là nút có đặc điểm: giá trị Suffix tại các nút con bằng nhau: v[A] = v[B] = v[C]
- Từ hình vẽ xác định các lá:

  [H Q S M E J N T P G]

XÁC ĐỊNH CÁC NÚT LÁ



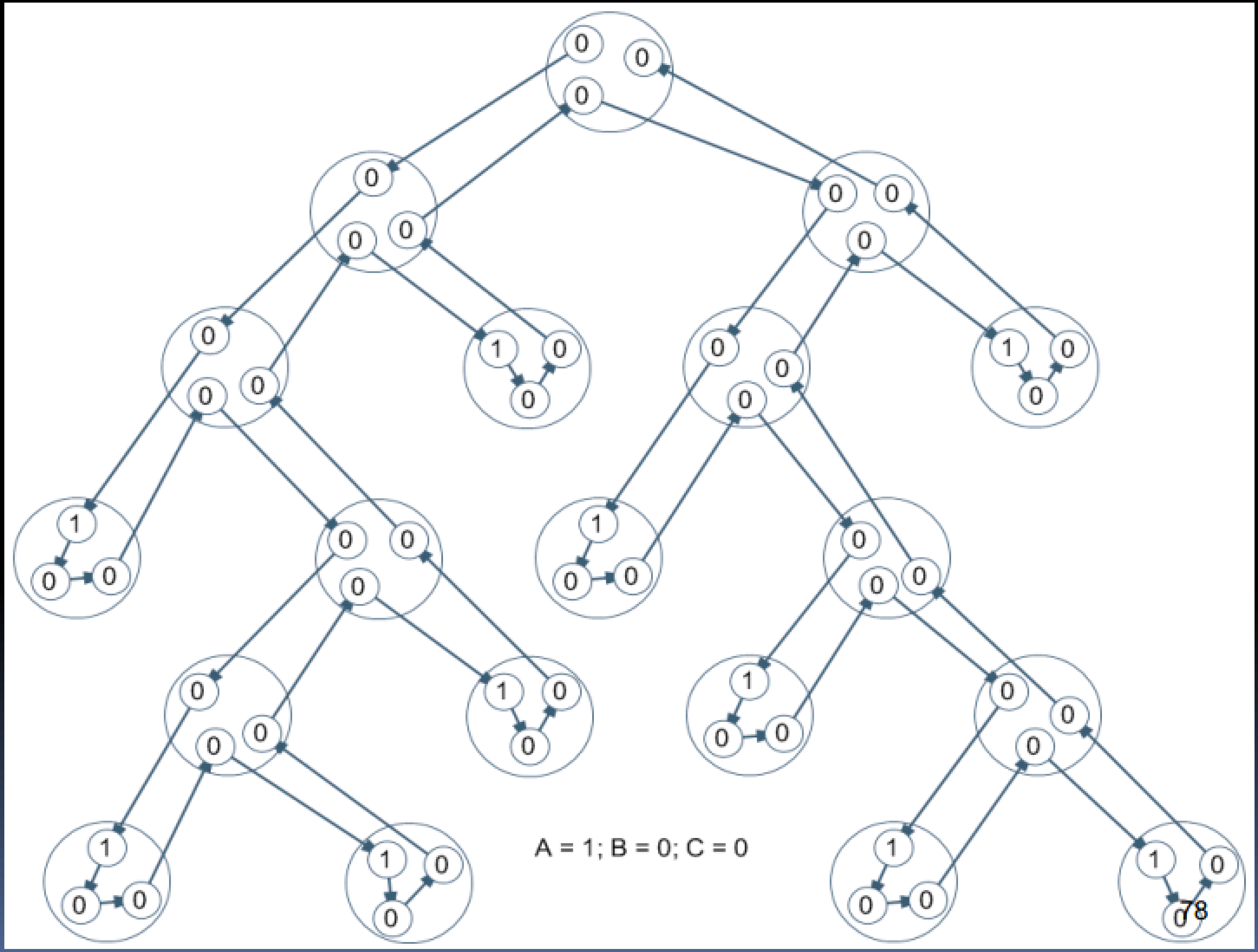
XÁC ĐỊNH CÁC NÚT LÁ



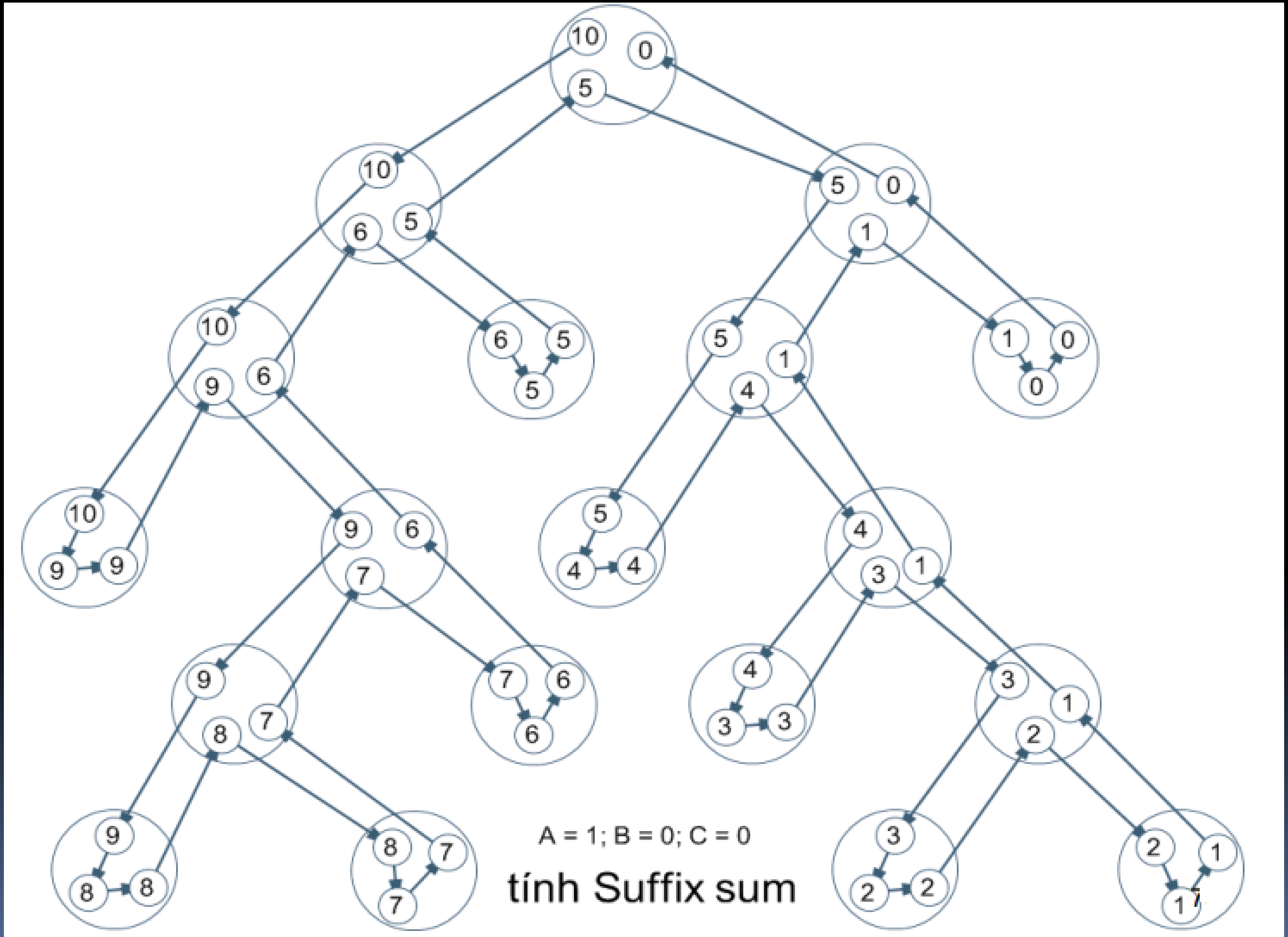
ĐẶT THỨ TỰ CHO CÁC LÁ

- Đặt A = 1; B = 0; C = 0
- Tính Suffix sum cho các nút trên danh sách sinh ra từ chu trình Euler.
- Thứ tự các lá sắp xếp từ phải qua trái thông qua giá trị v[A]
- Có thể đánh số từ trái qua phải bằng công thức |số lá| - v[A] + 1
- Lưu trữ các lá ngoại trừ các lá ngoài cùng bên trái và bên phải vào mảng Z[1..n]

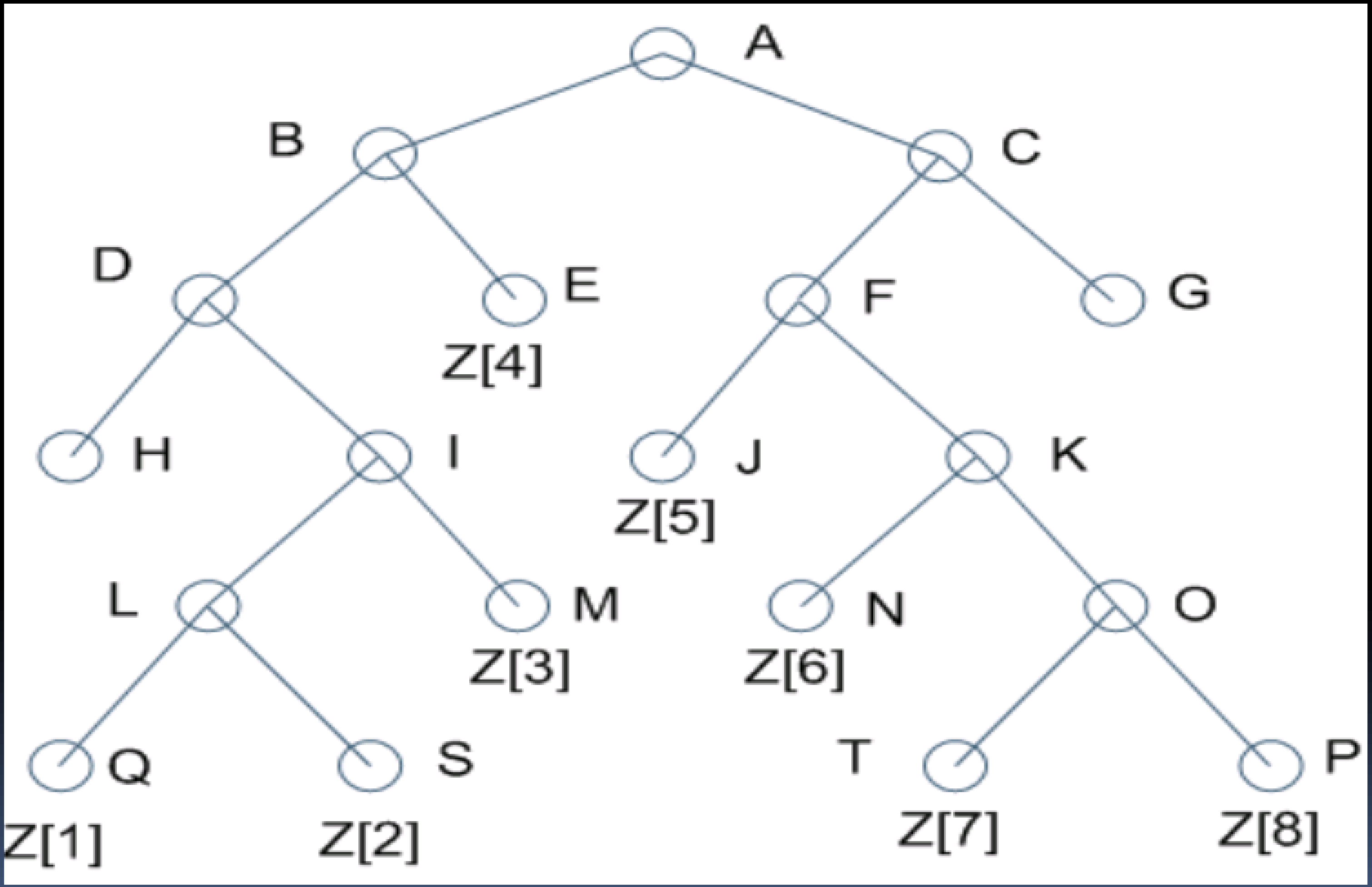
ĐẶT THỨ TỰ CHO CÁC LÁ



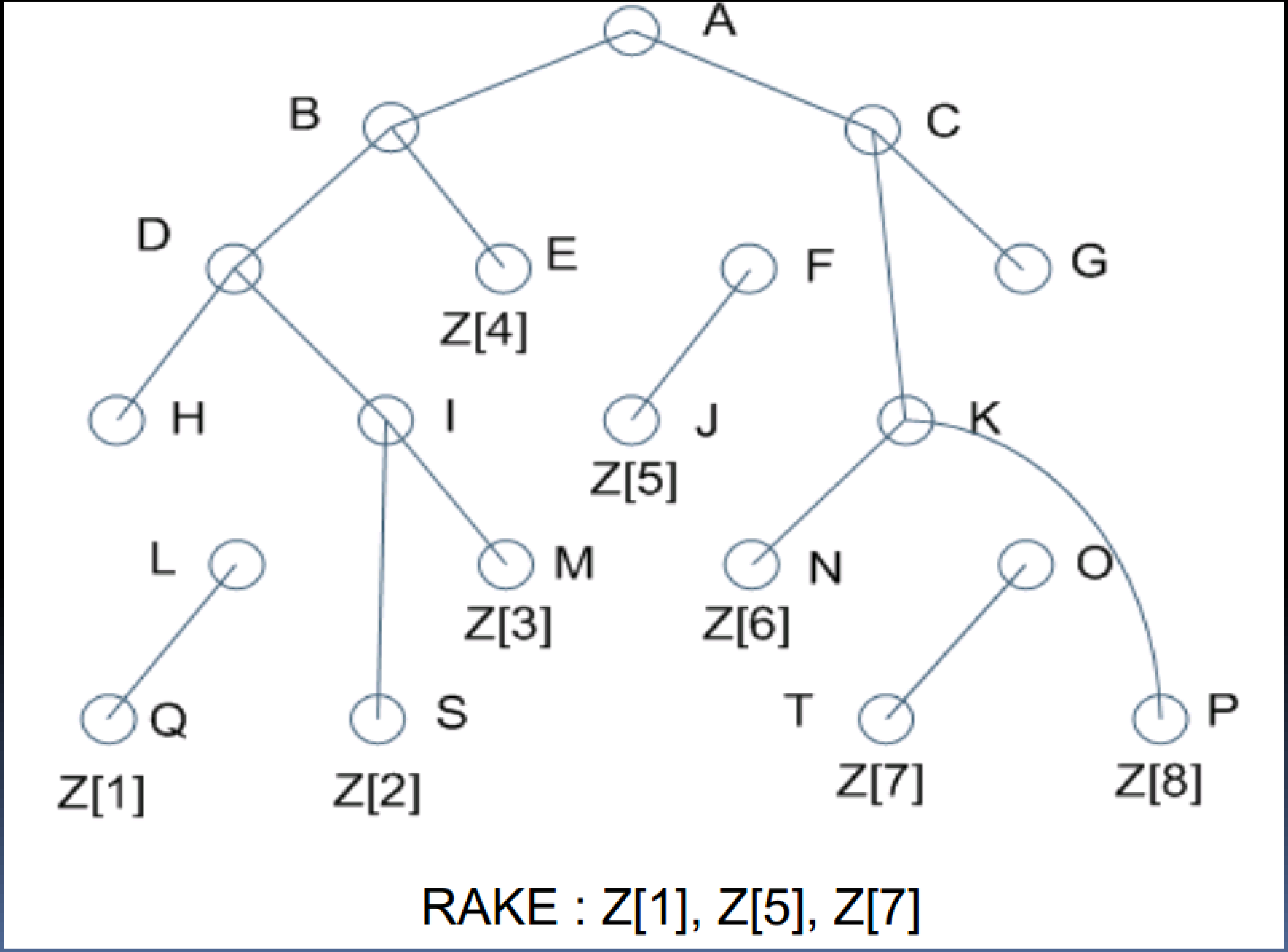
ĐẶT THỨ TỰ CHO CÁC LÁ



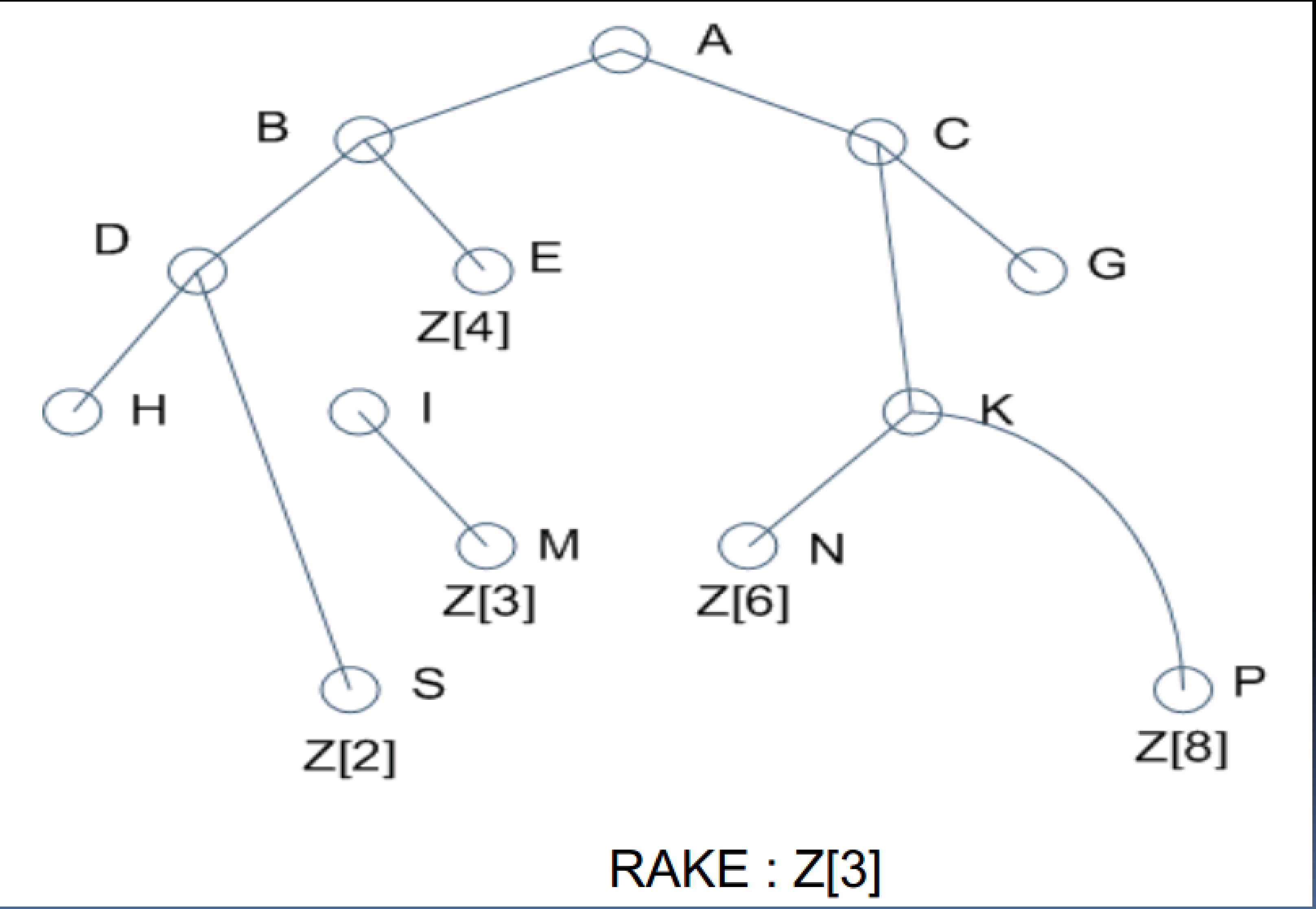
ĐÁNH THỨ TỰ TỪ TRÁI – PHẢI



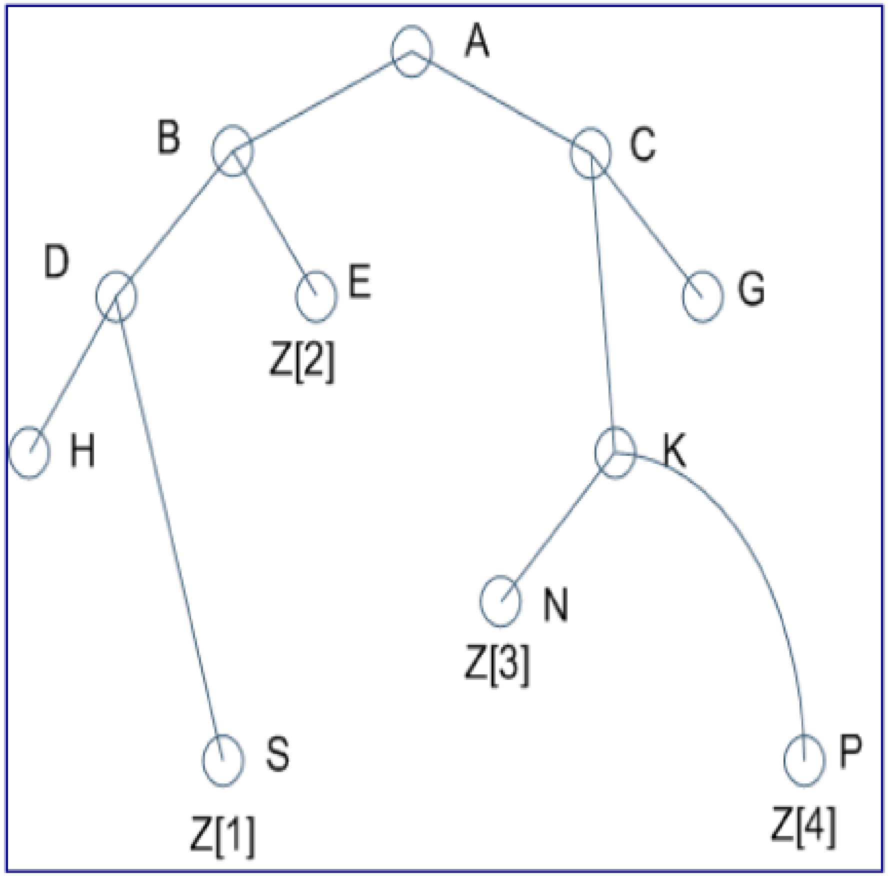
TREE CONTRACTION B1.1



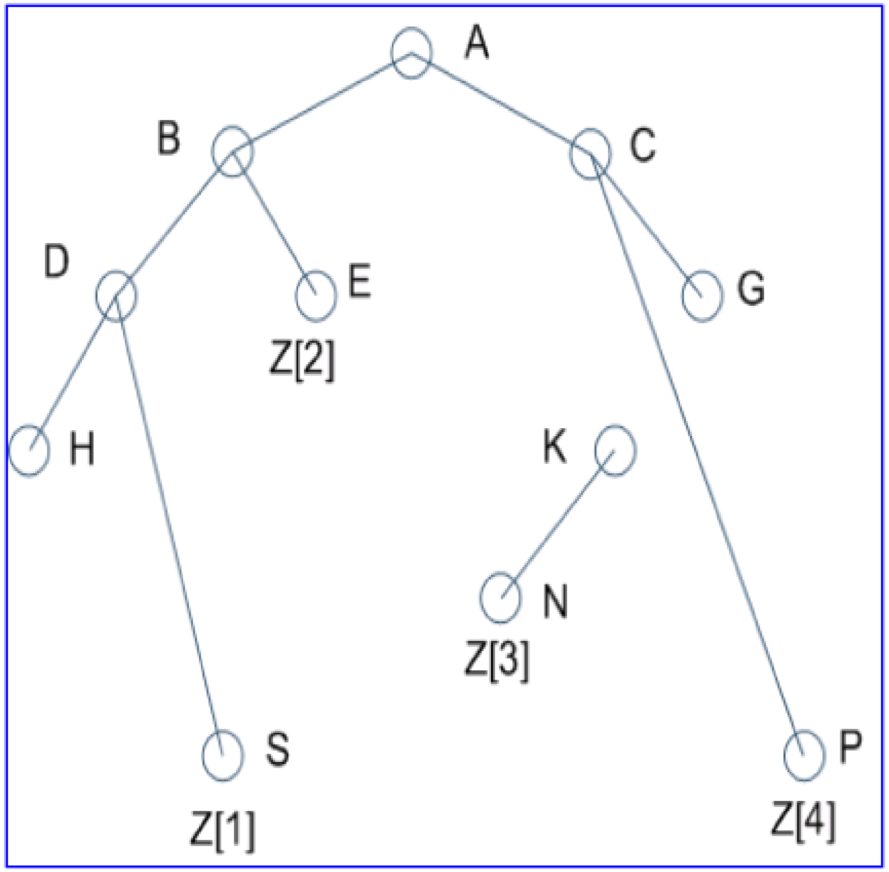
TREE CONTRACTION B1.2



TREE CONTRACTION (B1.2,B2.1)

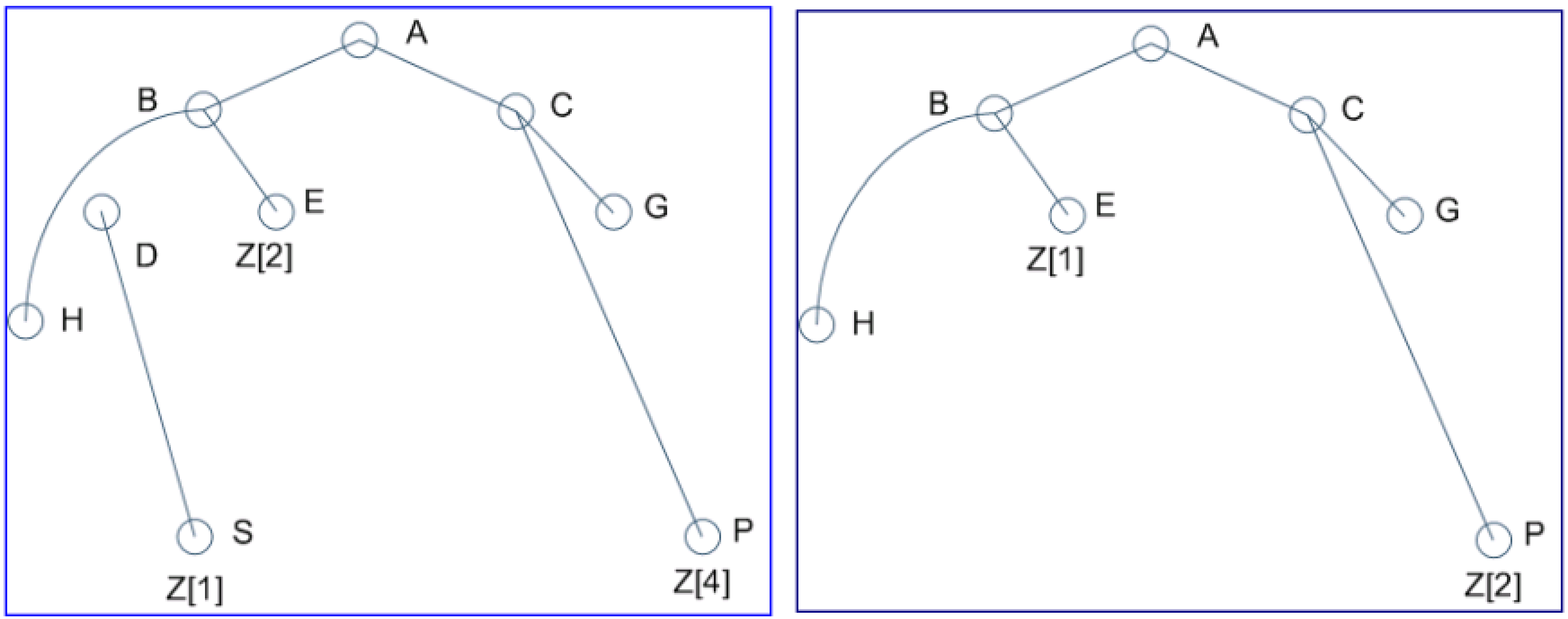


Gán Z = Z[k] với k chẵn



RAKE: Z[3]

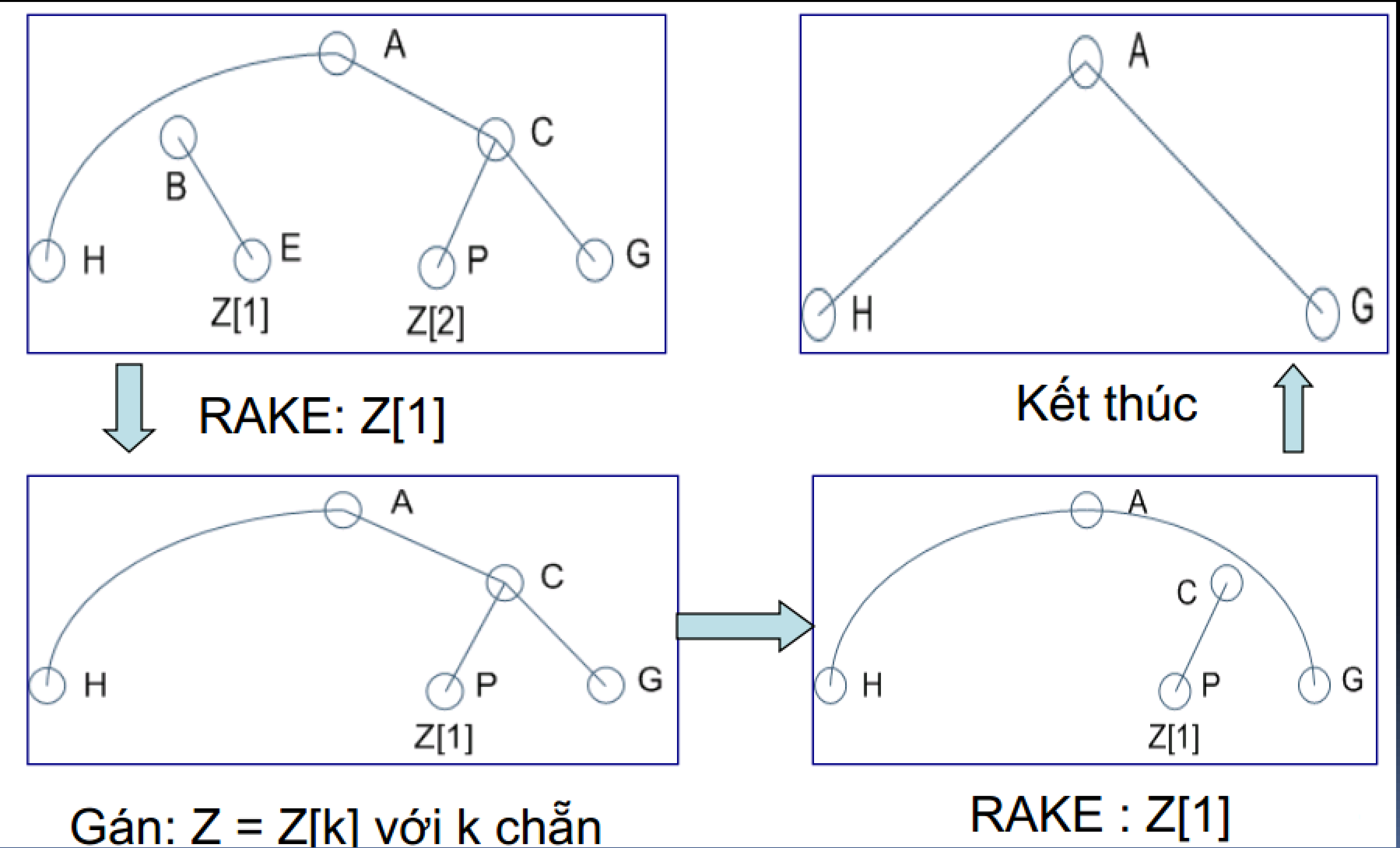
TREE CONTRACTION (B2.2,B2.3)



RAKE : Z[1]

Gán : Z = Z[k] với k chẵn

TREE CONTRACTION (B3.1,B3.2)



HẾT BÀI