

PARALLEL AND SEQUENTIAL ALGORITHMS AND DATA STRUCTURES

LECTURE 12

Binary Search Trees



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

BST is important?

- Searching is one of the most important operations in computer science
- What data structure can we use for searching?
 - Sequence?
 - Search tree?
 - Binary search tree?
 - Balanced binary search tree?

Difference?
Which one is better?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- Implementation via Balancing
- A Parametric Implementation
- Cost Specification
- Treaps
- Augmenting Trees
- Exercises



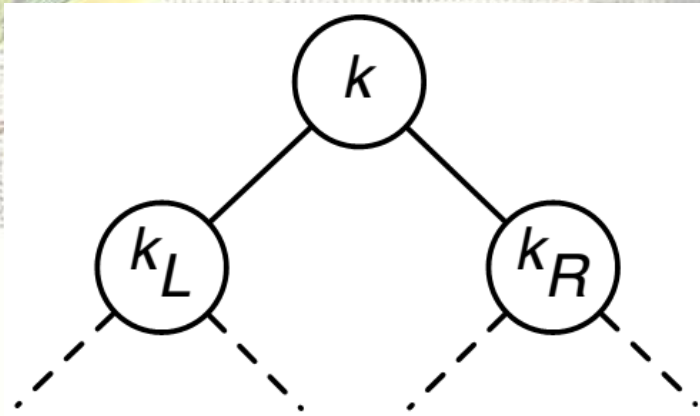
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Binary Trees

- **A rooted tree** is a tree with a distinguished root node that can be used to access all other nodes
- **A full binary tree** is an ordered rooted tree in which every node has **exactly** two children
 - Left child / Left subtree
 - Right child / Right subtree



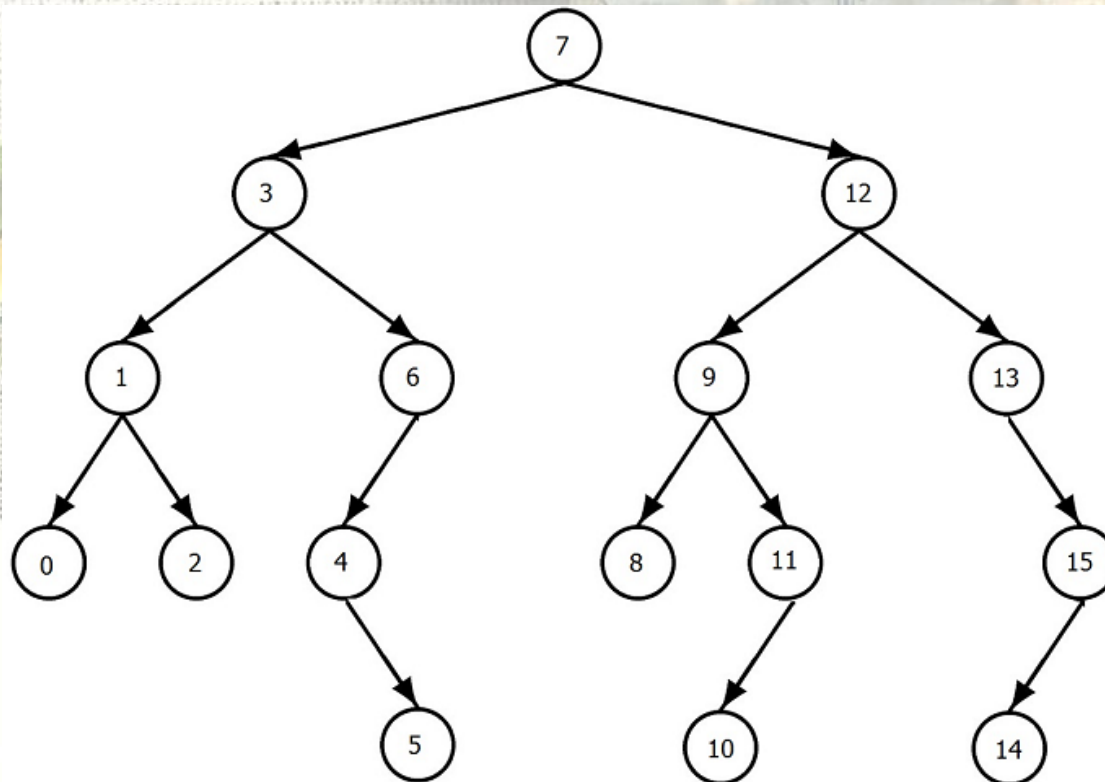
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Binary Search Trees

- Binary trees with the “search” property
- For each node v with key k
 - The key of the left child $k_L < k$
 - The key of the right child $k_R > k$



明

Binary Search Trees

- A binary search tree (BST) over a totally ordered set S is a full binary tree that satisfies the following conditions.
 - 1. There is a one-to-one mapping $k(v)$ from internal tree nodes to elements in S
 - 2. for every u in the left subtree of v , $k(u) < k(v)$
 - 3. for every u in the right subtree of v , $k(u) > k(v)$

Can you write a binary search tree over the set $\{1, 4, 7, 9, 13, 17\}$?

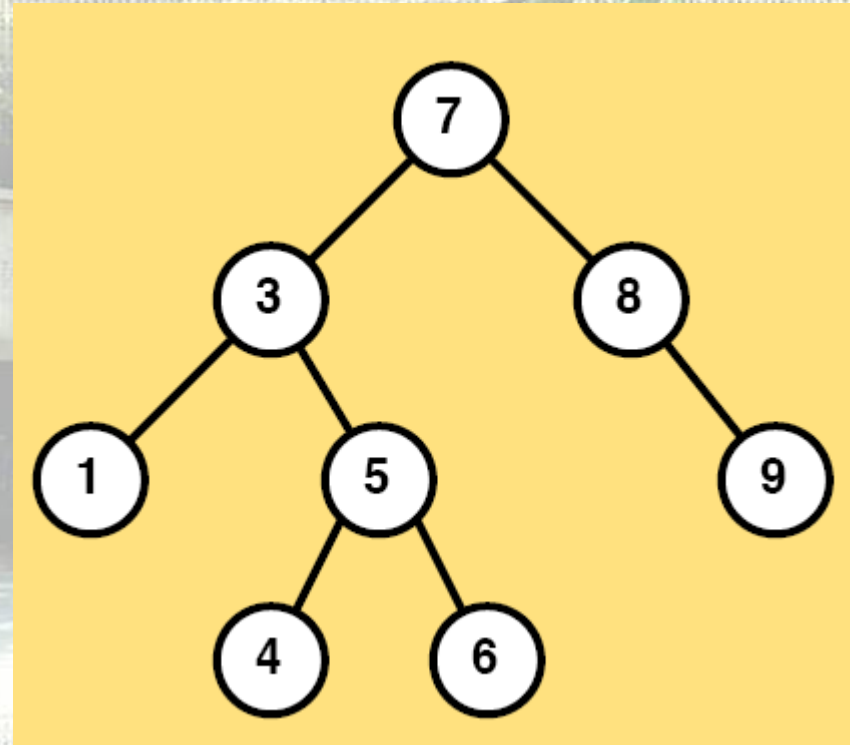
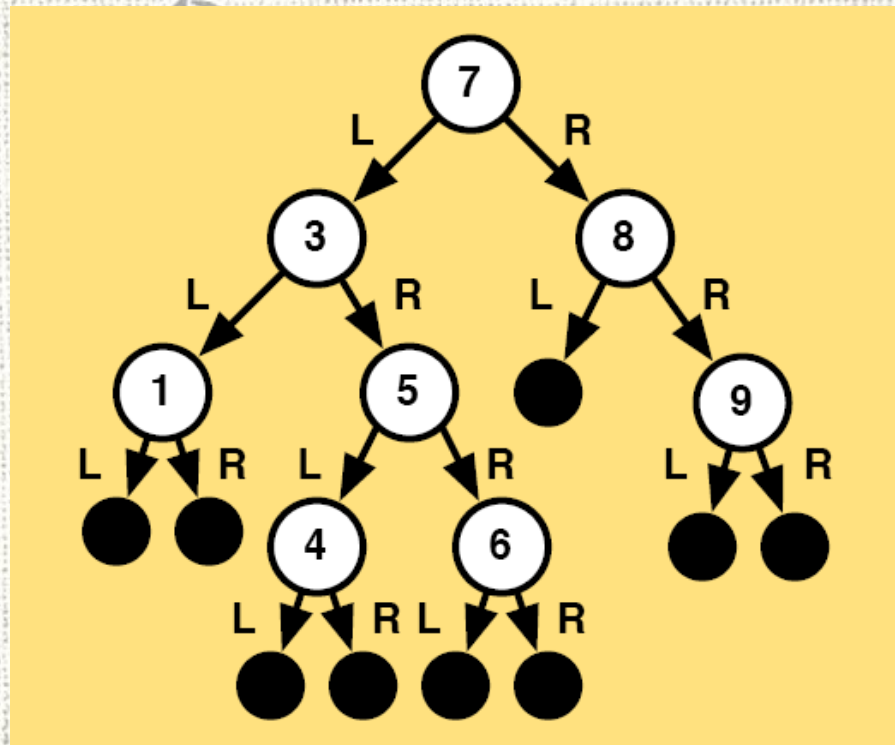


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Binary Search Trees



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SYNOPSIS

- Preliminaries
- The **BST Abstract Data Type**
- Implementation via Balancing
- A Parametric Implementation
- Cost Specification
- Treaps
- Augmenting Trees
- Exercises



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

EMPTY, SINGLETON

- **K & T**

➤ For a universe of totally ordered keys **K**, the BST ADT consists of a type **T** representing a power set of keys

- **[T]**

➤ for a tree **T**, **[T]** denotes the set of keys in the tree

<code>empty</code>	$:$	\mathbb{T}
<code>empty</code>	$=$	$T \text{ where } [T] = \emptyset$
<code>singleton</code>	$:$	$\mathbb{K} \rightarrow \mathbb{T}$
<code>singleton(k)</code>	$=$	$T \text{ where } [T] = \{k\}.$



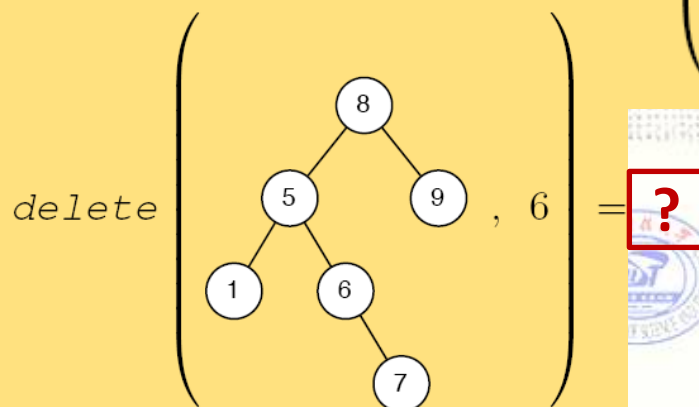
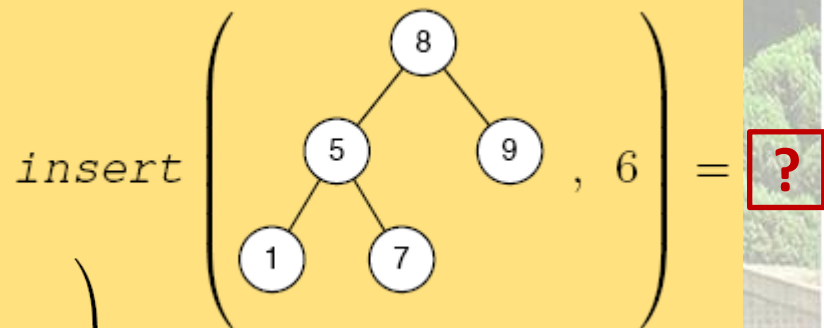
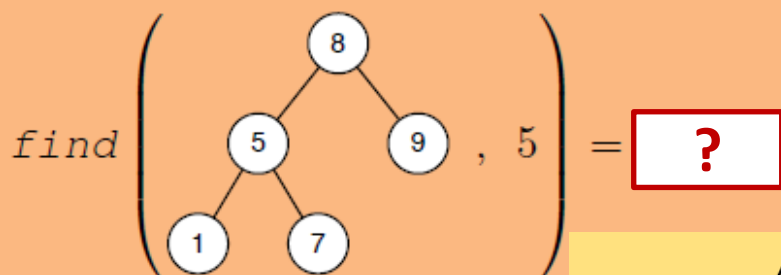
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

FIND, INSERT, DELETE

$find$: $(T \times \mathbb{K}) \rightarrow \mathbb{B}$
 $find(T, k)$ = true if and only if $k \in [T]$
 $insert$: $(T \times \mathbb{K}) \rightarrow T$
 $insert(T, k)$ = T' where $[T'] = [T] \cup \{k\}$
 $delete$: $(T \times \mathbb{K}) \rightarrow T$
 $delete(T, k)$ = T' where $[T'] = [T] \setminus \{k\}$.

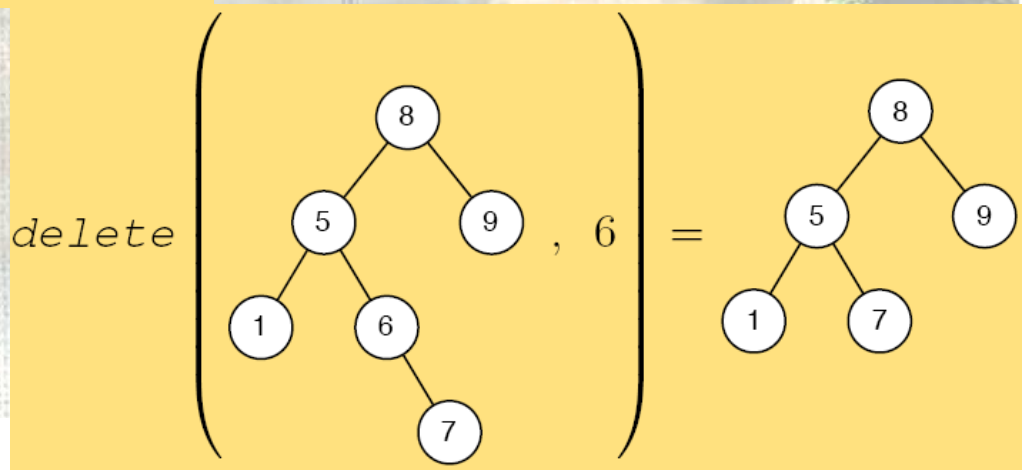
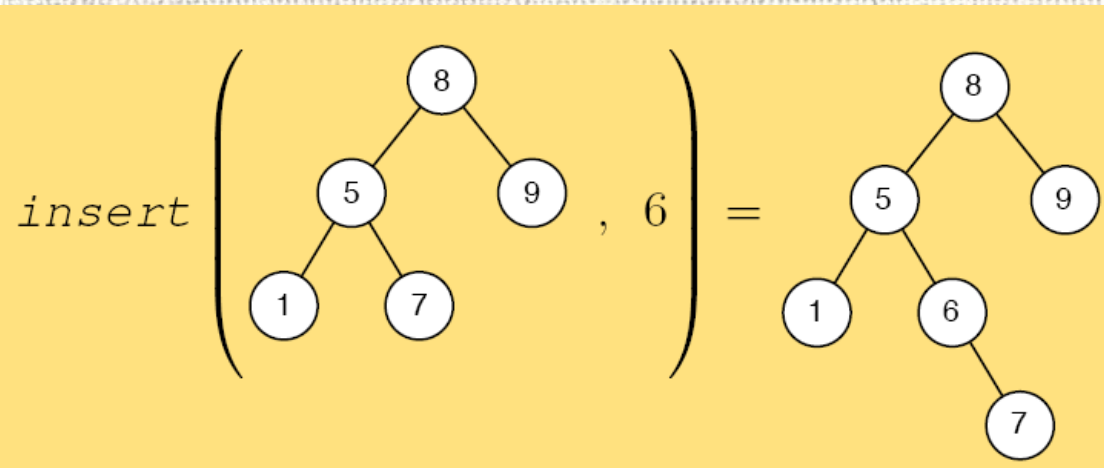


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

FIND, INSERT, DELETE



12-1 Insert

Design an algorithm for inserting a given key into a BST.

12-2 Delete

Design an algorithm for deleting a given key from a tree.



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

UNION, INTERSECTION, DIFF

明德
求道

$\text{union} : (\mathbb{T} \times \mathbb{T}) \rightarrow \mathbb{T}$
 $\text{union}(T_1, T_2) = T \text{ where } \llbracket T \rrbracket = \llbracket T_1 \rrbracket \cup \llbracket T_2 \rrbracket$

$\text{intersection} : (\mathbb{T} \times \mathbb{K}) \rightarrow \mathbb{T}$
 $\text{intersection}(T_1, T_2) = T \text{ where } \llbracket T \rrbracket = \llbracket T_1 \rrbracket \cap \llbracket T_2 \rrbracket$

$\text{difference} : (\mathbb{T} \times \mathbb{K}) \rightarrow \mathbb{T}$
 $\text{difference}(T_1, T_2) = T \text{ where } \llbracket T \rrbracket = \llbracket T_1 \rrbracket \setminus \llbracket T_2 \rrbracket$



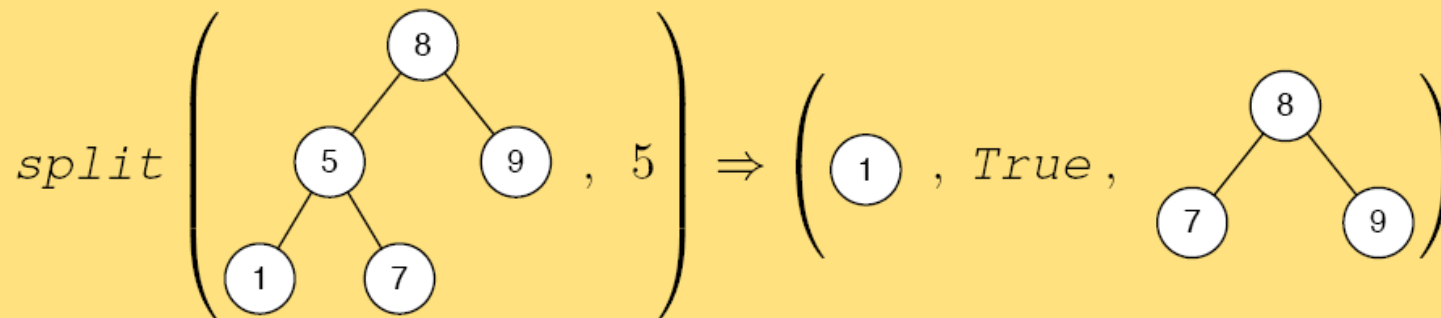
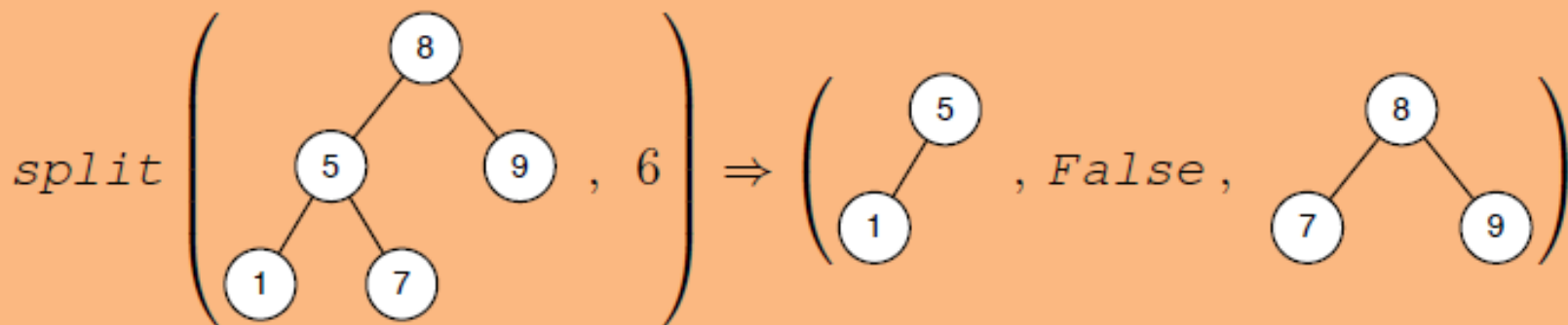
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SPLIT

split : $T \rightarrow K \rightarrow (T \times B \times T)$



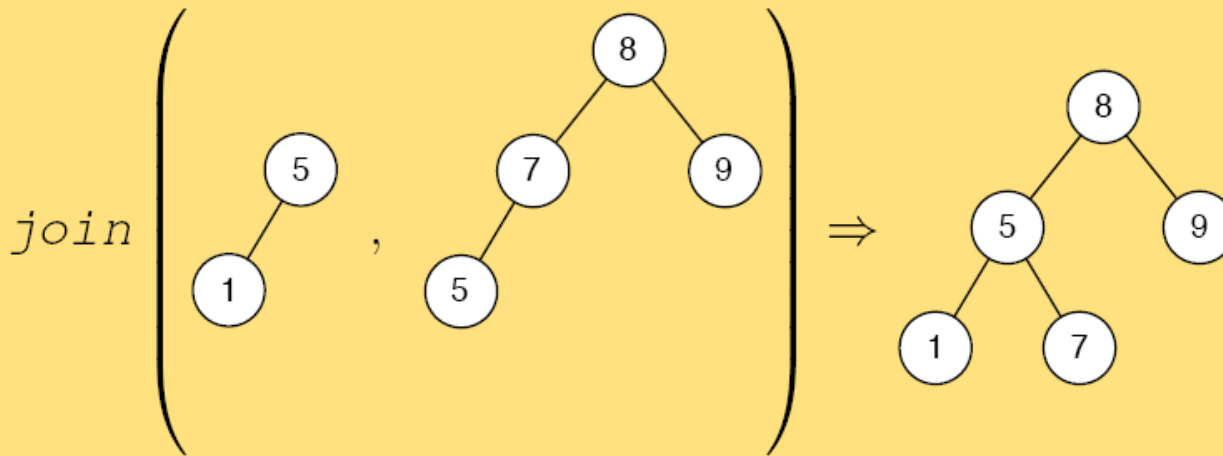
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

JOIN

- The function $\text{join}(T_1, T_2)$ takes two trees T_1 and T_2 such that **all the keys in T_1 are less than the keys in T_2** .

$$\begin{aligned} \text{join} &: (\mathbb{T} \times \mathbb{T}) \rightarrow \mathbb{T} \\ \text{join}(T_1, T_2) &= T \text{ where } [T] = [T_1] \cup [T_2] \end{aligned}$$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SPLIT, JOIN

- The exact structure of the trees returned by split can **differ from one implementation to another**
 - the specification only requires that the resulting trees to be valid BST's and that they contain the keys less than k and greater than k , leaving their structure otherwise unspecified
- The exact structure of the tree returned by join can **differ from one implementation to another**
 - the specification only requires that the resulting tree is a valid BST and that it contains all the keys in the trees joined.



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- **Implementation via Balancing**
- A Parametric Implementation
- Cost Specification
- Treaps
- Augmenting Trees
- Exercises



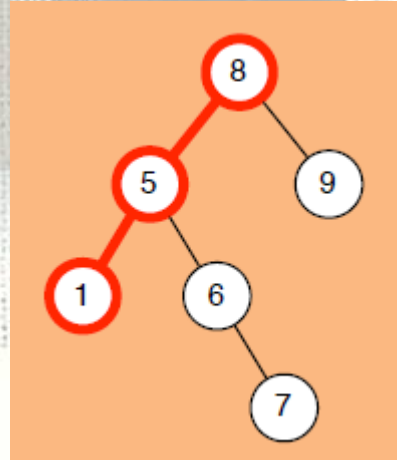
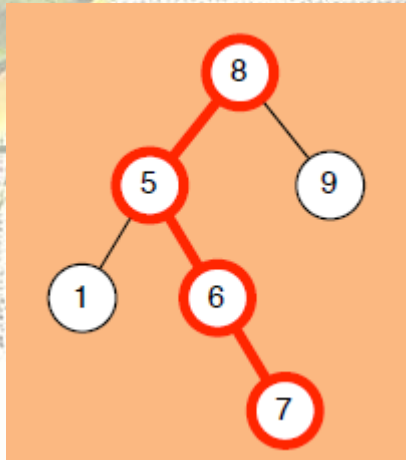
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Search tree

- The main idea behind BST's is to organize the keys such that
 - 1. a specific key can be searched by following a branch in the tree by doing key comparisons along the way
 - Search 7 & 4



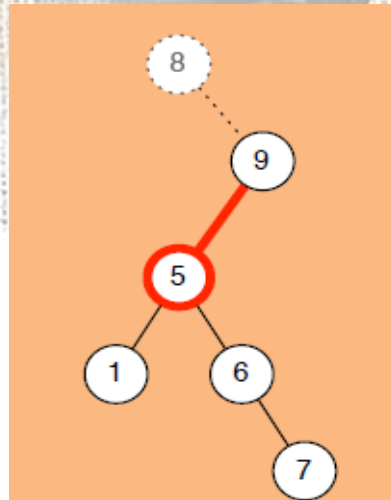
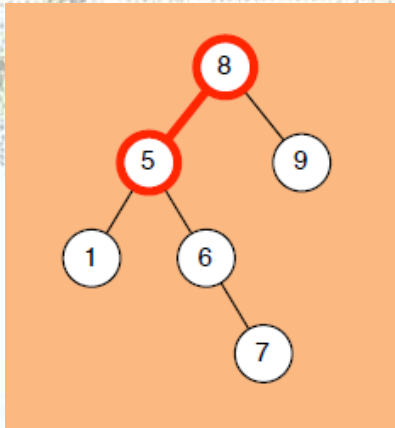
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Search tree

- The main idea behind BST's is to organize the keys such that
 - 2. a range of keys in a subtree can be operated on (e.g., moved) by performing constant work
 - ✓ each subtree in a binary tree contains the keys that all the keys within a specific range, e.g., all keys less than 8
 - ✓ Once we find a range of keys, we can operate on them as a group by handling the root



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Balanced tree

- The find & search operations in the BST ADT depend on the **paths** that we have to walk in the tree
- A binary tree is defined to be perfectly **balanced** if it **has the minimum possible height**
 - Both children are about the same height
 - Both subtrees are about the same size
- For a binary search tree over a set S , a perfectly balanced tree has height exactly $\lceil \log_2(|S| + 1) \rceil$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY



BALANCED BST

计算机科学中的树

二叉树	<ul style="list-style-type: none">▪ 二叉树▪ T树	<ul style="list-style-type: none">▪ 二叉查找树	<ul style="list-style-type: none">▪ 笛卡尔树	<ul style="list-style-type: none">▪ Top tree
自平衡二叉查找树	<ul style="list-style-type: none">▪ AA树▪ 树堆	<ul style="list-style-type: none">▪ AVL树▪ 节点大小平衡树	<ul style="list-style-type: none">▪ 红黑树	<ul style="list-style-type: none">▪ 伸展树
B树	<ul style="list-style-type: none">▪ B树▪ UB树▪ Dancing tree	<ul style="list-style-type: none">▪ B+树▪ 2-3树▪ H树	<ul style="list-style-type: none">▪ B*树▪ 2-3-4树	<ul style="list-style-type: none">▪ Bx树▪ (a,b)-树
Trie	<ul style="list-style-type: none">▪ 前缀树	<ul style="list-style-type: none">▪ 后缀树	<ul style="list-style-type: none">▪ 基数树	
空间划分树	<ul style="list-style-type: none">▪ 四叉树▪ R树▪ M树	<ul style="list-style-type: none">▪ 八叉树▪ R*树▪ 线段树	<ul style="list-style-type: none">▪ k-d树▪ R+树▪ 希尔伯特R树	<ul style="list-style-type: none">▪ vp-树▪ X树▪ 优先R树
非二叉树	<ul style="list-style-type: none">▪ Exponential tree▪ Range tree	<ul style="list-style-type: none">▪ Fusion tree▪ SPQR tree	<ul style="list-style-type: none">▪ 区间树▪ Van Emde Boas tree	<ul style="list-style-type: none">▪ PQ tree
其他类型	<ul style="list-style-type: none">▪ 堆▪ Cover tree▪ Link-cut tree	<ul style="list-style-type: none">▪ 散列树▪ BK-tree▪ 树状数组	<ul style="list-style-type: none">▪ Finger tree▪ Doubly-chained tree	<ul style="list-style-type: none">▪ Metric tree▪ iDistance

BALANCED BST

- There are many balanced BST data structures
 - **AVL trees** are the earliest **near-balance BST** data structure (1962). It maintains the invariant that the two children of each node **differ in height by at most one**
 - **Red-Black trees** maintain the invariant that all leaves have a depth that is within a factor of 2 of each other.
 - ✓ The depth invariant is ensured by a scheme of coloring the nodes red and black
 - **Weight balanced** (BB[α]) trees maintain the invariant that the left and right subtrees of a node of size n each have size at least αn for $0 < \alpha \leq 1/2$.
 - ✓ The BB stands for bounded balance, and adjusting α gives a tradeoff between search and update costs



BALANCED BST

- There are many balanced BST data structures
 - **Treaps** associate a **random priority** with every key and maintain the invariant that the keys are stored in heap order with respect to their priorities (treaps is short for tree-heaps)
 - ✓ Treaps guarantee near balance with high-probability
 - **Splay trees** are an amortized data structure that does not guarantee near balance, but instead guarantees that for any sequence of **m insert**, find and delete operations each does $O(\log n)$ amortized work



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- Implementation via Balancing
- **A Parametric Implementation**
- Cost Specification
- Treaps
- Augmenting Trees
- Exercises



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

A Parametric Implementation

- Implementing the BST ADT with split and join

`type T = Leaf | Node of (T × K × T)`

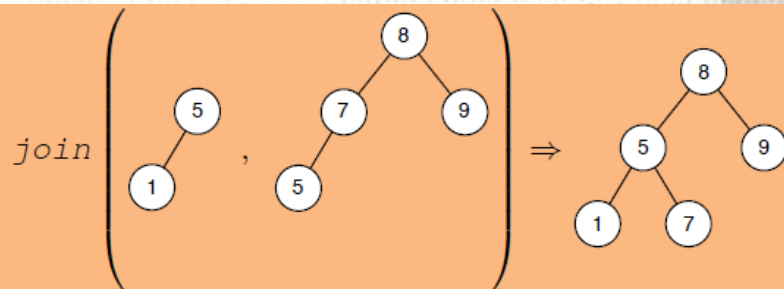
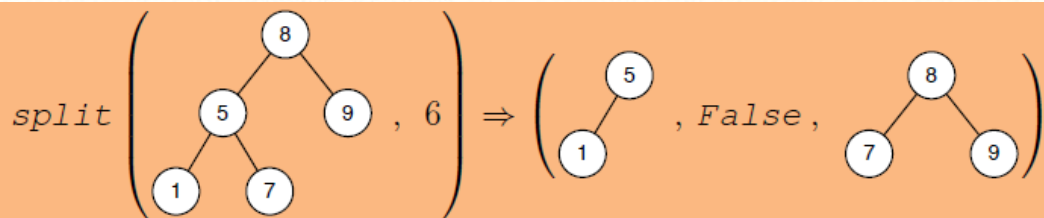
`split (T,k) = ... (* as given *)`

`join (T1,T2) = ... (* as given *)`

`joinM (T1,k,T2) = join (T1, join (singleton k, T2))`

`empty = Leaf`

`singleton (k) = Node(Leaf,k,Leaf)`



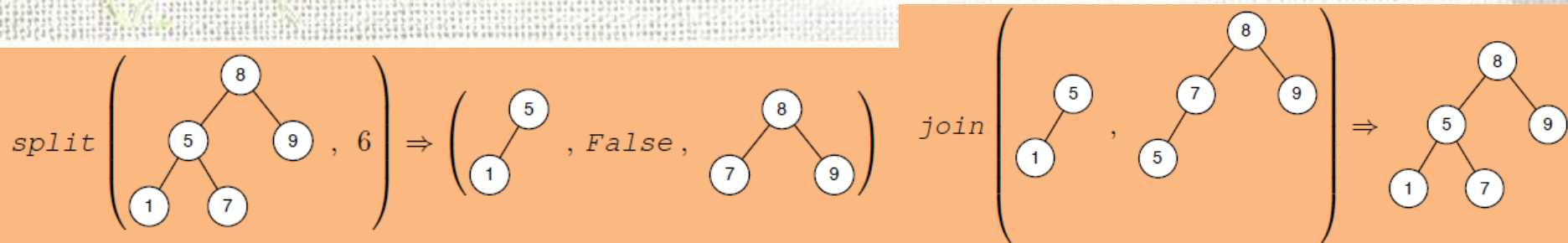
A Parametric Implementation

- Implementing the BST ADT with split and join
 - find, insert, delete?

$find(T, k) = \text{let } (_, v, _) = split(T, k) \text{ in } v \text{ end}$

$insert(T, k) = \text{let } (L, _, R) = split(T, k) \text{ in } joinM(L, k, R) \text{ end}$

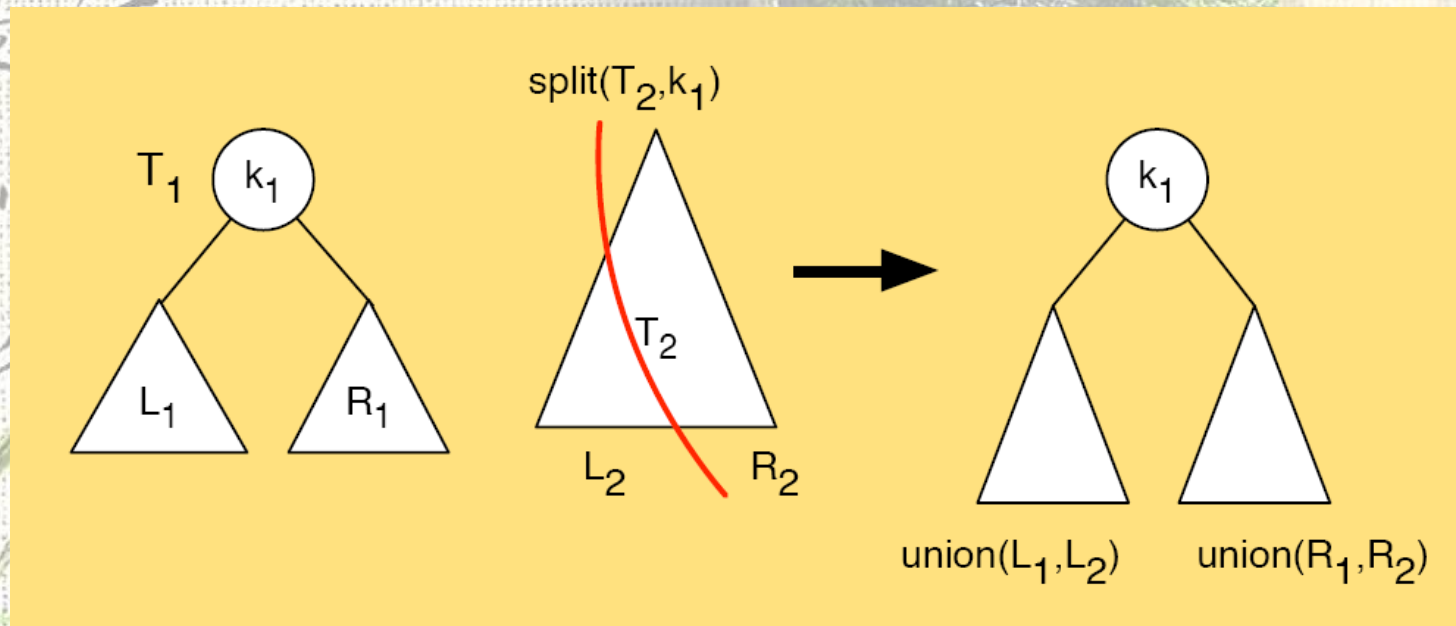
$delete(T, k) = \text{let } (L, _, R) = split(T, k) \text{ in } join(L, R) \text{ end}$



A Parametric Implementation

- Implementing the BST ADT with split and join

➤ union?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

A Parametric Implementation

- Implementing the BST ADT with split and join
 - union?

```
union t1 t2 =  
  case (t1, t2)  
  | (Leaf, _) => t2  
  | (_, Leaf) => t1  
  | (Node (l1, k1, r1), _) =>  
    let (l2, -, r2) = split t2 k1  
      (l, r) = (union l1 l2) || (union r1 r2)  
    in joinM l k1 r end
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

A Parametric Implementation

- Implementing the BST ADT with split and join

➤ Intersect?

```
intersect  $t_1$   $t_2$  =  
  case ( $t_1, t_2$ )  
  | (Leaf, _) => Leaf  
  | (_, Leaf) => Leaf  
  | (Node ( $l_1, k_1, r_1$ ), _) =>  
    let ( $l_2, b, r_2$ ) = split  $t_2$   $k_1$   
      ( $l, r$ ) = (intersect  $l_1$   $l_2$ ) || (intersect  $r_1$   $r_2$ )  
    in if  $b$  then joinM  $l$   $k_1$   $r$  else join  $l$   $r$  end
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

A Parametric Implementation

- Implementing the BST ADT with split and join

➤ Diff?

```
difference  $t_1$   $t_2$  =  
  case ( $t_1$ ,  $t_2$ )  
  | (Leaf, _) => Leaf  
  | (_, Leaf) =>  $t_1$   
  | (Node ( $l_1, k_1, r_1$ ), _) =>  
    let ( $l_2, b, r_2$ ) = split  $t_2$   $k_1$   
      ( $l, r$ ) = (difference  $l_1$   $l_2$ ) || (difference  $r_1$   $r_2$ )  
    in if  $b$  then join  $l$   $r$  else joinM  $L$   $k_1$   $r$  end
```



華中科技大學

Exercise 12.13. Prove correct the functions `intersection`, `difference`, and `union`.

明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- Implementation via Balancing
- A Parametric Implementation
- **Cost Specification**
- Treaps
- Augmenting Trees
- Exercises



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

NOTICES

- These implementations all use balancing techniques to ensure that the **depth of the BST** remains **$O(\log n)$**
- Our cost-specifications can be viewed as **worst-case bounds**
- variables n and m are defined as **$n = \max(|T_1|, |T_2|)$** and **$m = \min(|T_1|, |T_2|)$**

	Balanced BST	
	Work	Span
<i>empty</i>	$O(1)$	$O(1)$
<i>singletonv</i>	$O(1)$	$O(1)$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

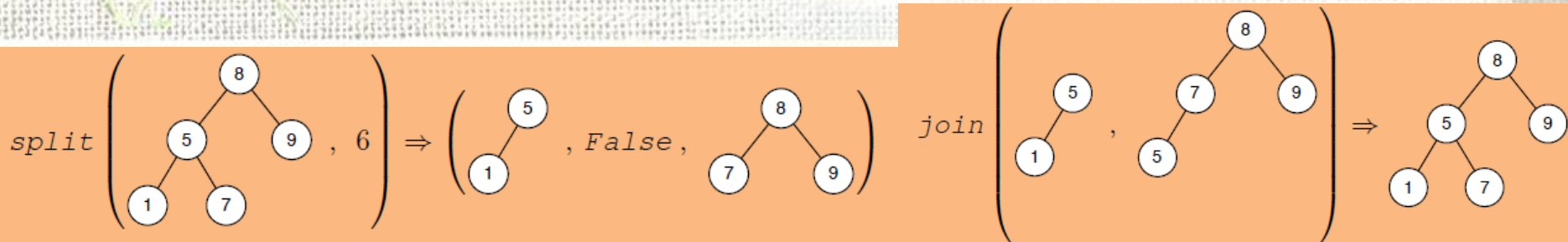
COST SPECIFICATION

- **Split(T, k) & join(T_1, T_2)**

➤ **W=? S=?**

	Work	Span
split $t k$	$O(\lg t)$	$O(\lg t)$
join $t_1 t_2$	$O(\lg (t_1 + t_2))$	$O(\lg (t_1 + t_2))$

how?



明

COST SPECIFICATION

- $\text{find}(T, k)$, $\text{insert}(T, k)$, $\text{delete}(T, k)$

➤ $W=?$, $S=?$

$\text{find}(T, k) = \text{let } (_, v, _) = \text{split}(T, k) \text{ in } v \text{ end}$

$\text{insert}(T, k) = \text{let } (L, _, R) = \text{split}(T, k) \text{ in } \text{joinM}(L, k, R) \text{ end}$

$\text{delete}(T, k) = \text{let } (L, _, R) = \text{split}(T, k) \text{ in } \text{join}(L, R) \text{ end}$

Why?

	Work	Span
$\text{find } t \ k$	$O(\lg t)$	$O(\lg t)$
$\text{insert } t \ k$	$O(\lg t)$	$O(\lg t)$
$\text{delete } t \ k$	$O(\lg t)$	$O(\lg t)$

科技大學

UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

COST SPECIFICATION

- $\text{union}(T_1, T_2)$, $\text{intersect}(T_1, T_2)$, $\text{diff}(T_1, T_2)$

➤ $W=?$, $S=?$

how?

	Work	Span
$\text{intersect } t_1 t_2$	$O\left(m \cdot \lg \frac{n+m}{m}\right)$	$O(\lg n)$
$\text{difference } t_1 t_2$	$O\left(m \cdot \lg \frac{n+m}{m}\right)$	$O(\lg n)$
$\text{union } t_1 t_2$	$O\left(m \cdot \lg \frac{n+m}{m}\right)$	$O(\lg n)$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Concrete Implementations: Union

- Recall union implement

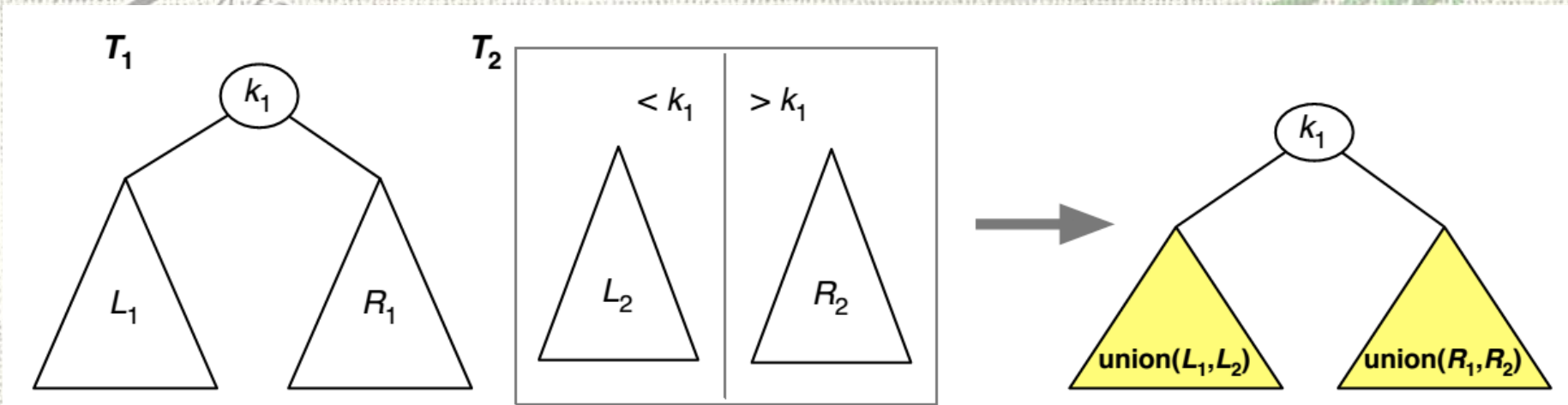
```
union t1 t2 =  
  case (t1, t2)  
  | (Leaf, _) => t2  
  | (_, Leaf) => t1  
  | (Node (l1, k1, r1), _) =>  
    let (l2, -, r2) = split t2 k1  
      (l, r) = (union l1 l2) || (union r1 r2)  
    in joinM l k1 r end
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Concrete Implementations: Union



- For T_1 with key k_1 and children L_1 and R_1 at the root, use k_1 to split T_2 into L_2 and R_2
- Recursively find $L_u = \text{union}(L_1, L_2)$ and $R_u = \text{union}(R_1, R_2)$
- Now $\text{join}(L_u, k_1, R_u)$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Analysis of Union

```
union  $t_1$   $t_2$  =  
  case ( $t_1, t_2$ )  
  | (Leaf, _) =>  $t_2$   
  | (_, Leaf) =>  $t_1$   
  | (Node ( $l_1, k_1, r_1$ ), _) =>  
    let ( $l_2, -, r_2$ ) = split  $t_2$   $k_1$   
      ( $l, r$ ) = (union  $l_1$   $l_2$ ) || (union  $r_1$   $r_2$ )  
    in joinM  $l$   $k_1$   $r$  end
```

- split costs $O(\lg |T_2|)$
- Two recursive calls to union
- join costs $O(\lg(|T_1| + |T_2|))$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of Union - Assumptions

- To simplify the analysis, we will make the following assumptions
 - T_1 is perfectly balanced
 - Each a key from T_1 splits T_2 , it splits exactly in half
 - without loss of generality let $|T_1| \leq |T_2|$
 - Then, $m = |T_1|$, $n = |T_2|$

$$\begin{aligned} W_{\text{union}}(m, n) &\leq 2W_{\text{union}}(m/2, n/2) + W_{\text{split}}(n) + W_{\text{join}}(n + m) + O(1) \\ &\leq 2W_{\text{union}}(m/2, n/2) + O(\lg n) . \end{aligned}$$

Why?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Analysis of Union

- When $|T_1| = 1$, case give us two empty subtrees L_1 and R_1
- $\text{union}(L_1, L_2)$ returns L_2 , $\text{union}(R_1, R_2)$ returns R_2 immediately!
- Joining these costs at most $O(\log(|T_1| + |T_2|)) = O(\log(1 + |T_2|))$

$$\begin{aligned} W_{\text{union}}(1, n) &\leq 2W_{\text{union}}(0, n/2) + W_{\text{split}}(n) + W_{\text{join}}(n) + O(1) \\ &\leq O(\lg n) . \end{aligned}$$

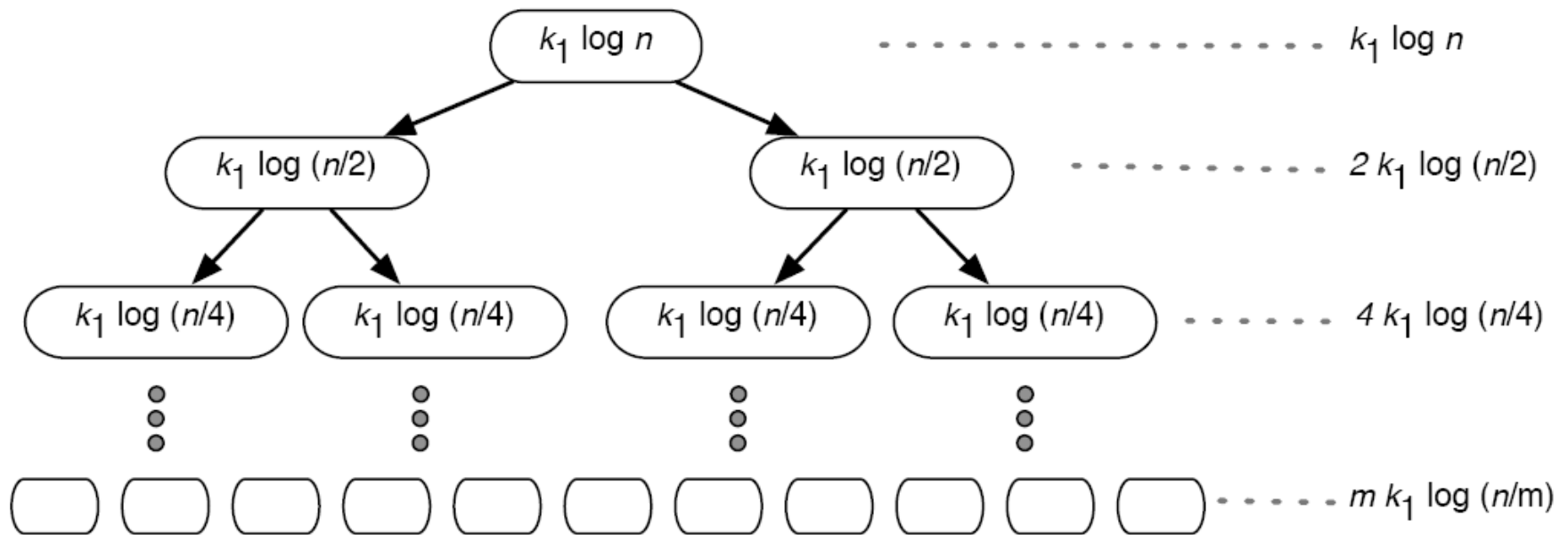


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

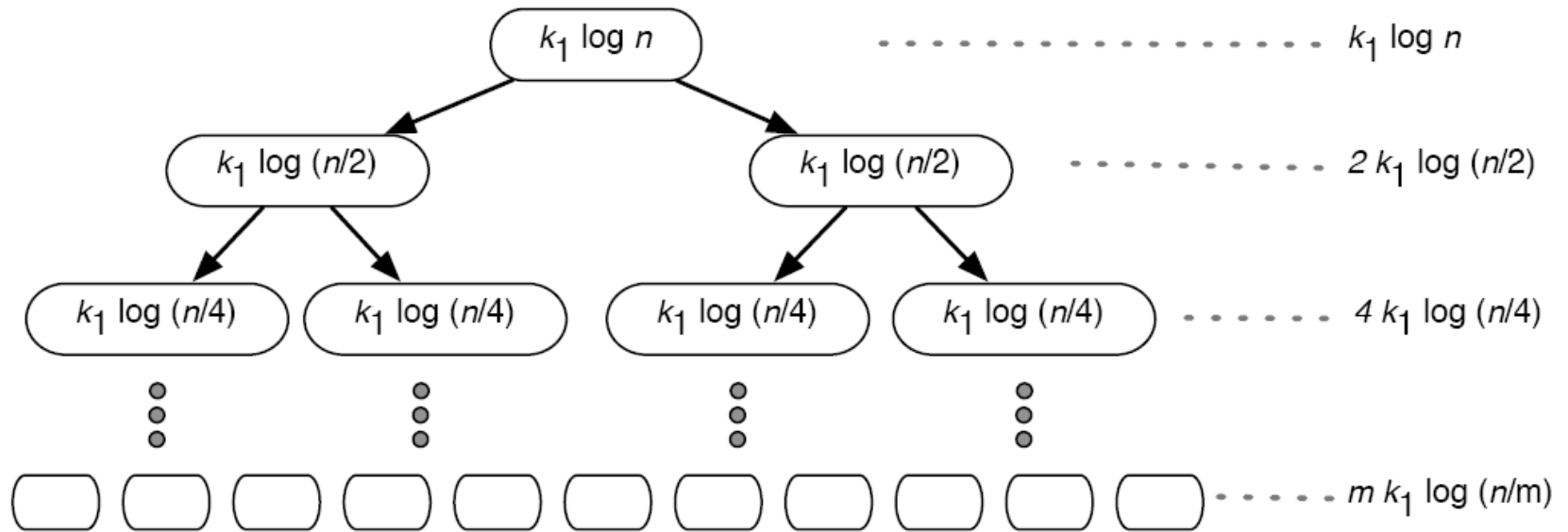
Analysis of Union

- If we draw the recursion tree that shows the work associated with splitting T_2 and joining the results, we obtain the following



$$W_{\text{union}}(1, n) \leq 2W_{\text{union}}(0, n/2) + W_{\text{split}}(n) + W_{\text{join}}(n) + O(1) \\ \leq O(\lg n) .$$

Analysis of Union



明

Analysis of Union

- How many leaves are there in this recursion tree?
 - T_2 has no impact
 - We get $m = |T_1|$ leaves — Why?
- How deep is the tree?
 - $1 + \log_2 m$ — Why?
- What is the size of T_2 at the leaves?
 - $n / 2^{\log_2 m} = n/m$ — Why?
- Total cost at the leaves = $O(m \log(n/m))$ — Why?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of Union

- We will now prove that the cost at the bottom level is indeed asymptotically the same as the total work.
 - It is possible to prove that the tree is leaves-dominated by computing the ratio of the work at adjacent levels,

$$\frac{2^{i-1} k_1 \lg n / 2^{i-1}}{2^i k_1 \lg n / 2^i} = \frac{1}{2} \frac{\lg n - i + 1}{\lg n - i}$$

✓ where $i \leq \lg m < \lg n$

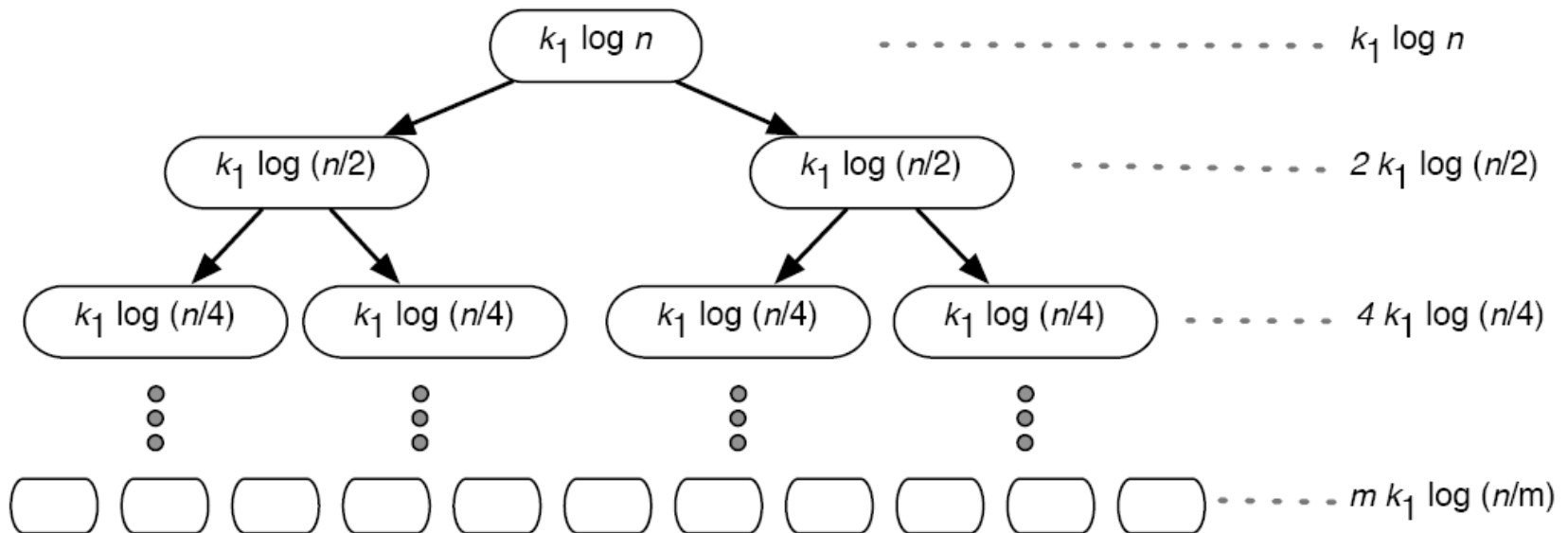
- This ratio is less than 1 for all levels except for the last level, where by taking $i = \lg n - 1$ we have

$$\frac{1}{2} \frac{\lg n - i + 1}{\lg n - i} \leq \frac{1}{2} \frac{1}{\lg n - \lg n + 1 + 1} \lg n - \lg n + 1 = \frac{1}{1}.$$



Analysis of Union

- Thus the total work is asymptotically dominated by the total work of the leaves, which is
 - $O(m \lg n/m)$

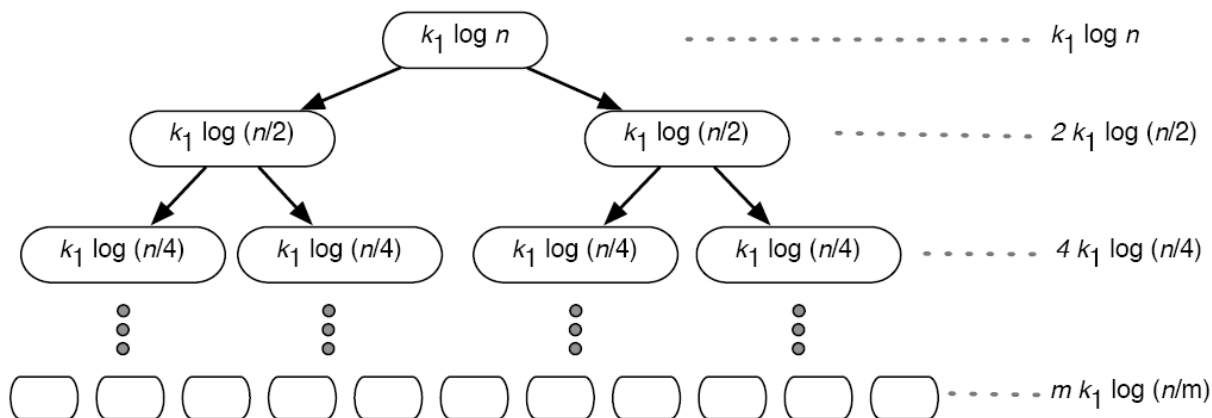


Analysis of Union

- **Direct derivation**

- We can establish the same fact more precisely.
- Let's start by writing the total cost across all levels, omitting for simplicity the constant factor k_1 , and assuming that $n = 2^a$ and $m = 2^b$

$$\sum_{i=0}^b 2^i \lg \frac{n}{2^i}.$$



Analysis of Union

- We can rewrite this sum as

$$\sum_{i=0}^b 2^i \lg \frac{n}{2^i} = \lg n \sum_{i=0}^b 2^i - \sum_{i=0}^b i 2^i = a \sum_{i=0}^b 2^i - \sum_{i=0}^b i 2^i$$

- Let's now focus on the second term

$$\sum_{i=0}^b i 2^i = \sum_{i=0}^b \sum_{j=i}^b 2^j = \sum_{i=0}^b \left(\sum_{j=0}^b 2^j - \sum_{k=0}^{i-1} 2^k \right)$$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Analysis of Union

- We will now substitute the close form for each inner summation and continue simplifying

$$\sum_{i=0}^b i 2^i = \sum_{i=0}^b \sum_{j=i}^b 2^j = \sum_{i=0}^b \left(\sum_{j=0}^b 2^j - \sum_{k=0}^{i-1} 2^k \right)$$

$$\begin{aligned} &= \sum_{i=0}^b ((2^{b+1} - 1) - (2^i - 1)). \\ &= (b+1)(2^{b+1} - 1) - \sum_{i=0}^b (2^i - 1) \\ &= (b+1)(2^{b+1} - 1) - (2^{b+1} - 1 - (b+1)) \\ &= (b+1)(2^{b+1} - 1) - (2^{b+1} - 1 - (b+1)) \\ &= b 2^{b+1} + 1. \end{aligned}$$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of Union

- Let's now go back and plug this into our original work bound, i.e.,

$$\sum_{i=0}^b i 2^i = \sum_{i=0}^b \sum_{j=i}^b 2^j = \sum_{i=0}^b \left(\sum_{j=0}^b 2^j - \sum_{k=0}^{i-1} 2^k \right)$$

$$\begin{aligned} &= a \sum_{i=0}^b 2^i - \sum_{i=0}^b i 2^i. \\ &= a (2^{b+1} - 1) - (b 2^{b+1} + 1) \\ &= a 2^{b+1} - a - b 2^{b+1} - 1 \\ &= 2m(a - b) - a - 1 \\ &= 2m(\lg n - \lg m) - a - 1 \\ &= 2m(\lg \frac{n}{m} - a - 1) \\ &= O(m \lg \frac{n}{m}). \end{aligned}$$

Analysis of Union

- While the direct method may seem complicated, it is **more robust than the brick method**
 - It can be applied to analyze essentially any algorithm, whereas the Brick method requires establishing a geometric relationship between the cost terms at the levels of the tree.



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of Union

- **Removing the Assumptions**

- Of course, in reality, our keys in T_1 won't split subtrees of T_2 in half every time

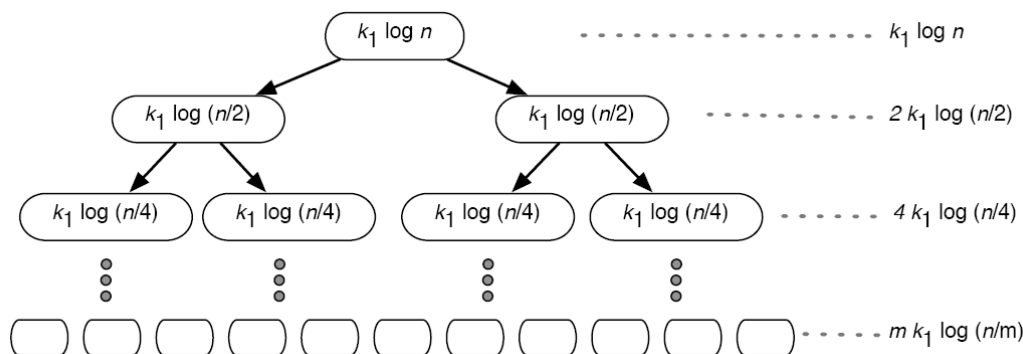
- **1. keep the assumption that T_1 is perfectly balanced**

- the shape of the recursion tree stays the same

- ✓ Let us try to analyze the cost at level i

- ✓ At this level, there are $k = 2^i$ nodes in the recursion tree

- ✓ Say the sizes of T_2 at these nodes are n_1, \dots, n_k , where $\sum_j n_j = n$



Analysis of Union

- 1. keep the assumption that T_1 is perfectly balanced
 - the total cost for this level is

$$c \cdot \sum_{j=1}^k \lg(n_j) \leq c \cdot \sum_{j=1}^k \lg(n/k) = c \cdot 2^i \cdot \lg(n/2^i),$$

- used the fact that the logarithm function is **concave**

- Thus, the tree remains leaf dominated and the same reasoning shows that the total work is **$O(m \lg(n/m))$**



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Analysis of Union

- 2. T_1 doesn't have to be perfectly balanced as we assumed
 - A similar reasoning can be used to show that T_1 only has to be approximately balanced.
 - We will leave this case as an exercise



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of Union

- Span

➤ the span of union is $O(\lg^2 n)$

how?

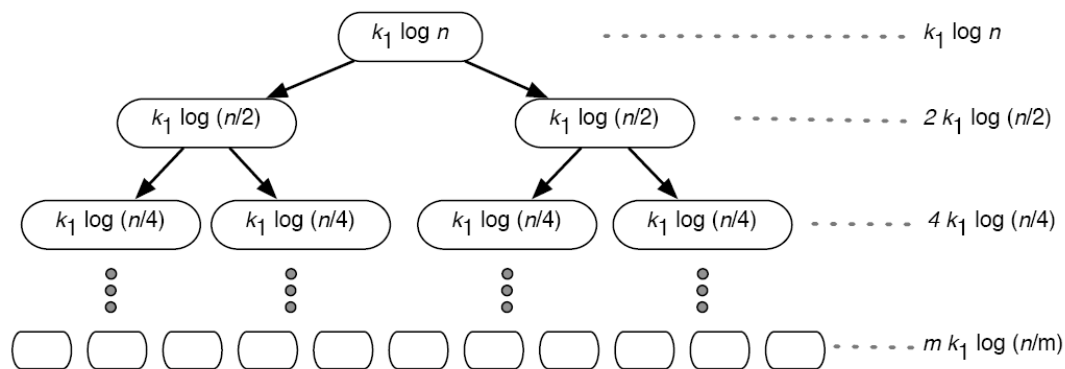
➤ but this can be improved to $O(\lg n)$ by changing the algorithm slightly

- In summary, union can be implemented in

➤ $O(m \lg(n/m))$ work and span $O(\lg n)$

```

union  $t_1$   $t_2$  =
  case ( $t_1, t_2$ )
  | (Leaf, _) =>  $t_2$ 
  | (_, Leaf) =>  $t_1$ 
  | (Node ( $l_1, k_1, r_1$ ), _) =>
    let ( $l_2, -, r_2$ ) = split  $t_2$   $k_1$ 
        ( $l, r$ ) = (union  $l_1$   $l_2$ ) || (union  $r_1$   $r_2$ )
    in joinM  $l$   $k_1$   $r$  end
  
```



明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- Implementation via Balancing
- A Parametric Implementation
- Cost Specification
- Treaps
- Augmenting Trees
- Exercises



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Treaps

- A **treap** is a randomized BST that maintains balance in a probabilistic way
- Each element/key gets a **unique random priority**
- The nodes in the treap satisfy **BST property**
 - Keys are stored in-order in the tree
- The associated priorities satisfy the **(max) heap property**



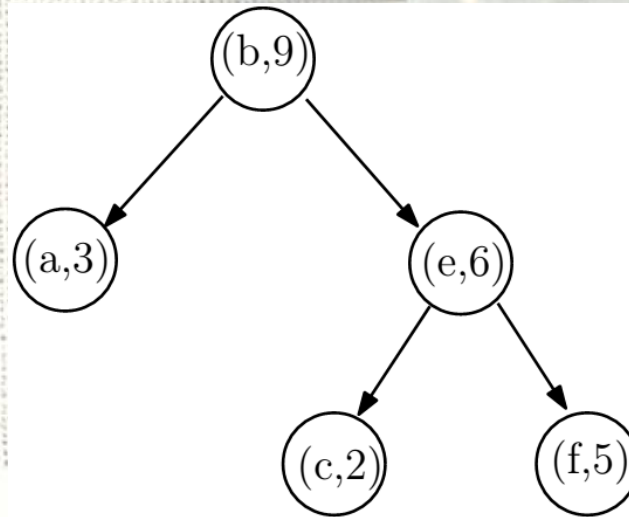
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明 The Max-heap Property

- Priority at each node is greater than the priorities of the children
- Suppose we have

$S=(a,3),(b,9),(c,2),(e,6),(f,5)$



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

treaps

- A treap is a binary search tree **T** over a set **S** along with a priority for each key given by
$$p : K \rightarrow Z$$
- that in addition to satisfying the BST property on the keys **S**, satisfies the heap property on the priorities $p(s)$, $s \in S$, i.e., for every node v :
$$p(k(v)) \geq p(k(L(v))) \text{ and } p(k(v)) \geq p(k(R(v)))$$
- where **k(v)** denotes the key of a node.



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Let's Do An Example

- Draw the treap for the following *(key, priority)* sequence

(G,50),(C,35),(E,33),(H,29),(I,25),(B,24),(A,21),
(L,16),(J,13),(K,9),(D,8)



華中科技大學

Exercise 12.17. Prove that if the priorities are unique, then there is exactly one tree structure that satisfies the Treap properties.

Implementing BST with Treaps

```
1 type T = Leaf | Node of (T × K × Z × T)
2
3 let empty = Leaf
4
5 singleton(k) = Node(Leaf, k, randomInt(), Leaf)
```

- **randomInt**
 - returns a (pseudo-)random number

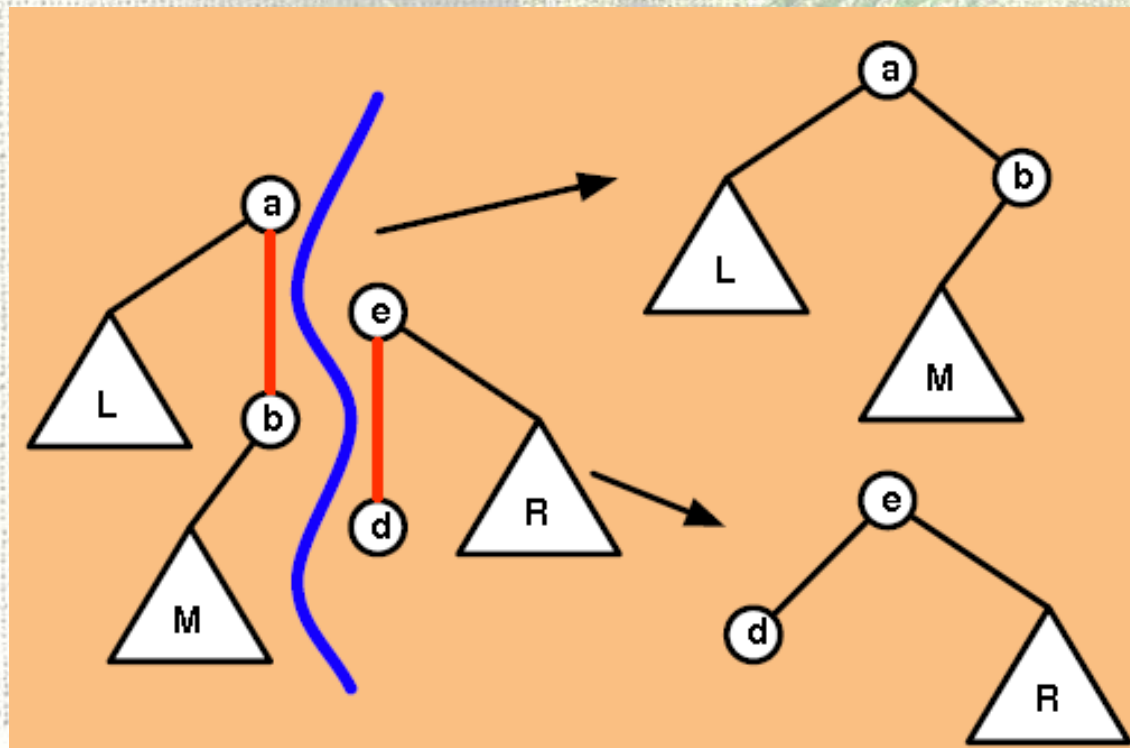
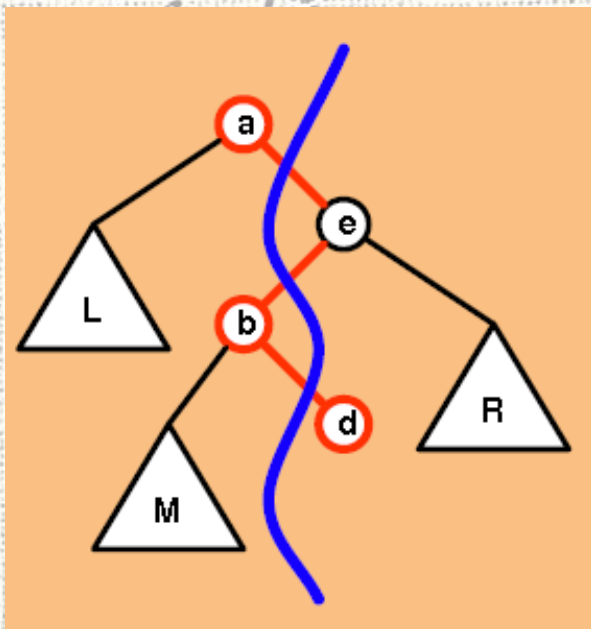


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Implementing split with Treaps

- Split (T, c)



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Implementing split with Treaps

```
7  split(T,k) =
8      case T
9      | Leaf  $\Rightarrow$  (Leaf, False, Leaf)
10     | Node (L,k',p',R) =
11         case compare (k,k')
12         | LESS  $\Rightarrow$ 
13             let (L',x,R') = split (L,k)
14             in (L',x,Node(R',k',p',R)) end
15         | EQUAL  $\Rightarrow$  (L,True,R)
16         | GREATER  $\Rightarrow$ 
17             let (L',x,R') = split (R,k)
18             in (Node (L,k',p',L'),x,R') end
```

- $T=\{(G,50),(C,35),(E,33),(H,29),(I,25),(B,24),(A,21),(L,16),(J,13),(K,9),(D,8)\}$
- `split(t,l)`

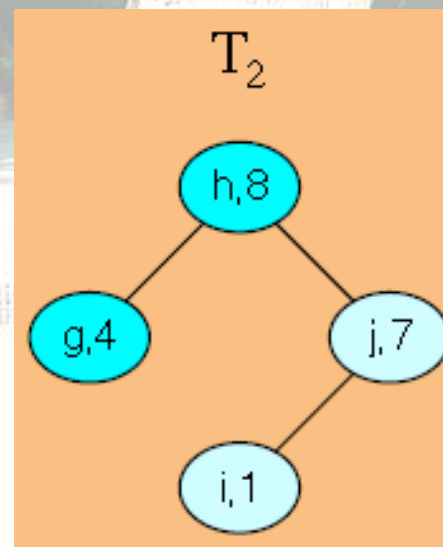
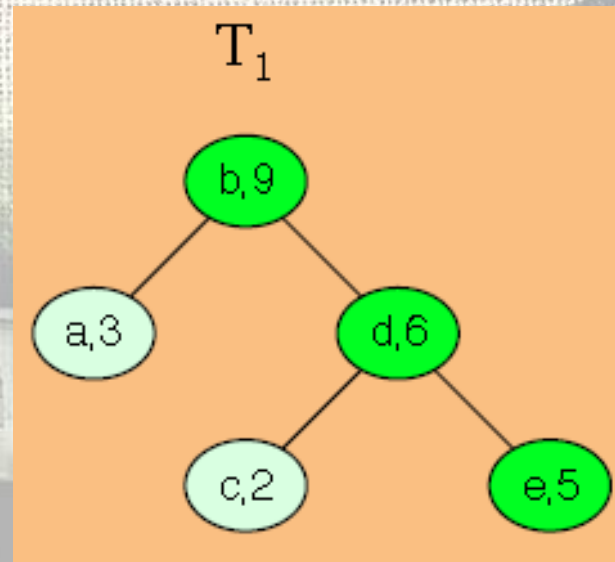
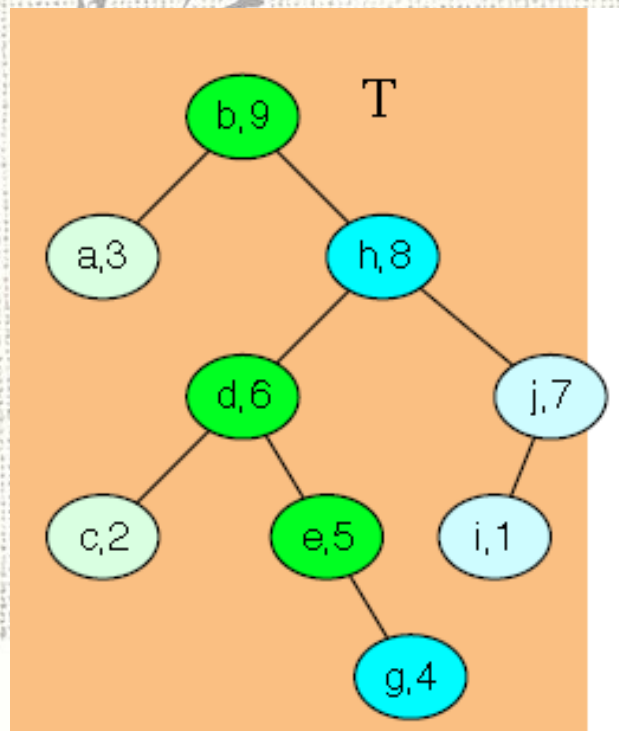


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

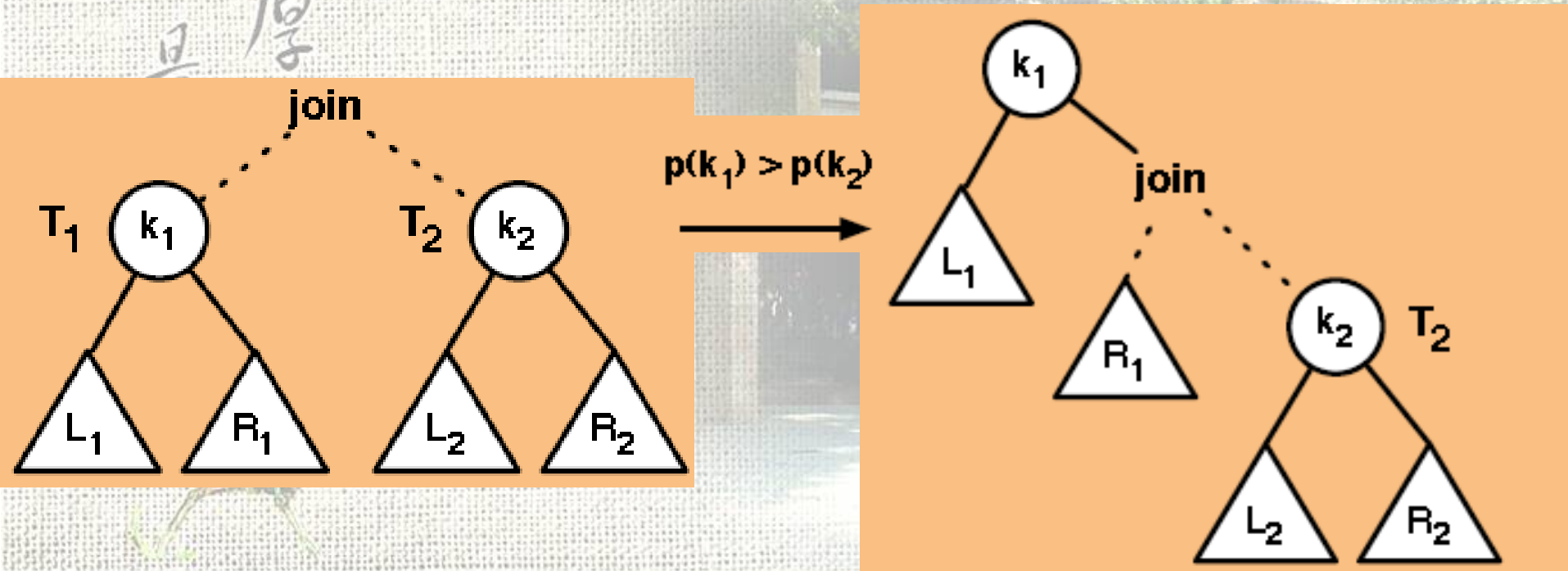
Implementing split with Treaps

- **Split(T, f)**



Implementing join with Treaps

- Join



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Implementing join with Treaps

```
21  join( $T_1, T_2$ ) =  
22      case ( $T_1, T_2$ )  
23      | (Leaf,  $\_$ )  $\Rightarrow T_2$   
24      | ( $\_ , \textit{Leaf}$ )  $\Rightarrow T_1$   
25      | (Node ( $L_1, k_1, p_1, R_1$ ), Node ( $L_2, k_2, p_2, R_2$ ))  $\Rightarrow$   
26          if ( $p_1 > p_2$ ) then  
27              Node ( $L_1, k_1, p_1, \textit{join}$  ( $R_1, T_2$ ))  
28          else  
29              Node ( $\textit{join}$ ( $T_1, L_2$ ),  $k_2, p_2, R_2$ )  
30  end
```

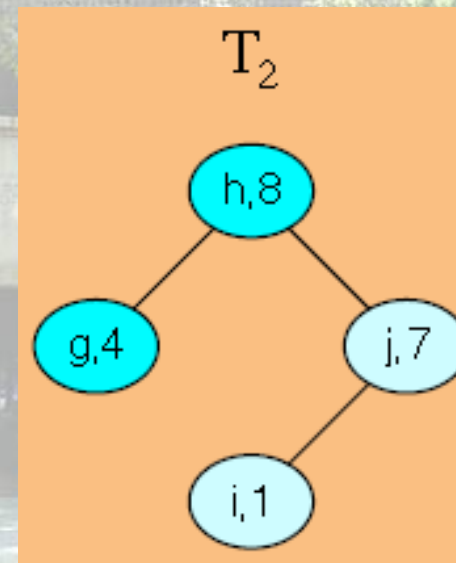
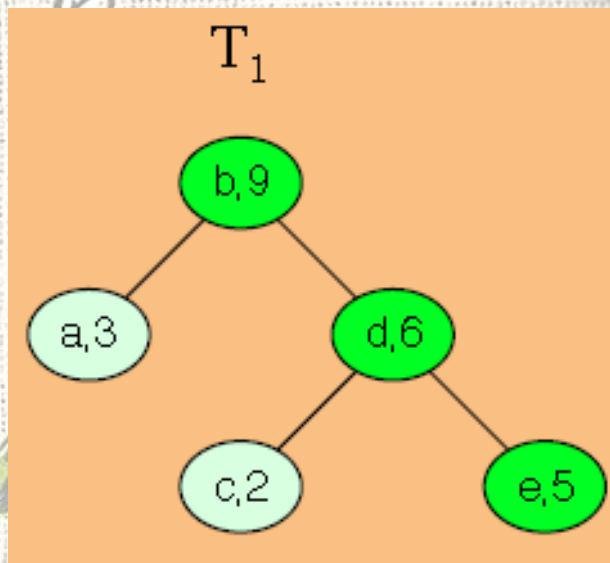


華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Implementing join with Treaps

- $\text{Join}(T_1, T_2)$



- Thinking: can we implement split and join just with BST?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Analysis of randomized treaps

- analyze the height of a treap assuming that the priorities are picked at random
- To do this we will relate treaps to quicksort

Algorithm 11.21. *Treap Generating Quicksort*

```
1  qsTree(S) =  
2    if  $|S| = 0$  then Leaf  
3    else let  
4      val pivot = the key  $k \in S$  for which  $p(k)$  is the largest  
5      val  $S_1 = \langle s \in S \mid s < pivot \rangle$   
6      val  $S_2 = \langle s \in S \mid s > pivot \rangle$   
7      val (L, R) = (qsTree( $S_1$ ) || qsTree( $S_2$ ))  
8    in  
9      Node(L, pivot, R)  
10   end
```

Analysis of randomized treaps

- The tree generated by $\text{qsTree}(S)$ is **the treap for S** , **why? Can you prove?**
- What does this tell us about the height of treaps?
 - the height of a treap is identical to the **recursion depth of quicksort**
 - if we pick the priorities at random, the recursion depth is **$O(\log n)$** with high probability



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Expected Max Depth of A Treap

- Expected depth of treap node is $O(\log n)$
 - Find takes on the average $O(\log n)$ work and span
- What is the **expected maximum depth of a treap**?
 - Why is this important?
 - Expected worst-case cost!
- But $E[\max_i\{A_i\}] \neq \max_i\{E[A_i]\}$!
- It turns out this is almost the same problem as the **expected span of the quicksort**



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SYNOPSIS

- Preliminaries
- The BST Abstract Data Type
- Implementation via Balancing
- A Parametric Implementation
- Cost Specification
- Treaps
- **Augmenting Trees**
- Exercises



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Augmenting Balanced Trees

- We can add other additional values to help with other search operations
 - Track key positions and certain subset sizes
- **rank(T, k)**
 - How many elements in T are less than k *or equal to* k ?
- **select(T, i)**
 - Returns the key with the rank i in T
- **splitIdx(S, i)**
 - Split S into two sets: first i keys and the remaining $n-i$ keys



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Augmenting Balanced Trees

- Let $T = \{1, 2, 3, 4, 5, 6\}$
- $\text{rank}(T, 4) = |\{1, 2, 3, 4\}| = 4$
- $\text{rank}(T, 4) = |\{1, 2, 3\}| = 3$
- $\text{select}(T, 4) = 4$ since $\text{rank}(S, 4) = 4$
- $\text{select}(T, 3) = 4$ since $\text{rank}(S, 4) = 3$
- $\text{splitIdx}(T, 3) = (\{1, 2, 3\}, \{4, 5, 6\})$

Which one is right?

Which one is right?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Augmenting Balanced Trees

- How to implement **Rank** (T, k)?

```
1  rank ( $T, k$ ) =  
2    case  $T$   
3    | Leaf  $\Rightarrow 0$   
4    | Node ( $L, k', R$ )  $\Rightarrow$   
5        case compare ( $k, k'$ )  
6        | LESS  $\Rightarrow$  rank ( $L, k$ )  
7        | EQUAL  $\Rightarrow |L|$   
8        | GREATER  $\Rightarrow |L| + 1 +$  rank ( $R, k$ )
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Augmenting Balanced Trees

- How to implement **select** (T, i)?

```
10 select ( $T, i$ ) =  
11   case ( $T$ )  
12   | Leaf  $\Rightarrow$  raise exception OutOfRange  
13   | Node ( $L, k, R$ )  $\Rightarrow$   
14       case compare ( $i, |L|$ ) of  
15         LESS  $\Rightarrow$  select ( $L, i$ )  
16         EQUAL  $\Rightarrow k$   
17         GREATER  $\Rightarrow$  select ( $R, i - |L| - 1$ )
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Augmenting Balanced Trees

- What is the work and span of these functions?
 - Rank (T, k): W=? S=?
 - Select (T, i): W=? S=?
 - Both: W=O(n), S= O(log(n))

how?

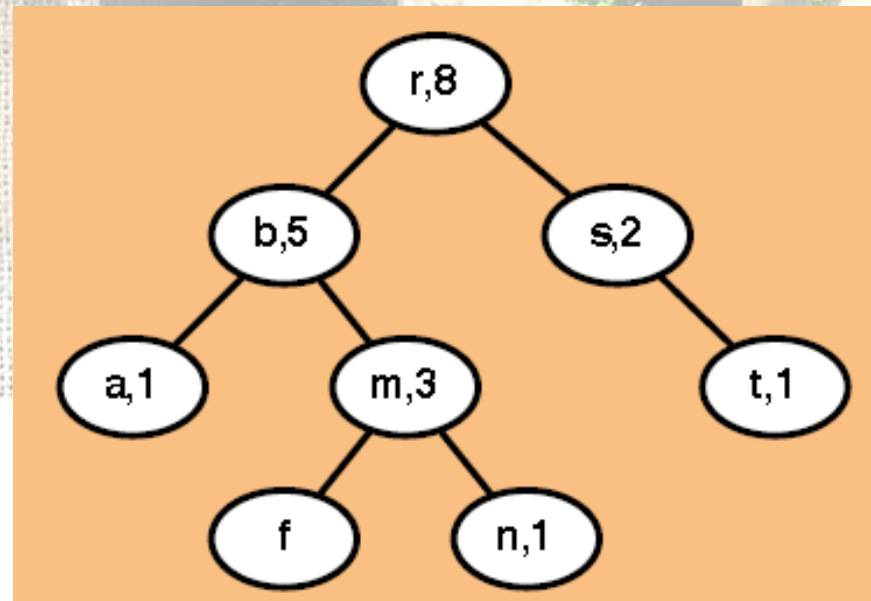
$$\text{rank}(S, k) : \mathbb{S} \times \mathbb{U} \rightarrow \text{int} = |\{k' \in S \mid k' < k\}|$$

$$\text{select}(S, i) : \mathbb{S} \times \text{int} \rightarrow \mathbb{U} = k \text{ such that } |\{k' \in S \mid k' < k\}| = i$$

$$\text{splitIdx}(S, i) : \begin{matrix} \mathbb{S} \times \text{int} \\ \mathbb{S} \times \mathbb{S} \end{matrix} \rightarrow \begin{matrix} \mathbb{S} \times \mathbb{S} \\ \mathbb{S} \times \mathbb{S} \end{matrix} = \begin{matrix} (\{k \in S \mid k < \text{select}(S, i)\}, \\ \{k \in S \mid k \geq \text{select}(S, i)\}) \end{matrix}$$

Augmenting Balanced Trees

- Can we compute size of subtrees more efficiently?
 - At each node keep the **size** of the subtree
 - This allows size and the three other operations in $O(d)$ work with d as the depth of the tree
 - Size can be computed on the fly by adding 1 to the sum of the subtree sizes!



Pairing nodes with reduced values

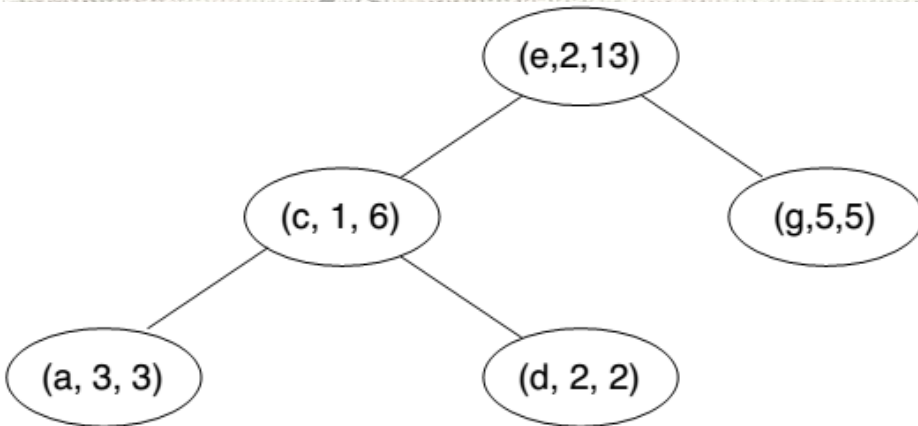
- Maintain at each node a “sum” based on an associative operator f
 - Updated during insert/delete, merger, extract, etc
- Given $f : v \times v \rightarrow v$, and I_f
 - All operations on ordered tables are supported, and
 - $reduceVal(A) : T \rightarrow v = reduce\ f\ I_f\ A$
 - We want to be able to do $reduceVal$ in $O(1)$ work (assuming f needs $O(1)$ work)
 - f is known beforehand!



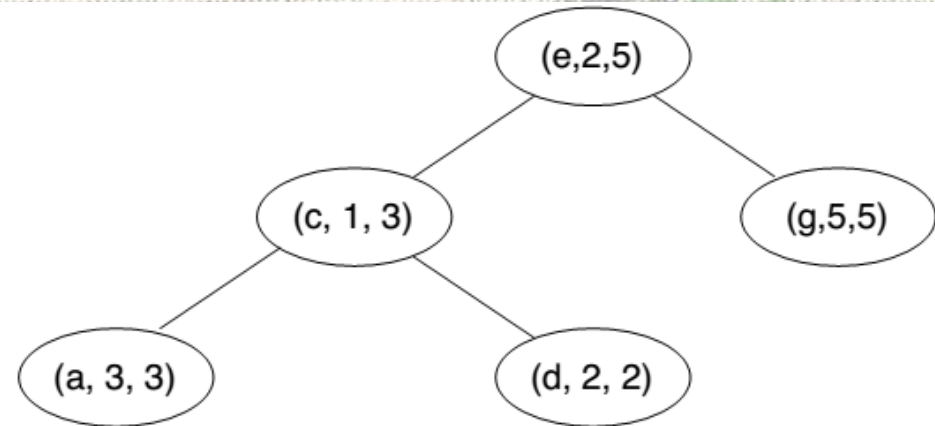
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

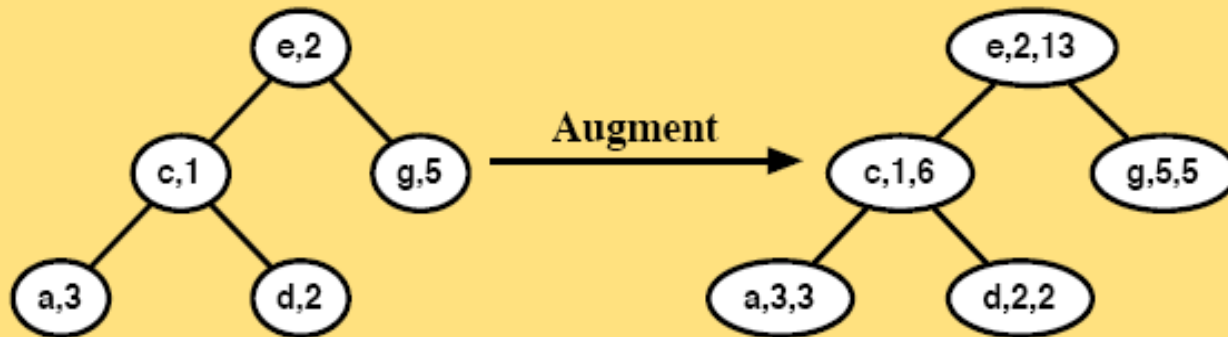
Pairing nodes with reduced values



f is +



f is max



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Implementation

道德

```
(* type of the reduced value, as specified. *)  
type rv = ...  
(* associative reducer function, as specified. *)  
f(x: rv, y: val, z: rv): rv = ...  
(* identity for the reducer function, as specified. *)  
idf : rv = ...  
  
type treap =  
  Leaf  
  | Node of (Treap × key × priority × (val × rv) × Treap)
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Implementation

- The only difference in the implementation of split and join functions is the use of **mkNode** instead of **Node**

```
rvOf t =  
  case t  
  | Leaf => idf  
  | Node (_,_,_(_,w),_) => w
```

```
mkNode (l,k,p,v,r) = Node (l,k,p,(v, f (rvOf l,v,rvOf r)),r)
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Implementation

- The only difference is the use of **mkNode** instead of **Node**

```
split t k =  
  case t  
  | Leaf => (Leaf, false, Leaf)  
  | Node (l, k', p', (v', w'), r) =  
    case compare (k, k')  
    | LESS =>  
      let (l', x, r') = split l k  
      in (l', x, mkNode (r', k', p', v', r)) end  
    | EQUAL => (l, true, r)  
    | GREATER =>  
      let (l', x, r') = split r k  
      in (mkNode (l, k', p', v', l'), x, r') end
```

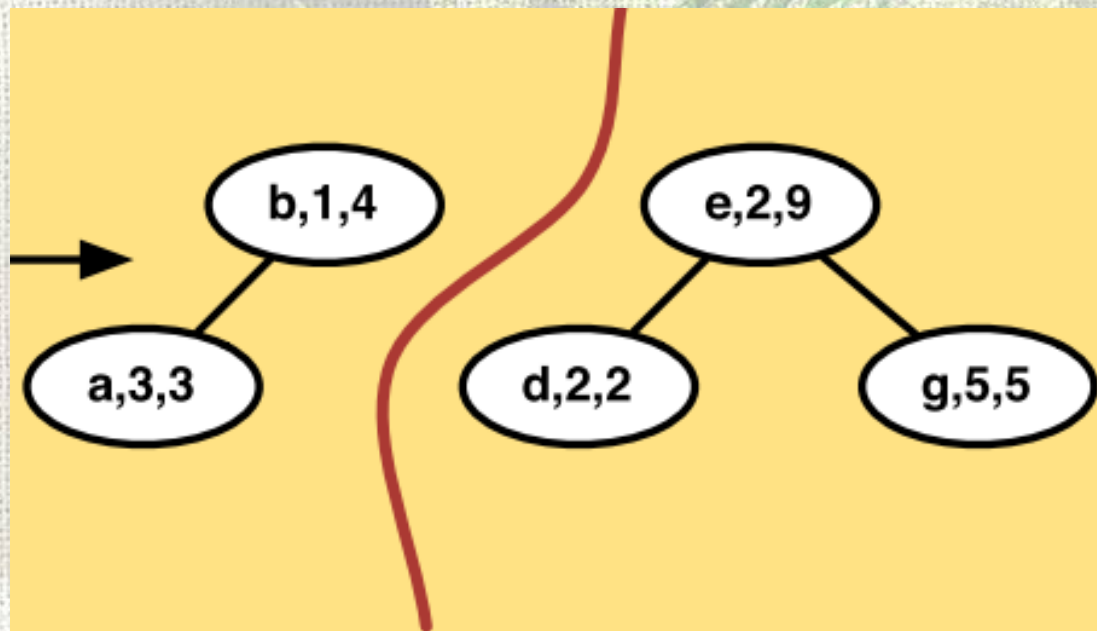
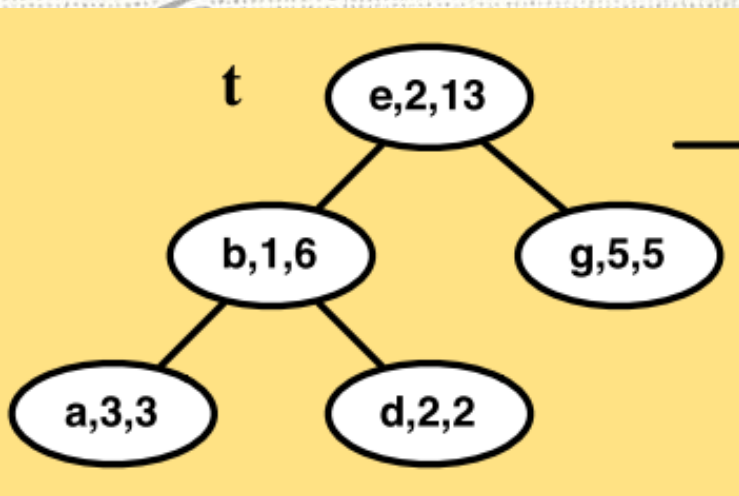
大學



明

Implementation

- Split t c



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Implementation

- The only difference is the use of **mkNode** instead of **Node**

```
join t1 t2 =  
  case (t1, t2) of  
    (Leaf, _) => t2  
  | (_, Leaf) => t1  
  | (Node (l1, k1, p1, (v1, w1), r1), Node (l2, k2, p2, (v2, w2), r2)) =>  
    if p1 > p2 then mkNode (l1, k1, p1, v1, join r1 t2)  
    else mkNode (join t1 l2, k2, v2, r2)
```



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Example Application – Sales Data

- Sales information are kept by the time stamp in an ordered table
 - (2/3/2013–12: 30, \$120)
- Find the total sales between t_1 and t_2
- f is +
- $reduceVal (getRange(T, t_1, t_2))$ takes $O(\log n)$ work

how?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Example Application – Stock Data

- Stock prices information are kept by the time stamp in an ordered table
 - (2/3/2013–12: 30, \$120/share)
- Find the maximum price between t_1 and t_2
- f is max
- $reduceVal (getRange(T, t_1, t_2))$ takes $O(\log n)$ work

how?



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Example Application – Interval Trees

- An **interval** is a region on the real number line starting at x_l and ending at x_r
- an interval table supports the following operations on intervals:

$insert(A, I) : \mathbb{T} \times (real \times real) \rightarrow \mathbb{T}$
 $delete(A, I) : \mathbb{T} \times (real \times real) \rightarrow \mathbb{T}$
 $count(A, x) : \mathbb{T} \times real \rightarrow int$

insert interval I into table A
delete interval I from table A
return the number of intervals crossing x in A

- **How to implement?**



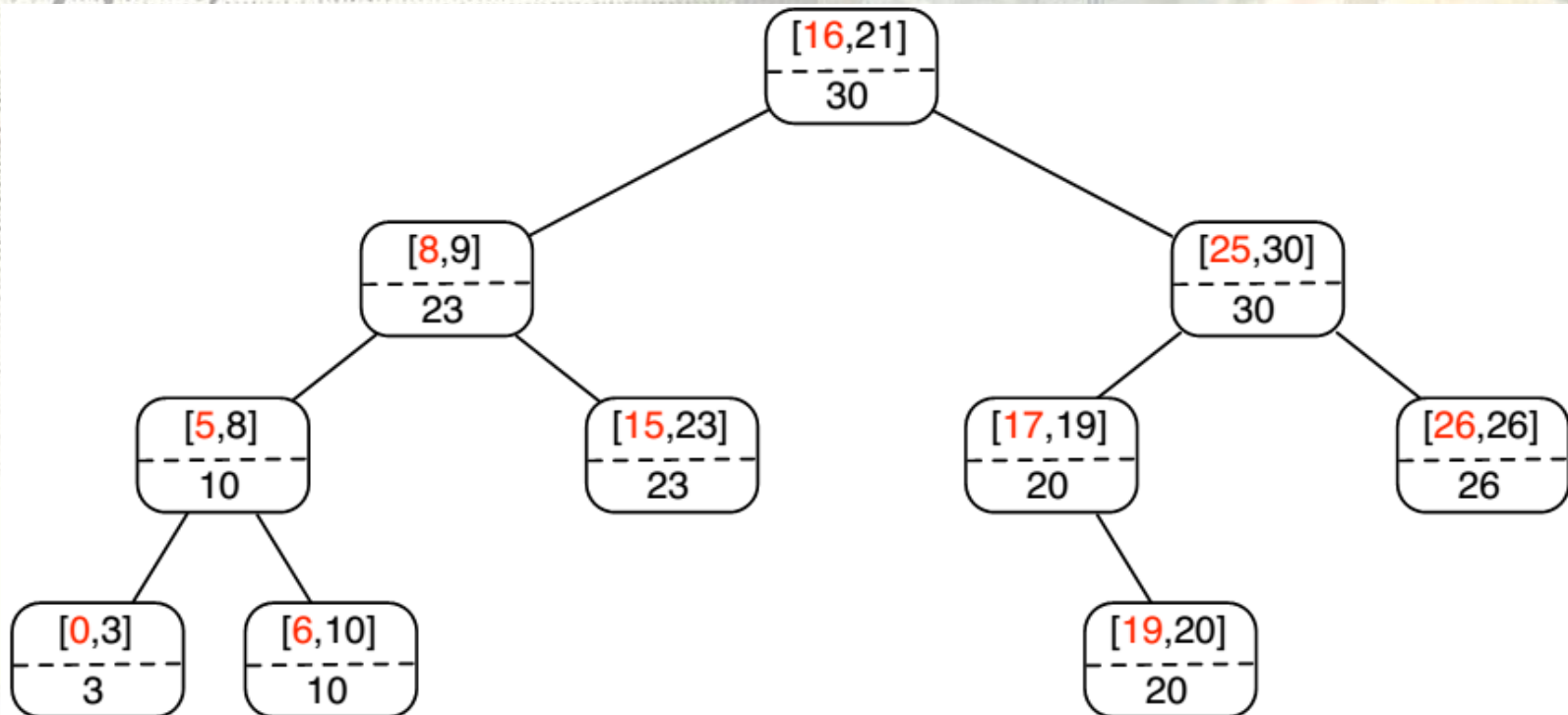
華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

Interval Trees

- Organize intervals as a BST based on **lower-boundary as key**
- Use the max upper boundary in the subtree as **additional information**



明

Counting Intervals

- How about the **Work and Span?**

```
1  datatype intTree = Leaf | Node of (intTree × intTree
2                                     × real × real × real)

3  fun overlap(x, low, high) =
4      if (x ≥ low & x ≤ high) then 1 else 0

5  fun countInt(T, x) =
6      case T of
7          Leaf ⇒ 0
8      | Node(L, R, low, high, max) ⇒
9          if (x > max) then 0
10         else countInt(L, x) +
11              overlap(x, low, high) +
12              if (x > low) then countInt(R, x) else 0
```


Exercises

12-3 Minimum height

Prove that the minimum possible height of a binary search tree with n keys is $\lceil \log_2(n+1) \rceil$.

12-4 Finding Ranges

Given a BST T and two keys $k_1 \leq k_2$ return a BST T' that contains all the keys in T that fall in the range $[k_1, k_2]$.

12-5 Tree rotations

In a BST T where the root v has two children, let u and w be the left and right child of v respectively. You are asked to reorganize T . For each reorganization design a constant work and span algorithm.

- **Left rotation.** Make w the root of the tree.
- **Right rotation.** Make u the root of the tree.



明

求德

Exercises

12-6 Size as reduced value

Show that size information can be computed as a reduced value. What is the function to reduce over?

12-7 Implementing `splitRank`

Implement the `splitRank` function.

12-8 Implementing `select`

Implement the `select` function using `splitRank`.



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

明

SUMMARY

- BST
- Balanced BST
- Treaps = Balanced BST? = QuickSort?
- Augmenting balanced BST



華中科技大學

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY