IE 495 Lecture 11

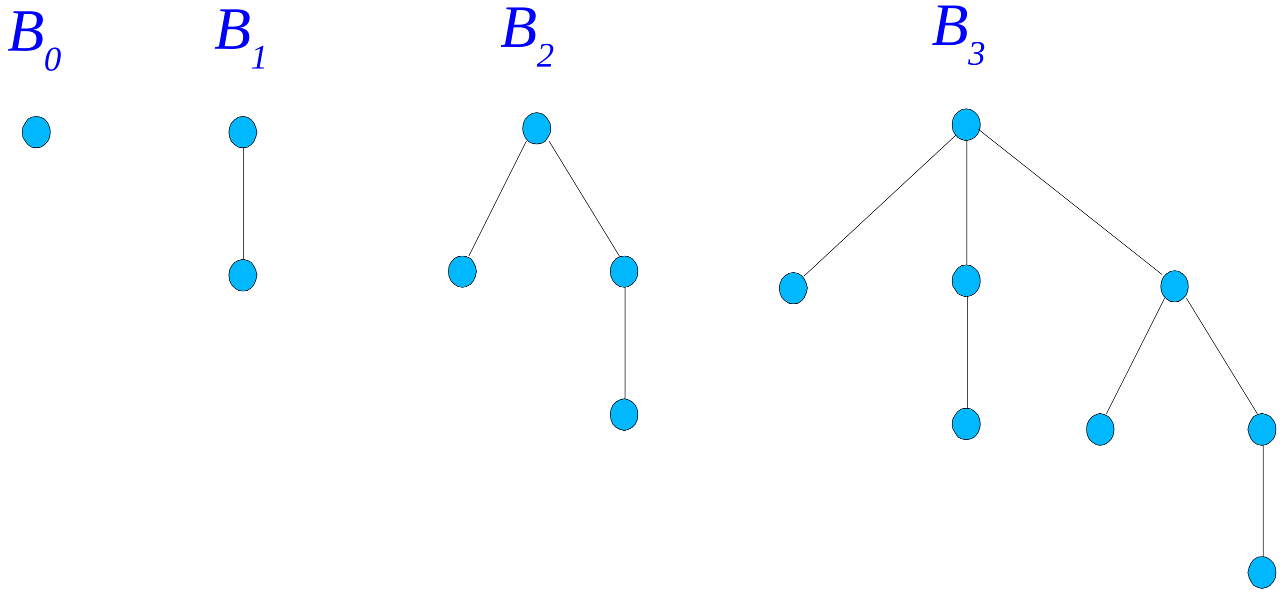
October 3, 2000

Reading for This Lecture

- Primary
  - Horowitz and Sahni, Chapter 2, Section 3
  - Kozen, Lectures 8-11

Binomial Trees

- The *binomial tree* of rank i (B_i) is defined recursively.
- B_i consists of a *root* with i children $B_0, \ldots, B_{i-1}$.



Binomial Heaps

- A *binomial heap* is a collection of heap ordered binomial trees and a pointer to the overall max/min.
- No more than one tree of each rank is allowed.
- The children of each vertex are maintained in a circular linked list.
- The basic operation is *linking*.
- Two trees of rank *i* can be combined into one tree of rank *i+1* in constant time.

Eager Meld

- We can combine two heaps by performing a *`meld()`* reminiscent of binary addition.
- Successively link trees of equal rank and "carry" one if necessary.
- Must track the position of the new min/max element.
- This operation takes $O(\log n)$ time.

Inserting into a Binomial Heap

- To `insert()` an element:
  - Make a new heap from the single element to be inserted.
  - Meld the new heap with the old one.
- To `make_heap()` from scratch, perform a sequence of inserts.
- To `delete()` the min/max element:
  - The children of this element form a new binomial heap.
  - Meld the old heap and the new one.

Amortized Analysis

- *`meld()`* and *`delete()`* both take O(log n).
- We will use *amortized analysis* to show that *`insert()`* is constant time overall.
- Idea: The total number of linking operations can never be more than the number of insert operations.
- This means that any sequence of inserts takes constant time *on average*.