Announcements

- Homework 3 is out. Due 3/9
- Sample midterm problems on Courseworks
- Midterm review March 9th
- Midterm Exam March 11th
Review

- Solving the Young Tableaux Recurrences
- buildHeap description and analysis
- HW2 solutions
Today’s Plan

- Clarification about isomorphism
- buildHeap example
- HeapSort and HeapSelect
Isomorphism in Trees

- *iso* means equal, *morph* means shape
- Isomorphic means “having the same shape”
- Ignore left and right, element data
- Consider only *structural* properties:
Isomorphic Binary Trees
Not Isomorphic Trees
Not Isomorphic Trees
buildHeap

- Start at deepest non-leaf node
  - in array, this is node N/2
- `percolateDown` on all nodes in reverse level-order
  - for i = N/2 to 1
    percolateDown(i)
Recall the selection problem: \text{findKth}(k, A[\ ]) 

find the kth smallest element in A

Old method:

Sort the array, then return the kth element

Sort the first k elements, insert the remaining (N-k) elements in the proper place
HeapSelect

- run buildHeap on the array A
  - $O(N)$
- call deleteMin() $k$ times
  - $k O(\log N) = O(k \log N)$
- HeapSelect runs in $O(N+k \log N)$
HeapSort

- Naturally, we can use this idea to sort the array
- Method 1:
  - buildHeap on the array
  - copy output of deleteMin into new array N times
  - buildHeap costs $O(N)$, deleteMin costs $O(\log N)$
  - Heapsort 1 costs $O(N+N \log N)=O(N \log N)$
HeapSort in Place

- We don’t need to allocate a new array
- Instead, use a **max-heap**
  - Reverse the heap order property: deleteMax
- After each deleteMax, heap size is 1 less
  - Stick the extracted max in the freed space
HeapSort Animation

Downloaded from http://en.wikipedia.org/wiki/Heapsort
Assignments

- Continue HW3
- Weiss 7.5 if you want to read about HeapSort
- Practice midterm samples