Separate Chaining Example

<table>
<thead>
<tr>
<th>element e</th>
<th>A</th>
<th>B</th>
<th>E</th>
<th>F</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash(e.hashCode())</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>stepHash(e.hashCode())</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Insert in order \{A, B, E, F, H, J\}

Table m = 8

Diagram of the chained hash table with elements A, B, E, F, H, J.
Ordered Collection ADT & Search Trees

Conceptual view \(<a, a, f, g, r, t, v, w, y>\)

**Methods**
- Iteration is sorted

```
most
- add, locate/search, remove
- min in ex α
- max in ex y
- successor in ex successor(k) is R
- predecessor in ex predecessor(k) is g
```

**Support these in logarithmic time**
Abstract Search Tree

Each node can hold any \# of elements.

Let $S$ be the \# elements (size) of a node.

Have $s+1$ children (possibly empty).
balance

allow some variation in path lengths to leaves (keep within a Factor 2)

allow some variation in node size but keep all path lengths the same
What is iteration order?

What is the predecessor of A?

What is the successor of D?
delete this node

min of this tree subtree is successor of X
Red-Black Tree

balanced binary search tree

Tool Rotation
rotate Left (Y)

rotate Right (Z)
liftUP(BSTNode y)

    parent = y. parent
    if (y. is Left Child())
        rotate Right (parent)
    else
        rotate Left (parent)

return parent
Red-black tree
guarantee longest path
$\sim 2 \log_2 n$

min, max, locate, insert, remove, successor, predecessor
$O(\log n)$
Red-Black Trees (Balanced Binary Search Trees)

**Inorder**

Root

Properties

- Every node is red or black
- Black Balanced: The number of black nodes on any path to a leaf is the same
- No Double Reds: No red node has a red child
- Root is Black

Black height of 2

Frontier (instead of null)