Today:

- Priority queues (Data Structures §6.4, §6.5)
- Range queries (§6.2)
- Java utilities: SortedSet, Map, etc.

Next topic: Hashing (Data Structures Chapter 7).

---

Priority Queues, Heaps

- Priority queue: defined by operations “add,” “find largest,” “remove largest.”
- Examples: scheduling long streams of actions to occur at various future times.
- Also useful for sorting (keep removing largest).
- Heap is common implementation.
- Enforces heap property: all labels in both children of node are less (or greater) than node’s label.
- So node at top has largest (or smallest) label.
- Are free to add smaller value to less bushy subtree, thus maintaining bushiness (keeping tree balanced).
- Insertion and deletion always proportional to $\log N$ in worst case.

---

Example: Inserting into a simple heap

Data:

1 17 4 5 9 0 -1 20

Initial Heap:

```
20
  /  \
17   9
 /   / \
5    4 0
  /  \
1 -1
```

Add 8: Dashed boxes show where heap property violated

```
20
  /  \
17   9
 /   / \
5    4 0
  /  \
1 -1
```

```
20
  /  \
17   9
 /   / \
5 8 4 0
  /  \
1 -1
```

```
20
  /  \
17   9
 /   / \
5 8 4 0
  /  \
1 5 -1
```

---

Heap insertion continued

Now insert 18:

```
20
  /  \
17   9
 /   / \
8 4 0 -1
  /  \
1 5 18
```

```
20
  /  \
17   9
 /   / \
8 4 0 -1
  /  \
1 5 18
```

```
20
  /  \
18   9
 /   / \
8 17 0 -1
  /  \
1 5 4
```

---
Removing Largest from Heap

To remove largest: Move bottommost, rightmost node to top, then re-heapify down as needed (swap offending node with larger child) to re-establish heap property.

Heaps in Arrays

- Since heaps are complete (missing items only at bottom level), can use arrays for compact representation.
- Example of removal from last slide (dashed arrows show children):

Ranges

- So far, have looked for specific items
- But for BSTs, need an ordering anyway, and can also support looking for ranges of values.
- Example: perform some action on all values in a BST that are within some range (in natural order):

\[
\text{/** Apply WHATTODO to all labels in } T \text{ that are} \\
\text{ * } \geq L \text{ and } < U, \text{ in ascending natural order. */} \\
\text{static void visitRange (BST } T, \text{ Comparable<Key> } L, \text{ Comparable<Key> } U, \\
\text{ Action whatToDo)} \\
\text{if (T != null) { \\
\text{ int compLeft = L.compareTo (T.label ()),} \\
\text{ compRight = U.compareTo (T.label ());} \\
\text{ if (compLeft < 0) } /* L < label */ \\
\text{ visitRange (T.left (), L, U, whatToDo);} \\
\text{ if (compLeft <= 0 && compRight > 0) } /* L <= label < U */ \\
\text{ whatToDo.action (T);} \\
\text{ if (compRight > 0) } /* label < U */ \\
\text{ visitRange (T.right (), L, U, whatToDo);} \\
\text{ } \\
\text{}}
\]

Time for Range Queries

- Time for range query \( \in O(h + M) \), where \( h \) is height of tree, and \( M \) is number of data items that turn out to be in the range.
- Consider searching the tree below for all values, \( x \), such that \( 25 \leq x < 40 \).
- In this example, the \( h \) comes from the starred nodes; the \( M \) comes from other non-dashed nodes. Dashed nodes are never looked at.
Ordered Sets and Range Queries in Java

- Class `SortedSet` supports range queries with views of set:
  - `S.headSet(U)`: subset of `S` that is < `U`.
  - `S.tailSet(L)`: subset that is ≥ `L`.
  - `S.subSet(L,U)`: subset that is ≥ `L`, < `U`.

- Changes to views modify `S`.
- Attempts to, e.g., add to a `headSet` beyond `U` are disallowed.
- Can iterate through a view to process a range:
  ```java
def SortedSet<String> fauna = new TreeSet<String>
  (Arrays.asList("axolotl", "elk", "dog", "hartebeest", "duck"));
  for (String item : fauna.subSet("bison", "gnu"))
      System.out.printf("%s, ", item);
```

- Java library type `TreeSet<T>` requires either that `T` be `Comparable`,
  or that you provide a Comparator:
  ```java
def SortedSet<String> rev_fauna = new TreeSet<String> (Collections.reverseOrder());
```

Example of Representation: BSTSet

- Use binary search tree to represent set. Can use same representation for both `BSTSet` and its subsets.
- Each set has pointer to BST, plus bounds (if any).
- In this representation, size is rather expensive!

```java
def SortedSet<String> fauna = new BSTSet<String> (collection of stuff);
def subset = fauna.subSet("bison","gnu");
def Iterator<String> i = subset.iterator ()
```

Diagram of BSTSet implementation.