CSE 326: Data Structures

Skew Heaps

Brian Curless

Spring 2008

Announcements (4/11/08)

- Written HW #2 due next Friday
- Project #2 Phase A out now
  - Partner sign-ups by 3pm today

Skew Heaps

Problems with <u>leftist</u> heaps

- Extra storage for npl
- Extra complexity/logic to maintain and check npl

Observation:

- Right side of leftist heap is "often" heavy and requires a switch.

Solution: <u>skew</u> heaps

- "Blindly" adjusting version of leftist heaps
- Merge *always* switches children when fixing right path
- <u>Worst case time</u>: merge, insert, deleteMin = O(*n*)
- <u>*Amortized* time</u>: merge, insert, deleteMin = O(log *n*)

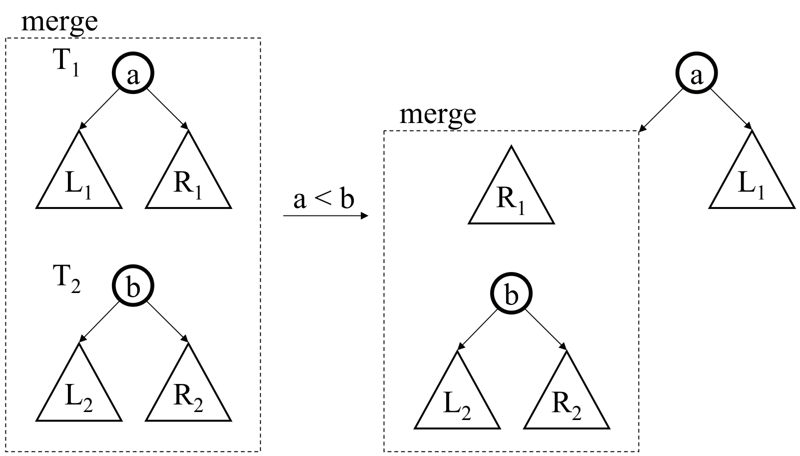
Amortized vs. average complexity

Recall:

- **Average-case complexity**: **avg** # steps algorithm takes on *random* inputs of size **N**
- **Amortized complexity (improved definition)**: **max** total # steps algorithm takes, in the worst case, for **M** *consecutive* operations on inputs of size **N**, divided by **M** (i.e., divide the max total by **M**).

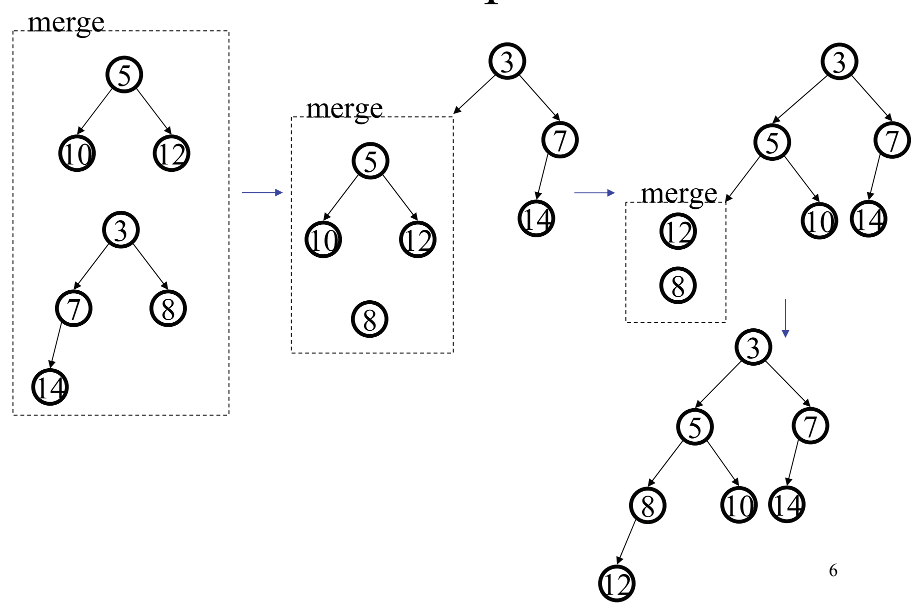
  Example: if M operations take total O(M log N) time in the worst case, *amortized* time per operation is O(log N).

Merging Two Skew Heaps



Only one step per iteration, with children *always* switched

Example



Skew Heap Code

```
SkewHeap merge(heap1, heap2) {
  case {
     heap1 == NULL: return heap2;
     heap2 == NULL: return heap1;
     heap1.findMin() <= heap2.findMin():
          temp = heap1.right;
          heap1.right = heap1.left;
          heap1.left = merge(heap2, temp);
          return heap1;
     otherwise:
          return merge(heap2, heap1);
  }
}
```


Runtime Analysis: Worst-case and Amortized

- No worst case guarantee on right path length!
- All operations rely on merge

  $\Rightarrow$ worst case complexity of all ops =

- It is known: M merges take time $\Theta(M \log n)$ in the worst case

  $\Rightarrow$ amortized complexity of all ops =