CSE 373 Lecture 11: Binomial Queues

Today’s Topics:
- Binomial Queues
  - Merge
  - Insert
  - DeleteMin
  - Implementation
- Other Priority Queues: d-heaps, leftist, and skew heaps
- Covered in Chapter 6 in the text

Binomial Queues

- Binomial queues support all three priority queue operations: Merge, Insert, and DeleteMin in O(log N) time.
- Idea: Maintain a collection of heap-ordered trees
- **Forest of binomial trees**
- Recursive Definition of Binomial Tree (based on height k):
  - Only one binomial tree for a given height
  - Binomial tree of height 0 = single root node
  - Binomial tree of height k = B_k = Attach B_{k-1} to root of another B_{k-1}

3 Steps to Building a Binomial Tree

- To construct a binomial tree B_k of height k:
  1. Take the binomial tree B_{k-1} of height k-1
  2. Place another copy of B_{k-1} one level below the first
  3. Attach the root nodes

- Binomial tree of height k has exactly 2^k nodes (by induction)

Definition of Binomial Queues

Binomial Queue = “forest” of heap-ordered binomial trees

- Binomial Queue H1
  - 5 elements = 101 base 2
  - \( B_2 B_0 \)
- Binomial Queue H2
  - 11 elements = 1011 base 2
  - \( B_3 B_1 B_0 \)
Binomial Queue Properties

✦ Suppose you are given a binomial queue of N nodes
1. There is a unique set of binomial trees for N nodes
2. What is the maximum number of trees that can be in an N-node queue?
   ✗ 1 node → 1 tree B₀; 2 nodes → 1 tree B₁; 3 nodes → 2 trees B₀ and B₁; 7 nodes → 3 trees B₀, B₁, and B₂ ...

Number of Trees in a Binomial Queue

✦ What is the maximum number of trees that can be in an N-node binomial queue?
   ✗ 1 node → 1 tree B₀; 2 nodes → 1 tree B₁; 3 nodes → 2 trees B₀ and B₁; 7 nodes → 3 trees B₀, B₁, and B₂ ...
   ✗ Trees B₀, B₁, ..., B_k can store up to 2⁰ + 2¹ + ... + 2^k = 2^k+1 – 1 nodes = N.
   ✗ Maximum is when all trees are used.
   ✗ So, number of trees in an N-node binomial queue is ≤ (log(N+1)+1)+1 = O(log N)

Binomial Queues: Merge

✦ Main Idea: Merge two binomial queues by merging individual binomial trees
   ✗ Since B_k+1 is just two B_k's attached together, merging trees is easy
✦ Steps for creating new queue by merging:
   1. Start with B_k for smallest k in either queue.
   2. If only one B_k, add B_k to new queue and go to next k.
   3. Merge two B_k's to get new B_{k+1} by making larger root the child of smaller root. Go to step 2 with k = k + 1.

Binomial Queues: Merge Example

✦ Merge H1 and H2

H1:

H2:
Binomial Queues: Merge and Insert

✦ What is the run time for Merge of two O(N) queues?
✦ How would you insert a new item into the queue?

Binomial Queues: Merge and Insert

✦ What is the run time for Merge of two O(N) queues?
   ✅ O(number of trees) = O(log N)
✦ How would you insert a new item into the queue?
   ✅ Create a single node queue B₀ with new item and merge with existing queue
   ✅ Again, O(log N) time
✦ On-board example: Insert 1, 2, 3, …, 7 into an empty binomial queue

Binomial Queues: DeleteMin

✦ Steps:
  1. Find tree Bₖ with the smallest root
  2. Remove Bₖ from the queue
  3. Delete root of Bₖ (return this value); You now have a second queue made up of the forest B₀, B₁, …, Bₖ₋₁
  4. Merge this queue with remainder of the original (from step 2)
✦ Example: Insert 1, 2, …, 7 into empty queue and DeleteMin
✦ Run time analysis: Steps 1 through 4 = how much time for an N-node queue?

Binomial Queues: DeleteMin

✦ Steps:
  1. Find tree Bₖ with the smallest root
  2. Remove Bₖ from the queue
  3. Delete root of Bₖ (return this value); You now have a new queue made up of the forest B₀, B₁, …, Bₖ₋₁
  4. Merge this queue with remainder of the original (from step 2)
✦ Example: Insert 1, 2, …, 7 into empty queue and DeleteMin
✦ Run time analysis: Step 1 is O(log N), step 2 and 3 are O(1), and step 4 is O(log N). Total time = O(log N)
Implementation of Binomial Queues

* DeleteMin requires fast access to all subtrees of root
  - Need pointer-based implementation
  - Use First-Child/Next-Sibling representation of trees
* Merge adds one binomial tree as child to another
  - This added tree will now be the largest subtree
* Question: Should we order subtrees in increasing or decreasing size?

Other Priority Queues: d-Heaps

* Similar to a binary heap, except we allow more than 2 children per node
* d-heap has d children per node
* Example: 3-heap – root is A[1]; children of node A[i] are at what locations?

Other Priority Queues: d-Heaps

* Similar to a binary heap, except we allow more than 2 children per node
* d-heap has d children per node
* Just as in B-tree, more children means shallower heap
  - Depth is \( O(\log_d N) \) instead of \( O(\log_2 N) \)
  - But, \( d-1 \) comparisons to find smallest child
  - Tradeoff between depth and “breadth”
* Optimal \( d \) value is application dependent