CSE504(Sec1)

Generating all distributions of objects to bins

Muhammad Abdullah Adnan and Md. Saidur Rahman

Presented By

GROUP-3
K. M. Safin Kamal (2024-1-96-006)

Submitted To

Dr. Md. Saidur Rahman
Professor, Adjunct Faculty,Dept of CSE
East West University

Problem Statement

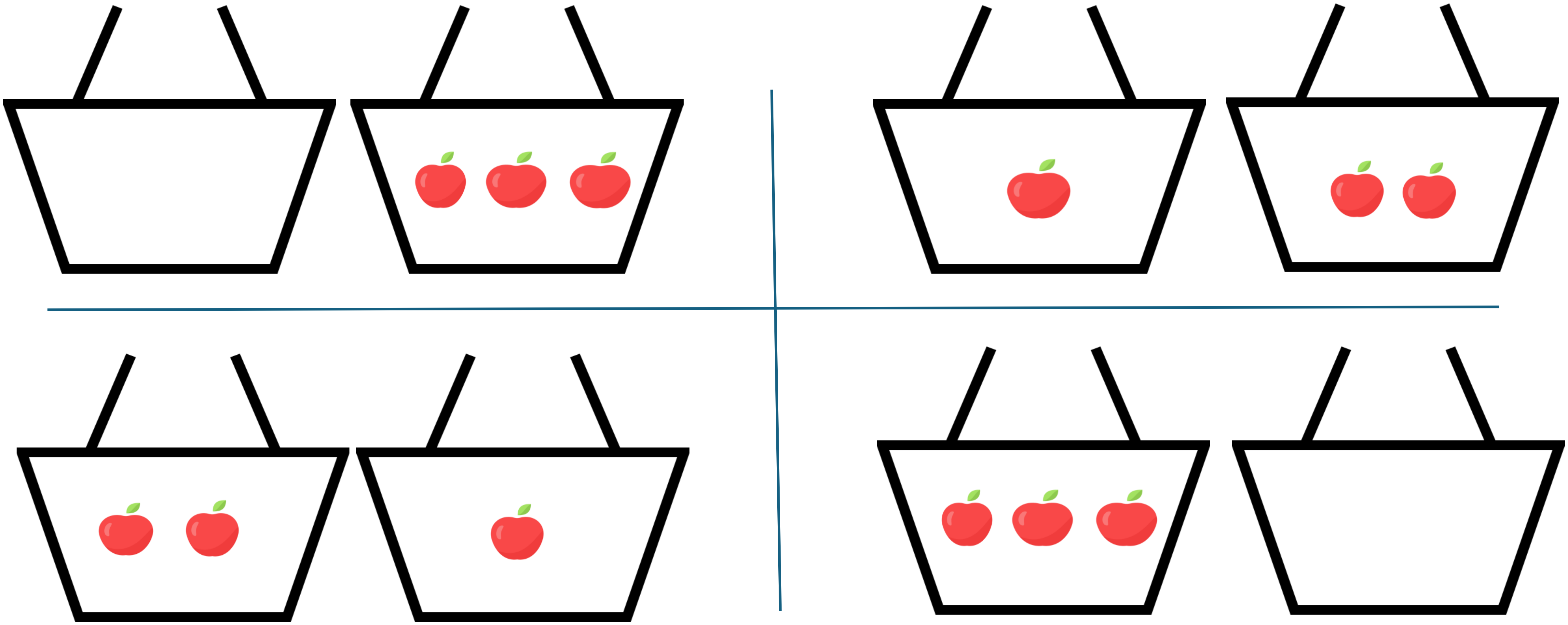
Can we generate all distributions of identical objects to bins in O(1) ?

Real world example of Distributing Object into bins

m= Number of bins = 2

n= Number of objects = 3

Number of distributions = **$(n + m - 1)!/n!(m - 1)!$**



The problems with Klingsberg’s algorithm

- Cannot generate each solution in O(1)
- Generates solutions in constant time on average
- Klingsberg’s method requires searching for the second non-zero element in the sequence for solutions that have a non-zero first elements
- Inefficient in the generation process

The Efficient Algorithm

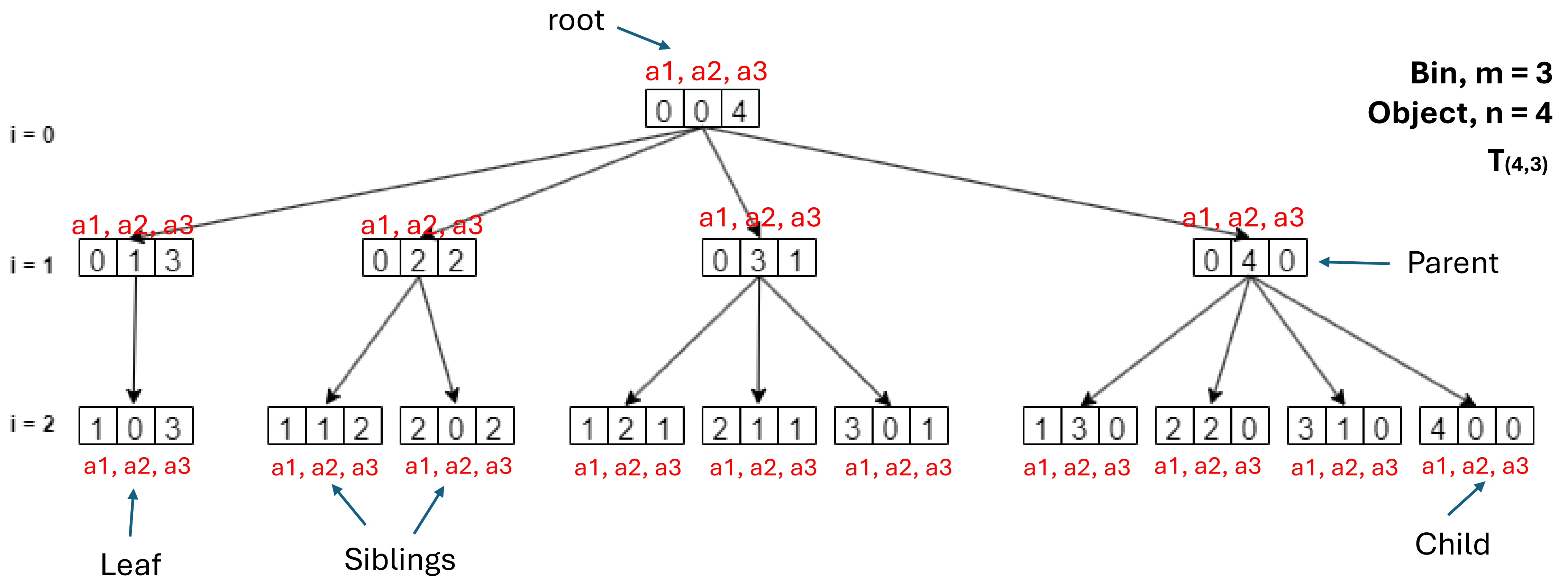
- **Constant Time Generation:** Generates each distribution in constant time (in ordinary sense)
- **Efficient Traversal:** Efficient tree traversal method to ensure each distribution is generated quickly.
- **Space Complexity:** $O(m \lg n)$
- **Optimization:** Reduce non-generation steps
- **No repetition.**
- **Generates in specific order.**

Application of the algorithm

- **Automation in Machines**
- **Computer Networks**
- **Client-Server Architecture**
- **Combinatorial Problems**

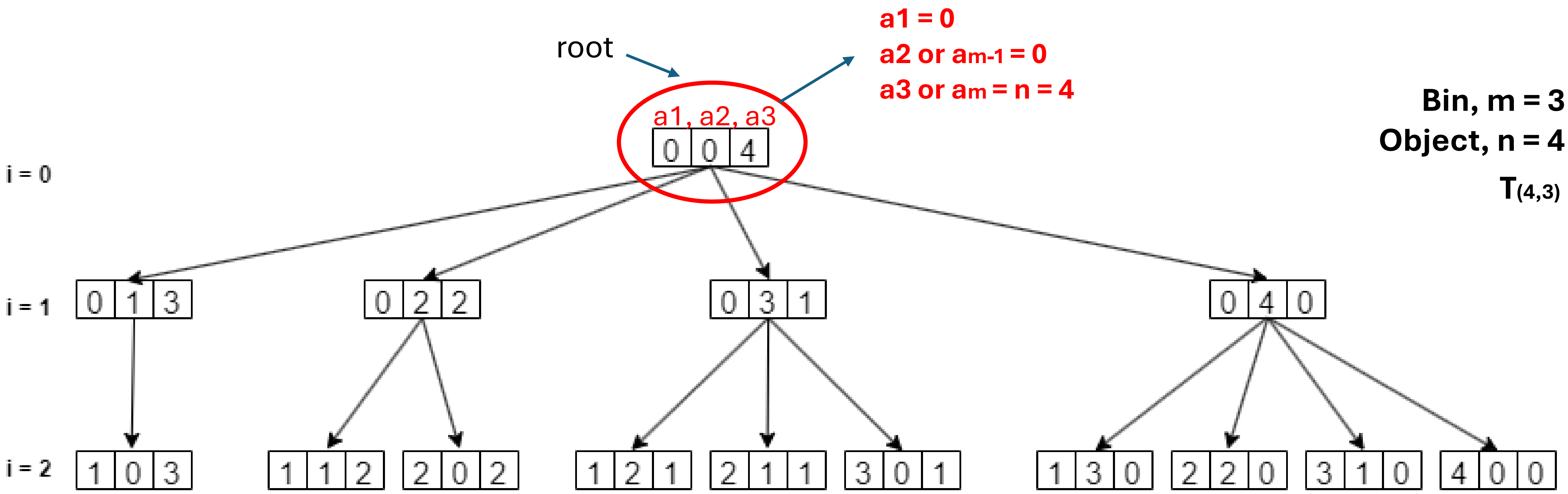
The family tree of distributions

We define a tree structure $T_{n,m}$ among the distributions in D(n, m). Each node of $T_{n,m}$ represents a distribution (a1, a2,...,am) ∈ D(n, m). If there are m bins then there are m levels $T_{n,m}$.



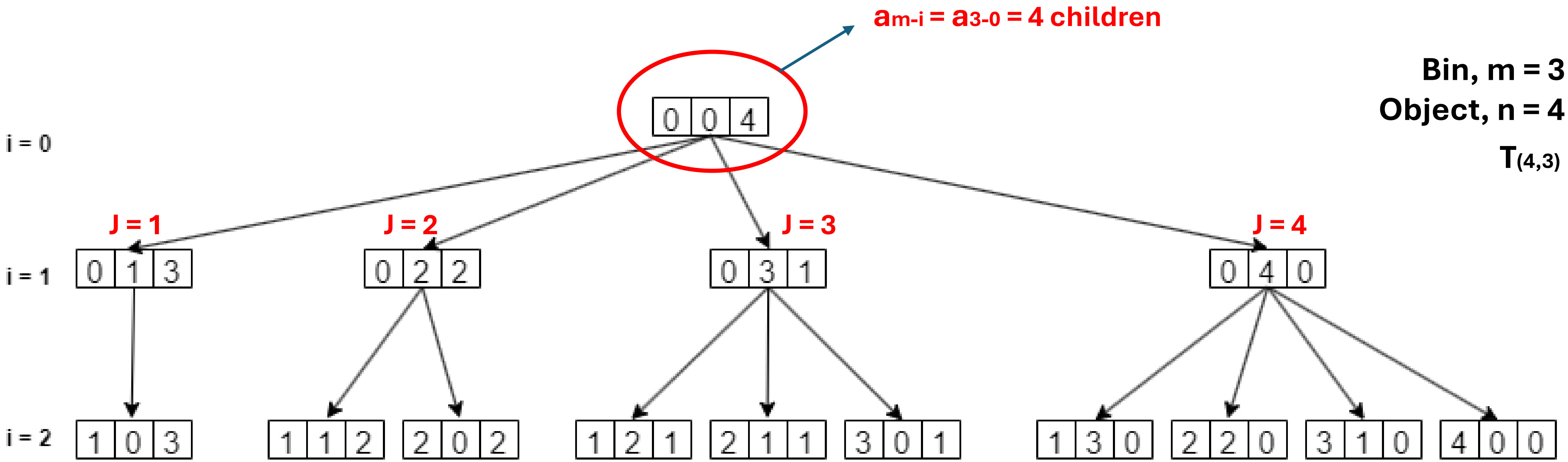
The family tree of distributions (cont.)

$T_{n,m}$ is a rooted tree we need a root, and the root is a node at level 0. we can observe that a node is at level 0 in $T_{n,m}$ and $\mathbf{a_1 = a_2 = \cdots = a_{m-1} = 0}$ **and** $\mathbf{a_m = n}$.



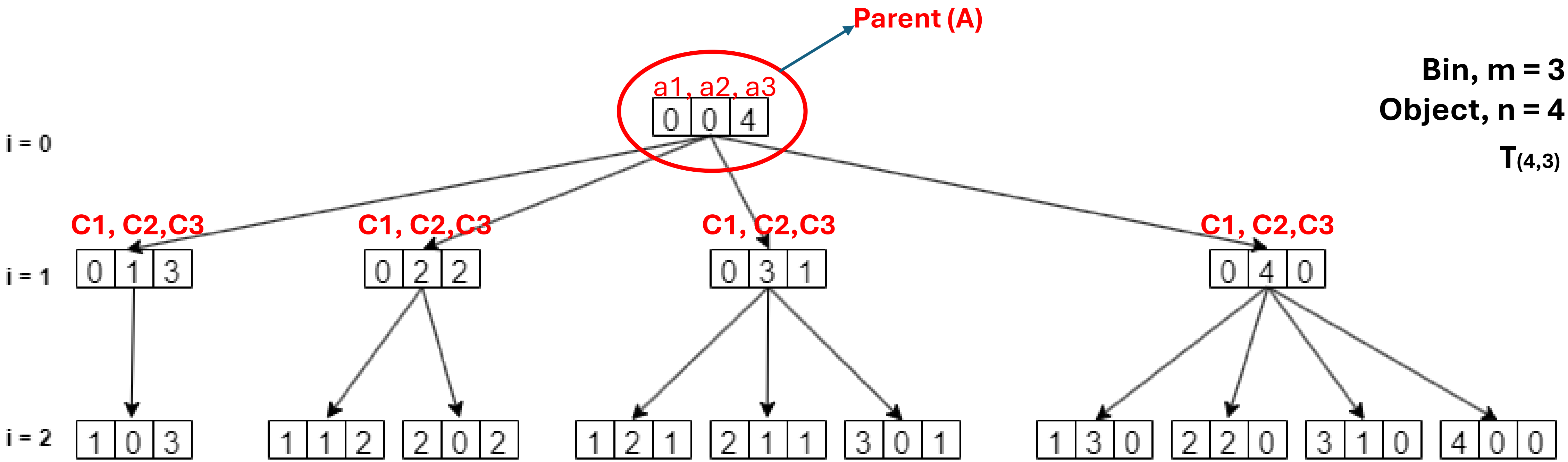
Parent–child relationship

The number of children a parent has is equal to $\mathbf{a_{m-i}}$



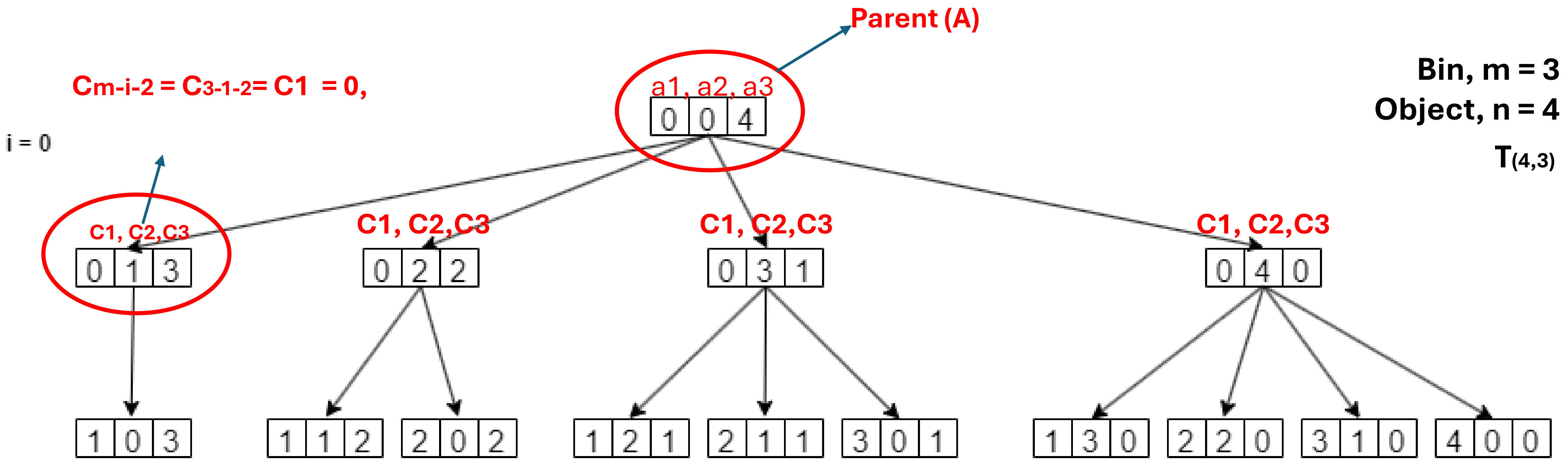
Parent–child relationship (cont.)

Let $C_j(A) \in D(n, m)$ be the sequence of j th child, $1 \le j \le a_{m-i}$ of A. Note that A is in level i of $T_{n,m}$ and $C_j(A)$ will be in level $i + 1$ of $T_{n,m}$. We define the sequence for $C_j(A)$ as $(c_1, c_2, \ldots, c_{m-i-1}, c_{m-i}, \ldots, c_m)$, where $0 \le I < m$, **$c_1 = c_2 = \cdots = c_{m-i-2} = 0$ and $c_{m-i-1} = j$, $c_{m-i} = a_{m-i} - j$ and $c_k = a_k$ for $m - i + 1 \le k \le m$**



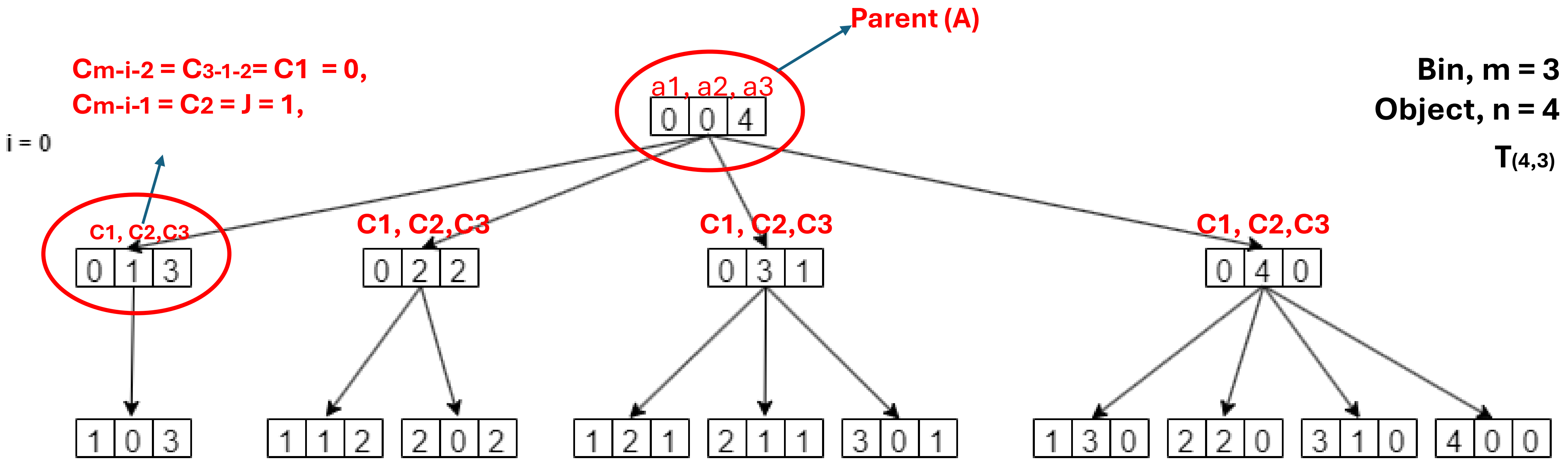
Parent–child relationship (cont.)

Let Cj (A) ∈ D(n, m) be the sequence of j th child, $1 \le j \le a_{m-i}$ of A. Note that A is in level i of Tn,m and Cj (A) will be in level i + 1 of Tn,m. We define the sequence for Cj (A) as (c1, c2,...,cm−i−1, cm−i,...,cm), where $0 \le I < m$, C**1 = C2 =···= Cm−i−2 = 0 and Cm−i−1 = j, Cm−i = a_{m-i} − j and Ck = a_k for m − i + 1 ≤ k ≤ m**



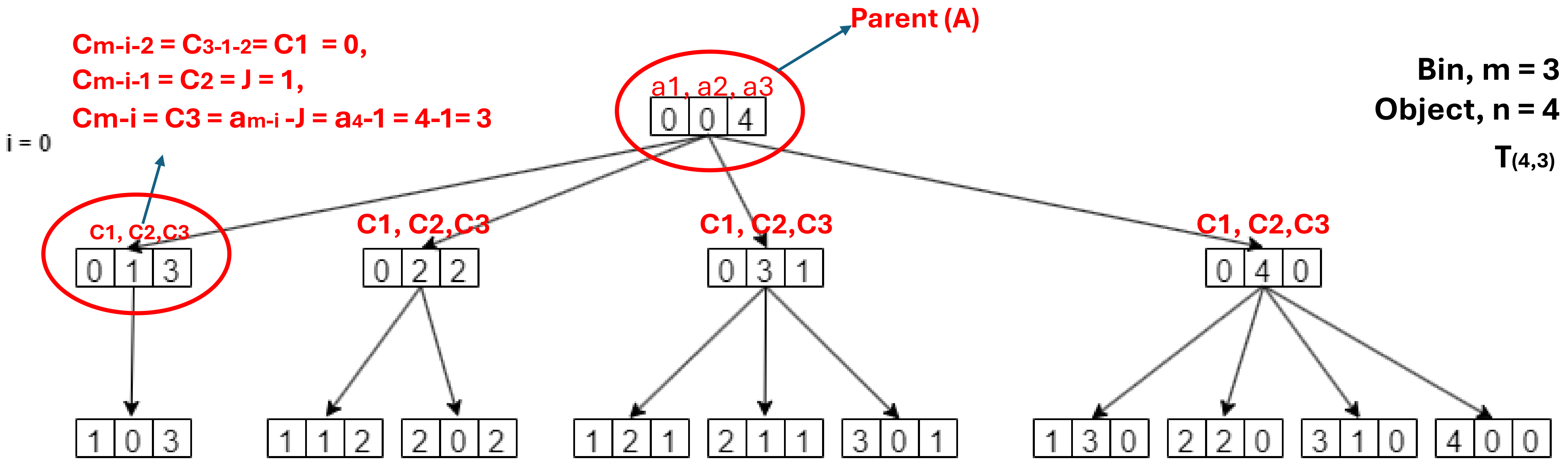
Parent–child relationship (cont.)

Let Cj (A) ∈ D(n, m) be the sequence of j th child, 1 ≤ j ≤ am−i of A. Note that A is in level i of Tn,m and Cj (A) will be in level i + 1 of Tn,m. We define the sequence for Cj (A) as (c1, c2,...,cm−i−1, cm−i,...,cm), where 0 ≤ I < m , C**1 = C2 =···= Cm−i−2 = 0 and Cm−i−1 = j, Cm−i = am−i − j and Ck = ak for m − i + 1 ≤ k ≤ m**



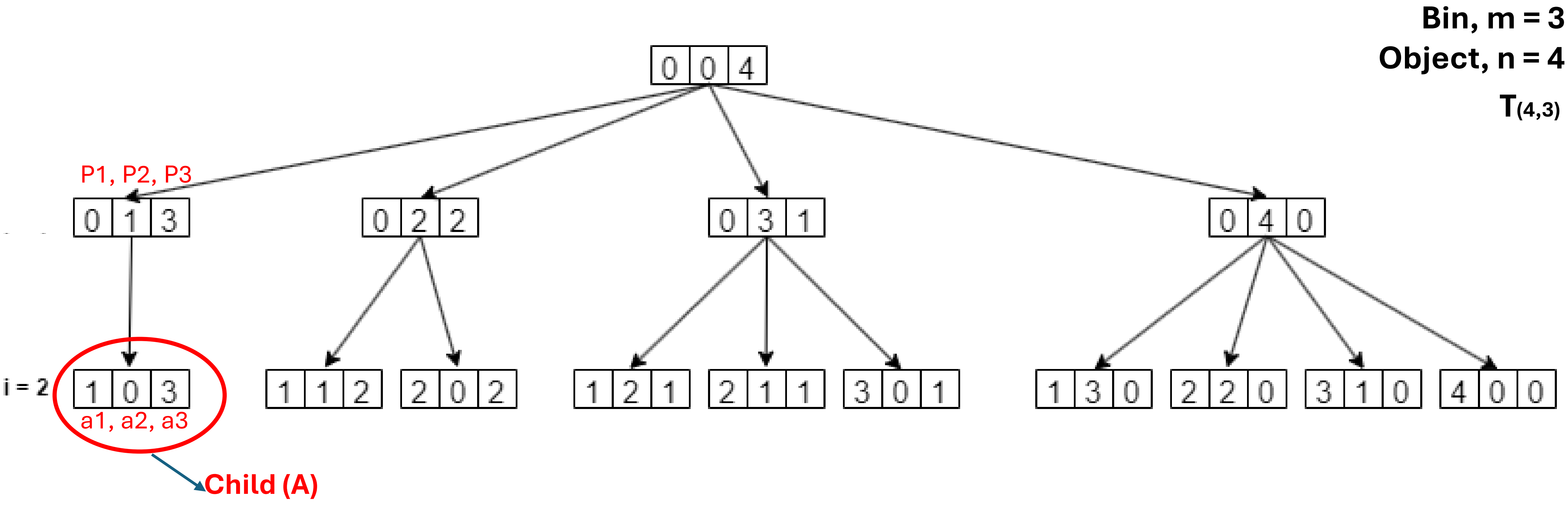
Parent–child relationship (cont.)

Let Cj (A) ∈ D(n, m) be the sequence of j th child, 1 ≤ j ≤ am−i of A. Note that A is in level i of Tn,m and Cj (A) will be in level i + 1 of Tn,m. We define the sequence for Cj (A) as (c1, c2,...,cm−i−1, cm−i,...,cm), where 0 ≤ I < m , C**1 = C2 =···= Cm−i−2 = 0 and Cm−i−1 = j, Cm−i = am−i − j and Ck = ak for m − i + 1 ≤ k ≤ m**



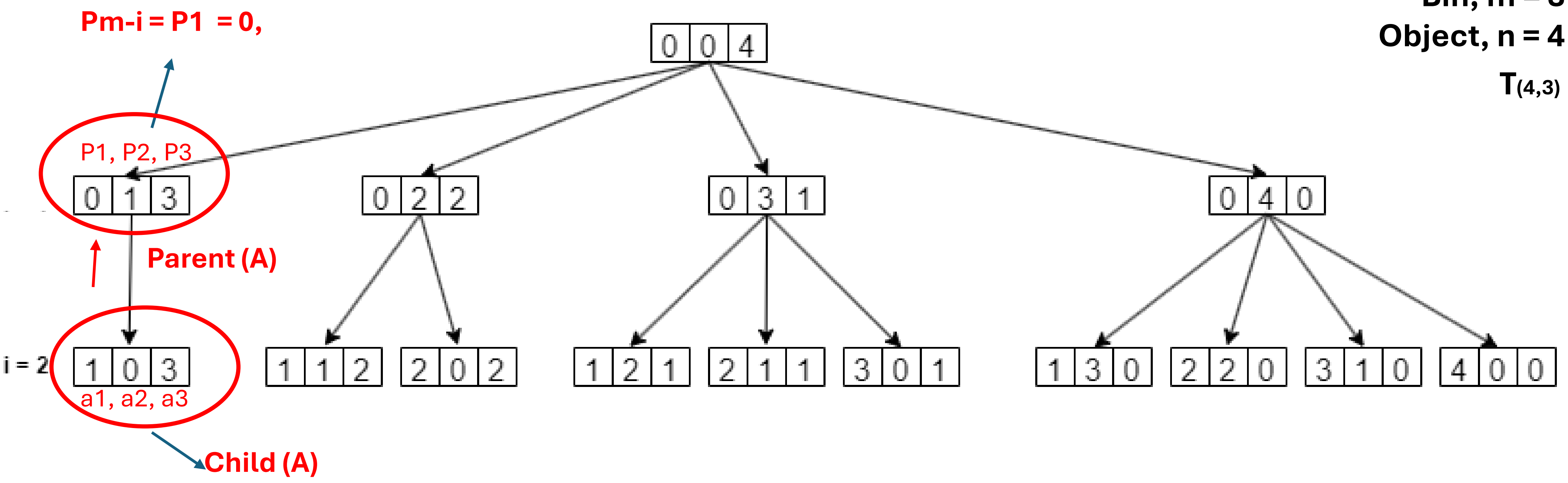
Child–parent relationship

The child–parent relation is just the reverse of parent–child relation. Let P (A) ∈ D(n, m) be the parent sequence of A. We define the sequence for P (A) as (p1, p2, ..., pm−i, pm−i+1, ..., pm) where 1 ≤ I < m, **p1 = p2 =···= pm−i = 0, pm−i+1 = am−i + am−i+1 and pj = aj for m − i + 1 < j ≤ m.**



Child–parent relationship(cont.)

The child–parent relation is just the reverse of parent–child relation. Let P (A) ∈ D(n, m) be the parent sequence of A. We define the sequence for P (A) as $(p_1, p_2, \ldots, p_{m-i}, p_{m-i+1}, \ldots, p_m)$ where $1 \leq I < m$, $\mathbf{p_1 = p_2 = \cdots = p_{m-i} = 0}$, $\mathbf{p_{m-i+1} = a_{m-i} + a_{m-i+1}}$ **and** $\mathbf{P_j = a_j}$ **for** $\mathbf{m - i + 1 < j \leq m}$**.**



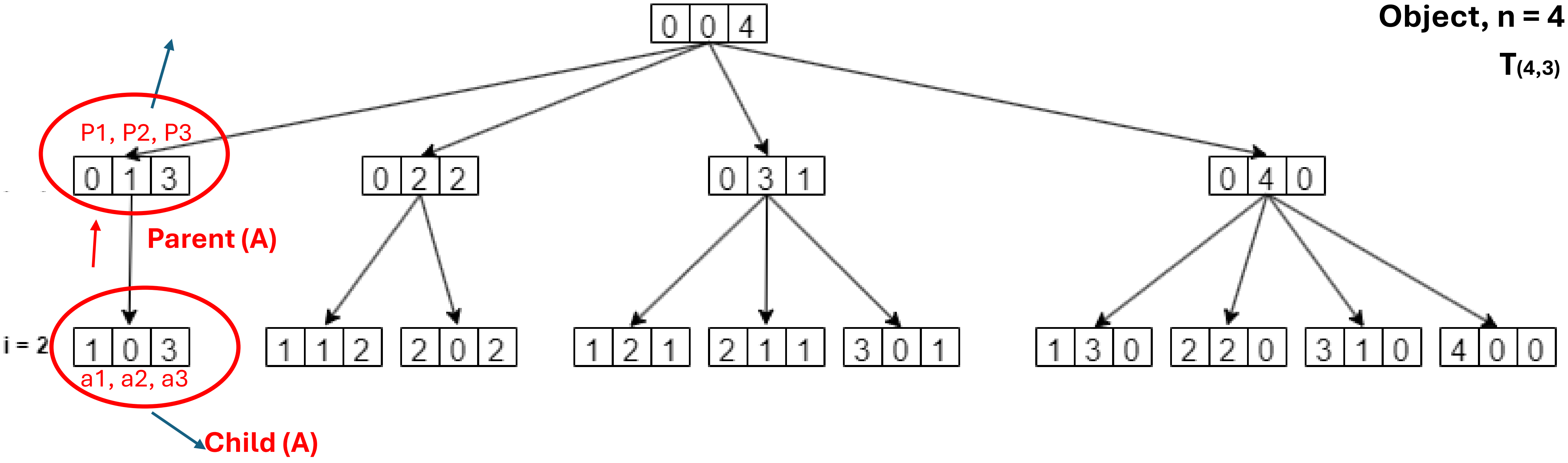
Bin, m = 3
Object, n = 4
$\mathbf{T_{(4,3)}}$

Child–parent relationship(cont.)

The child–parent relation is just the reverse of parent–child relation. Let P (A) ∈ D(n, m) be the parent sequence of A. We define the sequence for P (A) as ($p_1, p_2, \ldots, p_{m-i}, p_{m-i+1}, \ldots, p_m$) where $1 \leq l < m$, **$p_1 = p_2 = \cdots = p_{m-i} = 0$, $p_{m-i+1} = a_{m-i} + a_{m-i+1}$ and $P_j = a_j$ for $m - i + 1 < j \leq m$.**

$P_{m-i} = P_1 = 0$,
$P_{m-i+1} = P_2 = a_{m-i} + a_{m-i+1} = a_1 + a_2 = 1 + 0 = 1$

Bin, m = 3
Object, n = 4
$T_{(4,3)}$

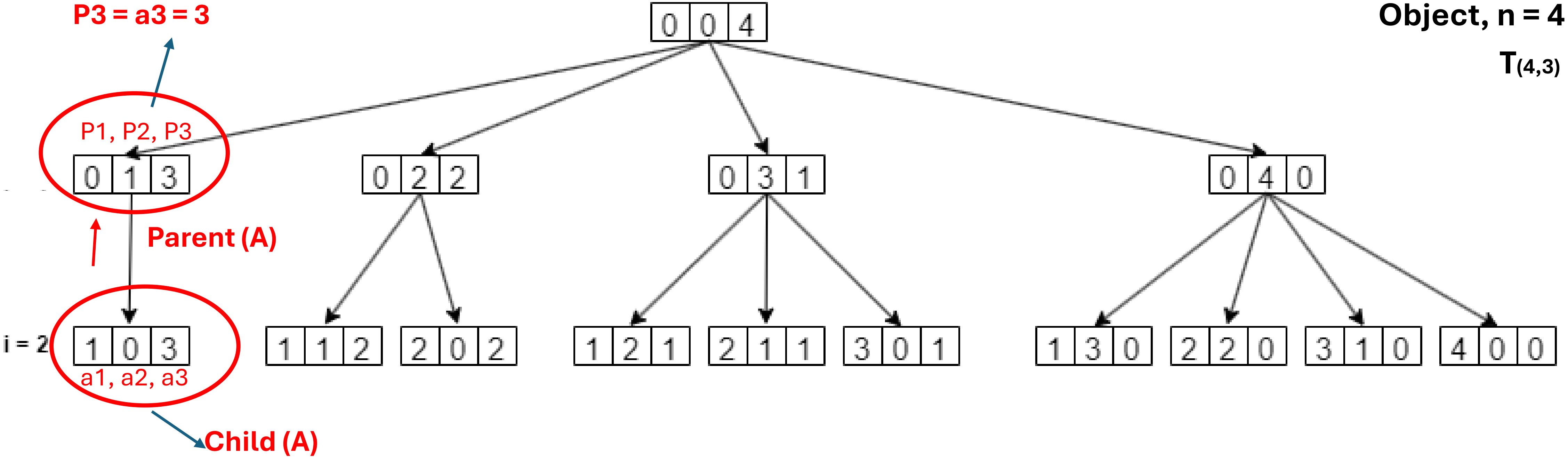


Child–parent relationship(cont.)

The child–parent relation is just the reverse of parent–child relation. Let P (A) $\in$ D(n, m) be the parent sequence of A. We define the sequence for P (A) as ($p_1, p_2, \ldots, p_{m-i}, p_{m-i+1}, \ldots, p_m$) where $1 \le I < m$, $\mathbf{p_1 = p_2 = \cdots = p_{m-i} = 0}$, $\mathbf{p_{m-i+1} = a_{m-i} + a_{m-i+1}}$ **and** $\mathbf{P_j = a_j}$ **for** $\mathbf{m - i + 1 < j \le m}$.

Pm-i = P1 = 0,
Pm-i+1 = P2= am−i + am−i+1 = a1+a2 = 1 + 0 = 1
P3 = a3 = 3

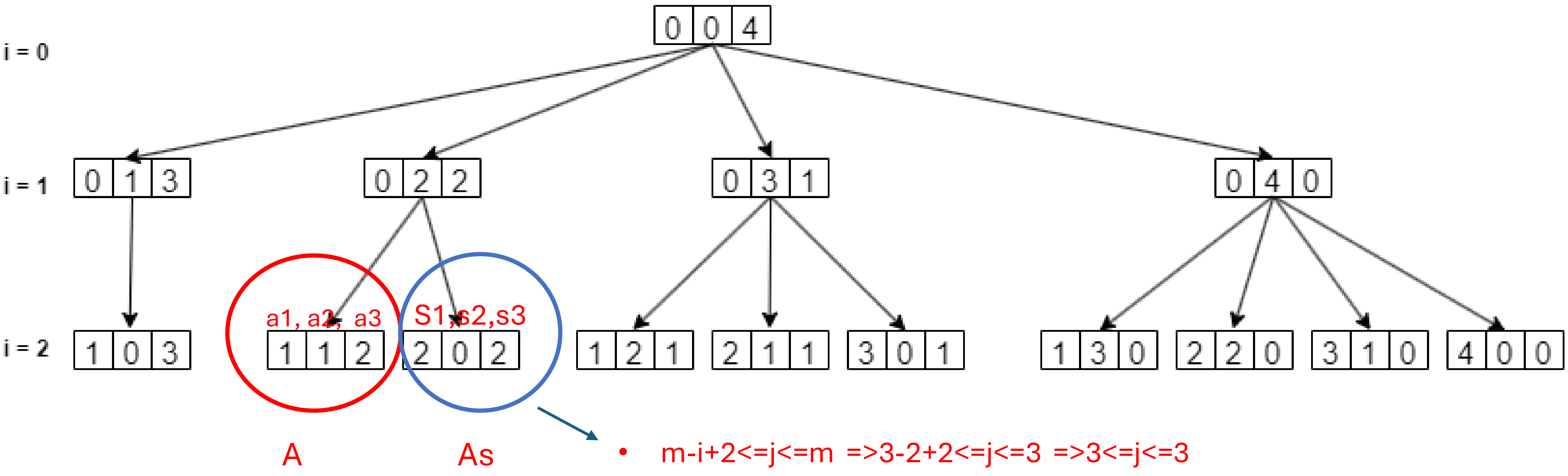
Bin, m = 3
Object, n = 4
$T_{(4,3)}$



Efficient tree traversal O(1) time

Relationship between left sibling and right sibling

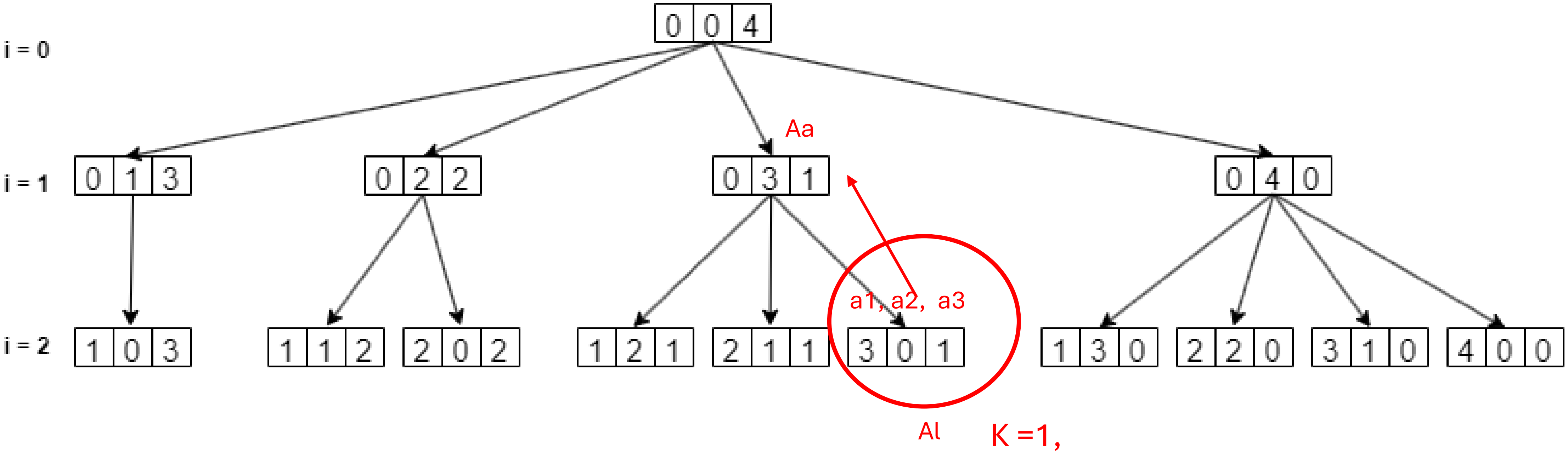
Right sibling $A_s \in D(n,m)$ of node A exists if $\mathbf{a_{m-i+1} \neq 0}$ at level i of $T_{n,m}$. the sequence for A_s as $(s_1,s_2,\ldots,s_{m-i},s_{m-i+1},\ldots,s_m), 1 \leq i < m$ where $\mathbf{s_1 = s_2 = \cdots = s_{m-i-1} = 0,\ s_{m-i} = a_{m-i+1},\ s_{m-i+1} = a_{m-i+1} - 1}$ and $\mathbf{s_j = a_j}$ for $\mathbf{m-i+2 \leq j \leq m}$.



- m-i+2<=j<=m =>3-2+2<=j<=3 =>3<=j<=3 =>s3=a3 =>**s3=2**
- sm−i+1 = am−i+1 –1 => **s2** = a2-1 = 1-1 = **0**
- sm-i =am−i+1 –1 =>**s1** = a1+1 = 1+1 =**2**

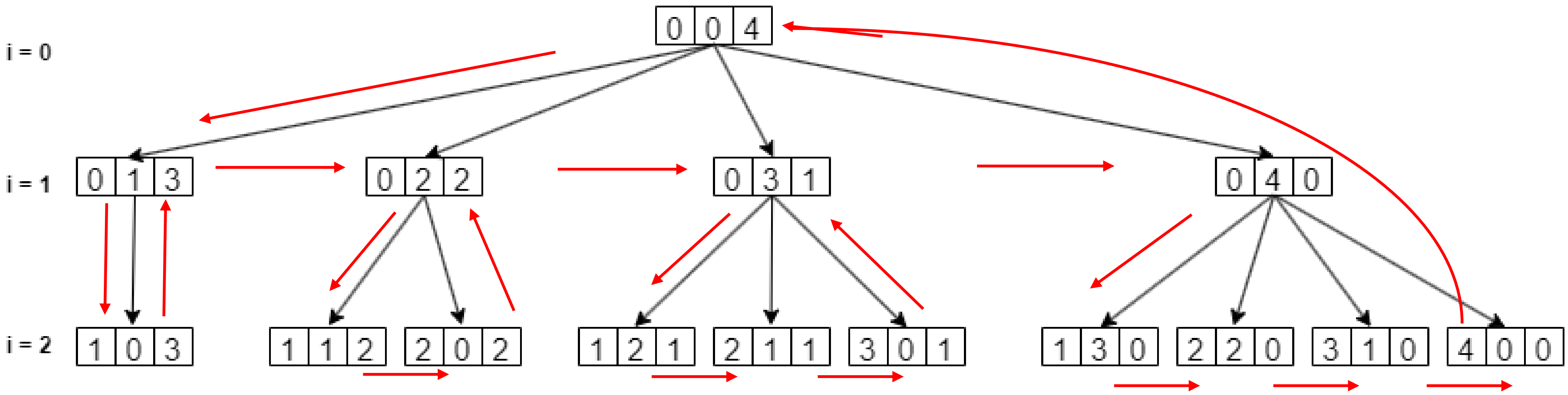
Leaf–ancestor relationship

Al = rightmost leaf, Aa= Nearest ancestor which has right siblings, k = number of consecutive 0's,Aa∈ D(n,m) of node Al exists if a2 = 0. Nearest ancestor is obtained by swapping a1 and ak+1 at level m-1-k



K =1,
i=m-1-k = 3-1-1 = 1
Aa = swapping a1 and ak+1 then Aa = 0,3,1

Efficient tree traversal



Pseudo Code

Algorithm Find-All-Distributions(n, m)
$\{A_r$ is the root sequence, S indicates the current stack $\}$
begin
  Find-All-Child-Distributions($A_r = (0, \ldots, 0, n), 0, S$);
end.

Call the function by root

Procedure Find-All-Child-Distributions$(A = (a_1, a_2, \ldots, a_m), i, S)$
$\{$ A is the current sequence, i indicates the current level, A_c is the child sequence, A_s is the right sibling sequence, A_a is the ancestor sequence and S indicates the current stack $\}$
begin
  Output A {Output the difference from the previous distribution}
  if $a_1 = 0$ then
    begin
      $\{A$ has child$\}$
      if $a_{m-i} - 1 = 0$ then
        if $a_{m-i+1} \neq 0$ then **Push**$(1, S)$;
        else Top(S)=**Top**$(S)+1$;
      Find-All-Child-Distributions($A_c = (a_1, a_2, \ldots, a_{m-i-2}, 1, (a_{m-i} - 1), \ldots, a_m), i + 1, S)$;
    end

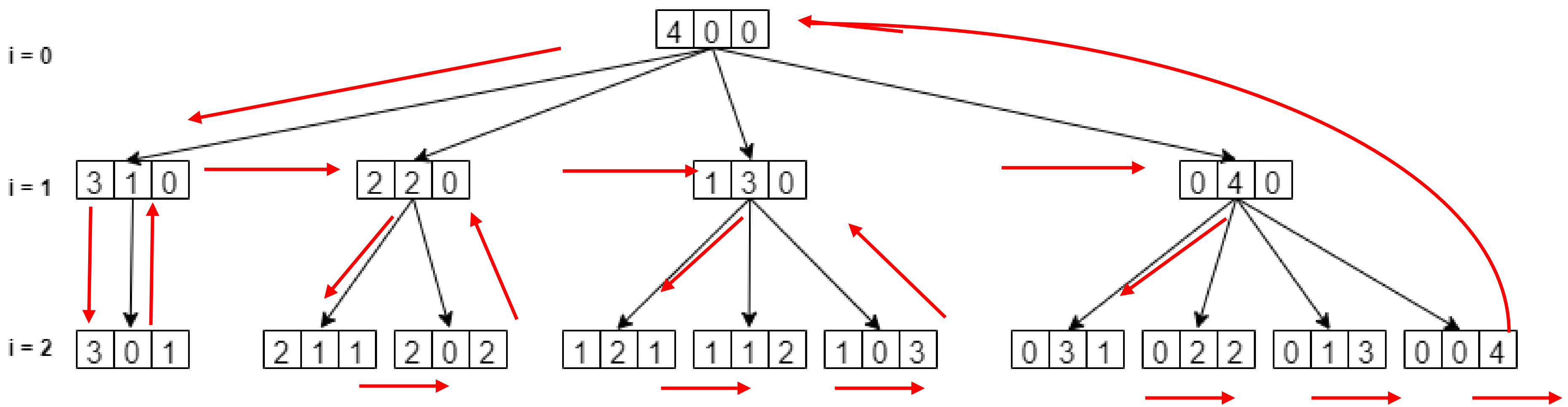
Find Child

```
else if a_2 ≠ 0 then
  begin
    {A has right sibling}
    if a_2 − 1 = 0 then
      if a_3 ≠ 0 then Push(1, S);
      else Top(S)=Top(S)+1;
    Find-All-Child-Distributions( A_s = ((a_1 + 1), (a_2 − 1), . . . , a_m), i, S)
  end
else
  begin
    k=Pop(S);
    Swap(a_1, a_{k+1}); {Generate the ancestor A_a of A}
    if k = m − 1 then return ; {A_a is the root}
    else
    begin
      {A_a has right sibling}
      if a_{k+2} − 1 = 0 then
        if a_{k+3} ≠ 0 or k + 2 = m then Push(1, S);
        else Top(S)=Top(S)+1;
      Find-All-Child-Distributions(A_as = (a_1, a_2, . . . , (a_{k+1} + 1), (a_{k+2} − 1), . . . , a_m), m − 1 − k, S);
    end
  end
end;
```

Find Right Sibling

Find Ancestor's Right Sibling

Anti-lexicographical Order



Conclusion

This paper presents a simple, efficient algorithm for generating all distributions in D(n, m) with specified order, including anti-lexicographic, operating in constant time.

Reference

- [1] A.V. Aho and J.D. Ullman, Foundation of Computer Science, Computer Science Press, New York, 1995.
- [2] D. Avis and K. Fukuda, Reverse search for enumeration, Discrete Appl. Math. 65 (1996), pp. 21–46.
- [3] T.I. Fenner and G. Loizou, A binary tree representation and related algorithms for generating integer partitions, Comp. J. 23 (1979), pp. 332–337.
- [4] S. Kawano and S. Nakano, Constant time generation of set partition, IEICE Trans. Fundam. E88-A(4) (2005), pp. 930–934.
- [5] P. Klingsberg, A gray code for compositions, J. Algorithms 3 (1982), pp. 41–44. Downloaded By: [Saidur Rahman, Md.] At: 12:43 8 March 2009 392 M.A. Adnan and M.S. Rahman
- [6] S. Nakano and T. Uno, Constant time generation of trees with specified diameter, Proc. of WG 2004, LNCS 3353 (2004), pp. 33–45.
- [10] A.S. Tanenbaum, Computer Networks, Prentice Hall, Upper Saddle River, New Jersey, 2002.
- [11]Modern Operating Systems, Prentice Hall, Upper Saddle River, New Jersey, 2004.
- [12] K. Yamanaka, et al., Constant time generation of integer partitions, IEICE Trans. Fundam. E-90-A (5) (2007), pp. 888–895.
- [13] A. Zoghbi and I. Stojmenovic, Fast algorithm for generating integer partitions, Intern. J. Comput. Math. 70 (1998), pp. 319–332.

Thank You