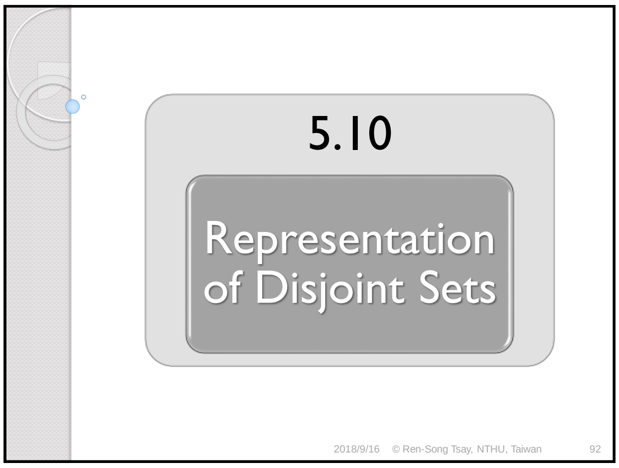
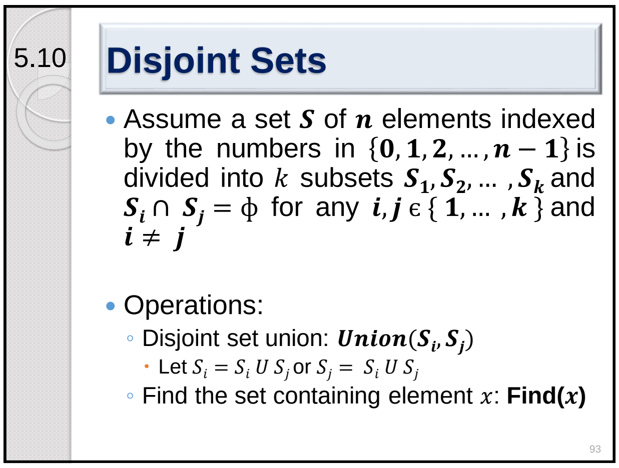


5.10

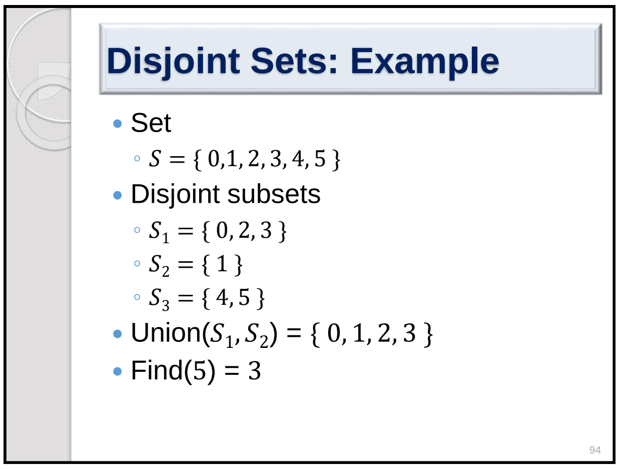
Representation of Disjoint Sets



5.10

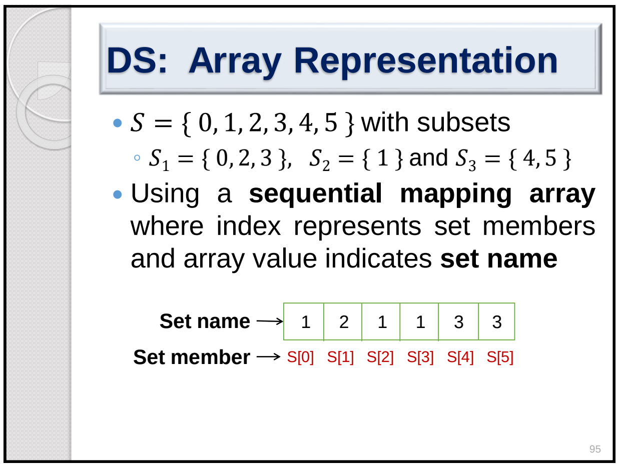
Disjoint Sets

- Assume a set $\boldsymbol{S}$ of $\boldsymbol{n}$ elements indexed by the numbers in $\{\mathbf{0},\mathbf{1},\mathbf{2},\dots,\boldsymbol{n}-\mathbf{1}\}$ is divided into k subsets $\boldsymbol{S_1},\boldsymbol{S_2},\dots,\boldsymbol{S_k}$ and $\boldsymbol{S_i}\cap\boldsymbol{S_j}=\phi$ for any $\boldsymbol{i},\boldsymbol{j}\in\{\mathbf{1},\dots,\boldsymbol{k}\}$ and $\boldsymbol{i}\neq\boldsymbol{j}$
- Operations:
  - Disjoint set union: $\boldsymbol{Union}(\boldsymbol{S_i},\boldsymbol{S_j})$
    - Let $S_i = S_i\ U\ S_j$ or $S_j = S_i\ U\ S_j$
  - Find the set containing element x: **Find($\boldsymbol{x}$)**



Disjoint Sets: Example

- Set
  - $S = \{0,1,2,3,4,5\}$
- Disjoint subsets
  - $S_1 = \{0,2,3\}$
  - $S_2 = \{1\}$
  - $S_3 = \{4,5\}$
- Union(S_1, S_2) = $\{0,1,2,3\}$
- Find(5) = 3

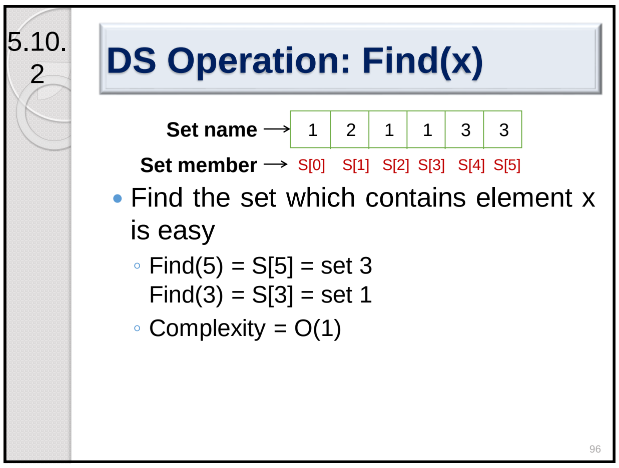


DS: Array Representation

- $S = \{ 0, 1, 2, 3, 4, 5 \}$ with subsets
  - $S_1 = \{ 0, 2, 3 \}$, $S_2 = \{ 1 \}$ and $S_3 = \{ 4, 5 \}$
- Using a **sequential mapping array** where index represents set members and array value indicates **set name**

Set name ⟶	1	2	1	1	3	3
Set member ⟶	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]

95



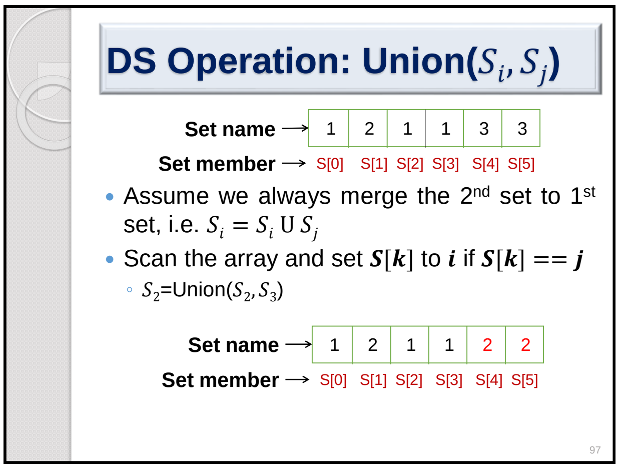
5.10.2

DS Operation: Find(x)

Set name ⟶	1	2	1	1	3	3
Set member ⟶	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]

- Find the set which contains element x is easy
  - Find(5) = S[5] = set 3
    Find(3) = S[3] = set 1
  - Complexity = O(1)

96



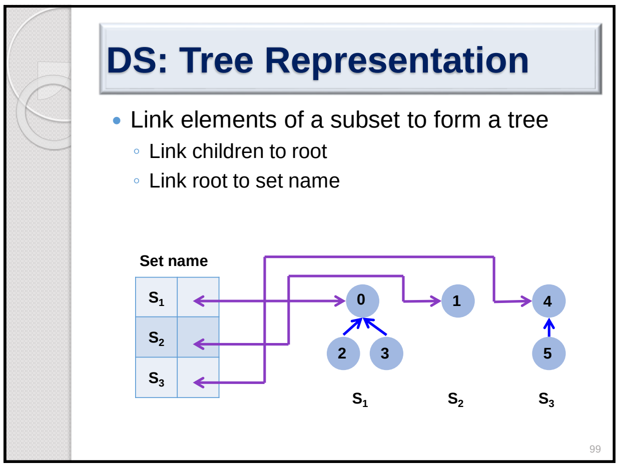
DS Operation: Union(S_i, S_j)

Set name ⟶	1	2	1	1	3	3
Set member ⟶	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]

- Assume we always merge the 2nd set to 1st set, i.e. $S_i = S_i \cup S_j$
- Scan the array and set $\boldsymbol{S[k]}$ to $\boldsymbol{i}$ if $\boldsymbol{S[k] == j}$
  - S_2=Union(S_2, S_3)

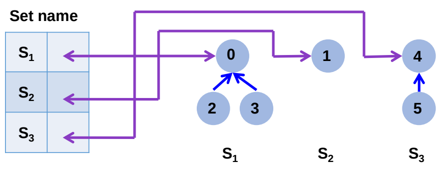
Set name ⟶	1	2	1	1	2	2
Set member ⟶	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]

97

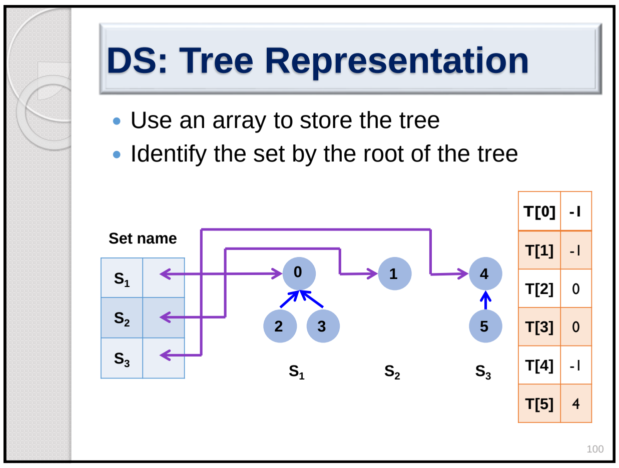


DS: Tree Representation

- Link elements of a subset to form a tree
  - Link children to root
  - Link root to set name

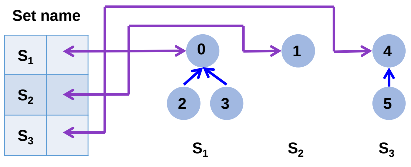


99



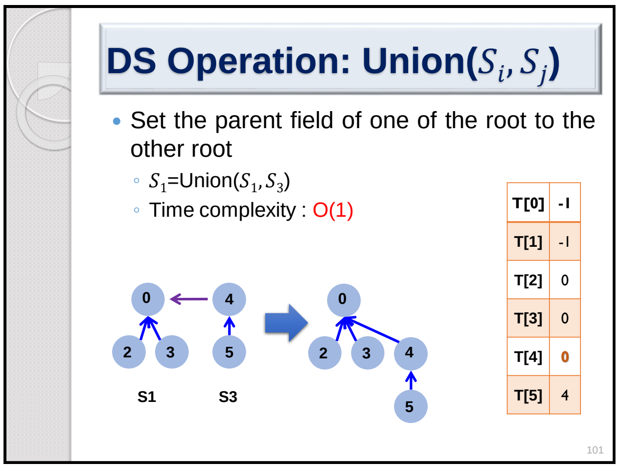
DS: Tree Representation

- Use an array to store the tree
- Identify the set by the root of the tree



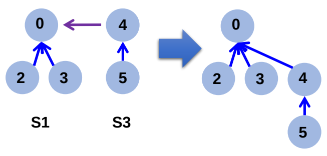
T[0]	-1
T[1]	-1
T[2]	0
T[3]	0
T[4]	-1
T[5]	4

100



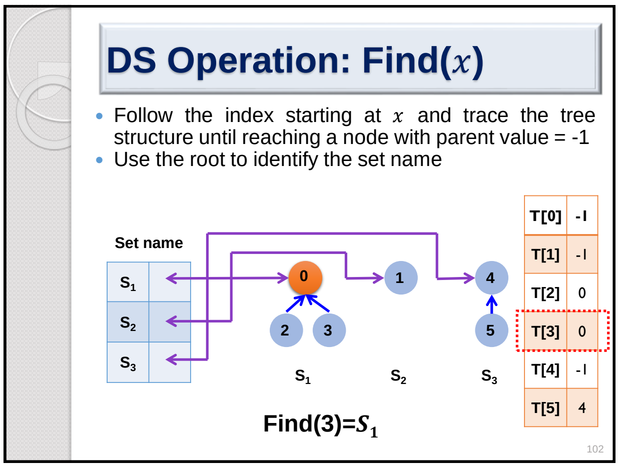
DS Operation: Union(S_i, S_j)

- Set the parent field of one of the root to the other root
  - S_1=Union(S_1, S_3)
  - Time complexity : O(1)



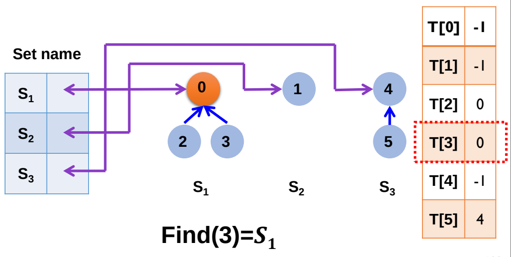
T[0]	-1
T[1]	-1
T[2]	0
T[3]	0
T[4]	0
T[5]	4

101



DS Operation: Find(x)

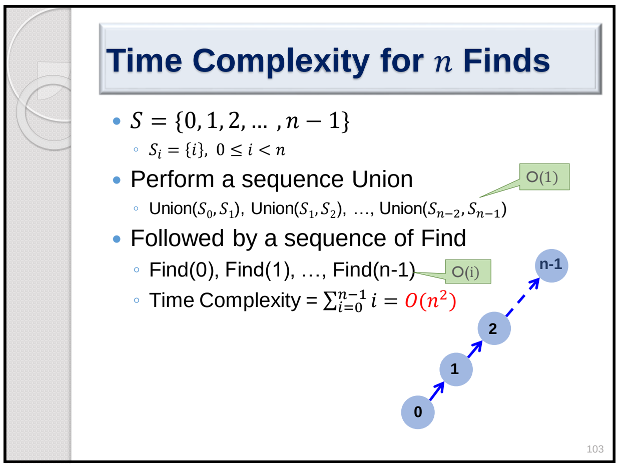
- Follow the index starting at x and trace the tree structure until reaching a node with parent value = -1
- Use the root to identify the set name



T[0]	-1
T[1]	-1
T[2]	0
T[3]	0
T[4]	-1
T[5]	4

Find(3)=S_1

102

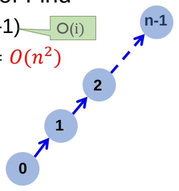


Time Complexity for n Finds

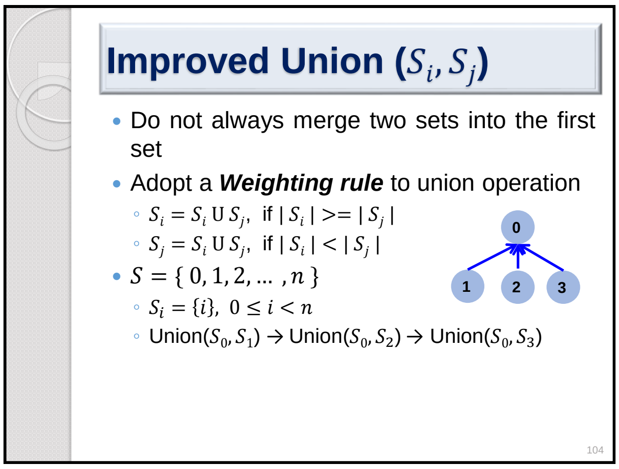
- $S = \{0, 1, 2, \dots, n-1\}$
  - $S_i = \{i\},\ 0 \le i < n$
- Perform a sequence Union
  - Union(S_0, S_1), Union(S_1, S_2), …, Union(S_{n-2}, S_{n-1})
- Followed by a sequence of Find
  - Find(0), Find(1), …, Find(n-1)
  - Time Complexity = $\sum_{i=0}^{n-1} i = O(n^2)$

O(1)

O(i)

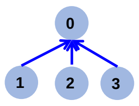


103

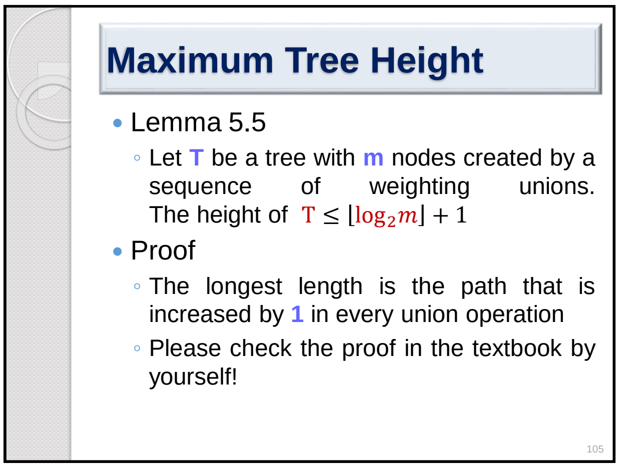


Improved Union (S_i, S_j)

- Do not always merge two sets into the first set
- Adopt a ***Weighting rule*** to union operation
  - $S_i = S_i \cup S_j$, if $|S_i| >= |S_j|$
  - $S_j = S_i \cup S_j$, if $|S_i| < |S_j|$
- $S = \{0, 1, 2, \dots, n\}$
  - $S_i = \{i\},\ 0 \le i < n$
  - Union(S_0, S_1) → Union(S_0, S_2) → Union(S_0, S_3)



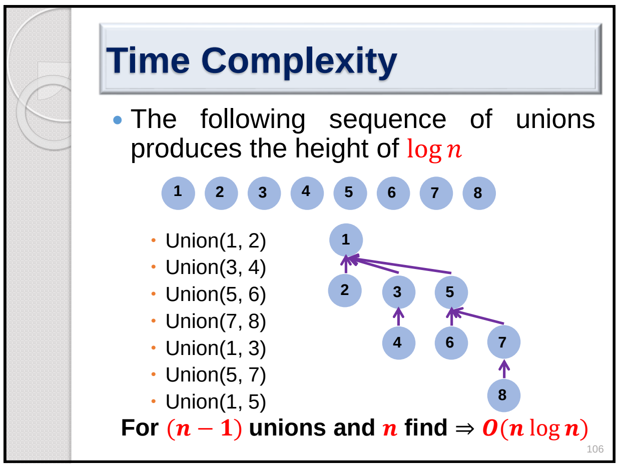
104



Maximum Tree Height

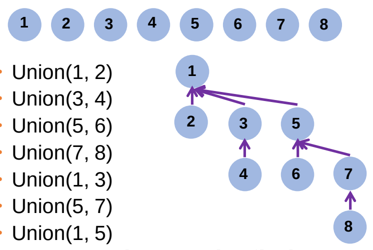
- Lemma 5.5
  - Let T be a tree with m nodes created by a sequence of weighting unions. The height of $T \le \lfloor \log_2 m \rfloor + 1$
- Proof
  - The longest length is the path that is increased by 1 in every union operation
  - Please check the proof in the textbook by yourself!

105



Time Complexity

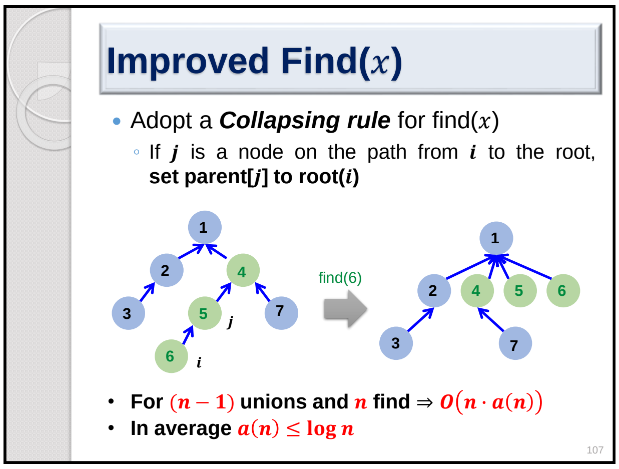
- The following sequence of unions produces the height of $\log n$



- Union(1, 2)
- Union(3, 4)
- Union(5, 6)
- Union(7, 8)
- Union(1, 3)
- Union(5, 7)
- Union(1, 5)

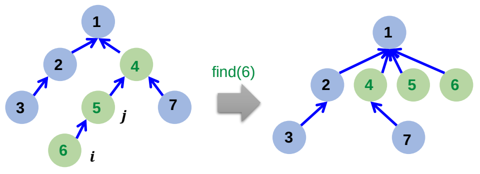
For $(n-1)$ unions and n find $\Rightarrow O(n \log n)$

106



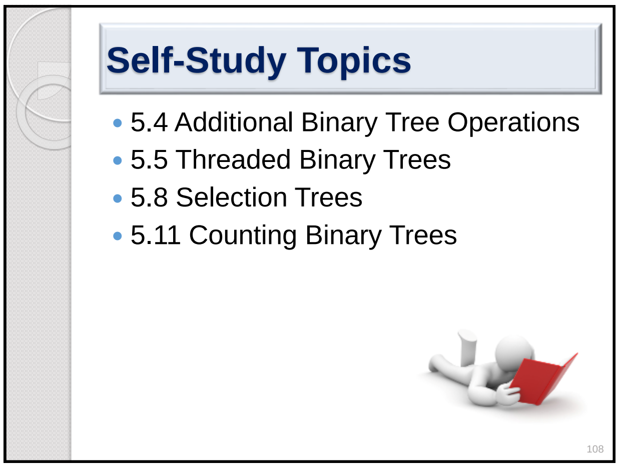
Improved Find(x)

- Adopt a ***Collapsing rule*** for find(x)
  - If j is a node on the path from i to the root, **set parent[j] to root(i)**



- **For $(n-1)$ unions and n find $\Rightarrow O(n \cdot a(n))$**
- **In average $a(n) \le \log n$**

107



Self-Study Topics

- 5.4 Additional Binary Tree Operations
- 5.5 Threaded Binary Trees
- 5.8 Selection Trees
- 5.11 Counting Binary Trees

108