

# Analysis Of Binomial Heaps

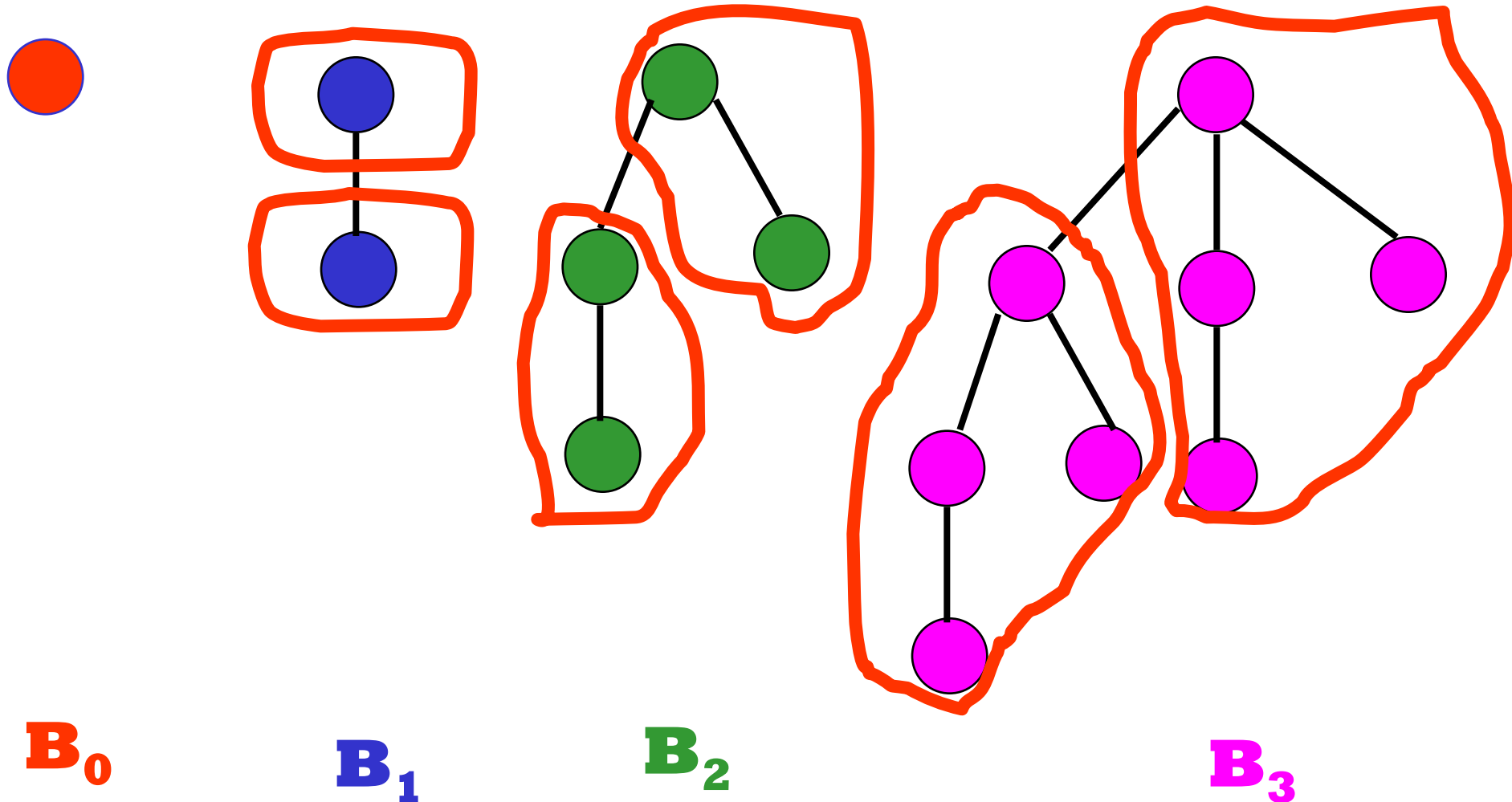
	Leftist trees	Binomial heaps	
		Actual	Amortized
Insert	$O(\log n)$	$O(1)$	$O(1)$
Remove min (or max)	$O(\log n)$	$O(n)$	$O(\log n)$
Meld	$O(\log n)$	$O(1)$	$O(1)$

# Operations

- Insert
  - Add a new min tree to top-level circular list.
- Meld
  - Combine two circular lists.
- Remove min
  - Pairwise combine min trees whose roots have equal degree.
  - $O(\text{MaxDegree} + s)$ , where  $s$  is number of min trees following removal of min element but before pairwise combining.

# Binomial Trees

- $B_k$ ,  $k > 0$ , is two  $B_{k-1}$ s.
- One of these is a subtree of the other.



# All Trees In Binomial Heap Are Binomial Trees

- Initially, all trees in system are Binomial trees (actually, there are no trees initially).
- Assume true before an operation, show true after the operation.
- Insert creates a  $B_0$ .
- Meld does not create new trees.
- Remove Min
  - Reinserted subtrees are binomial trees.
  - Pairwise combine takes two trees of equal degree and makes one a subtree of the other.

# Complexity of Remove Min

- Let  $n$  be the number of operations performed.
  - Number of inserts is at most  $n$ .
  - No binomial tree has more than  $n$  elements.
  - $\text{MaxDegree} \leq \log_2 n$ .
  - Complexity of remove min is  $O(\log n + s) = O(n)$ .

# Aggregate Method

- Get a good bound on the cost of every sequence of operations and divide by the number of operations.
- Results in same amortized cost for each operation, regardless of operation type.
- Can't use this method, because we want to show a different amortized cost for remove mins than for inserts and melds.

# Aggregate Method – Alternative

- Get a good bound on the cost of every sequence of remove mins and divide by the number of remove mins.
- Consider the sequence **insert, insert, ..., insert, remove min.**
  - The cost of the remove min is  $O(n)$ , where  $n$  is the number of operations in the sequence.
  - So, amortized cost of a remove min is  $O(n/1) = O(n)$ .

# Accounting Method

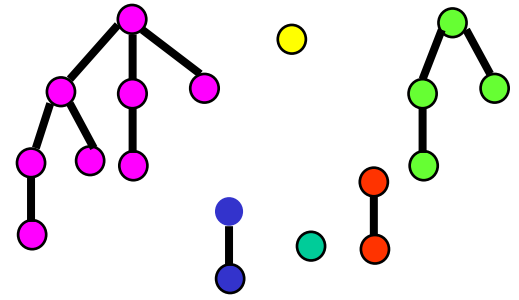
- Guess the amortized cost.
  - Insert  $\Rightarrow 2$ .
  - Meld  $\Rightarrow 1$ .
  - Remove min  $\Rightarrow 3\log_2 n$ .
- Show that  $P(i) - P(0) \geq 0$  for all  $i$ .



# Potential Function

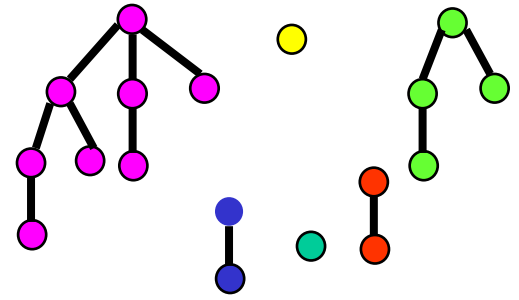
- $P(i) = \text{amortizedCost}(i) - \text{actualCost}(i) + P(i - 1)$
- $P(i) - P(0)$  is the amount by which the first  $i$  operations have been over charged.
- We shall use a credit scheme to keep track of (some of) the over charge.
- There will be 1 credit on each min tree.
- Initially,  $\#trees = 0$  and so total credits and  $P(0) = 0$ .
- Since number of trees cannot be  $< 0$ , the total credits is always  $\geq 0$  and hence  $P(i) \geq 0$  for all  $i$ .

# Insert



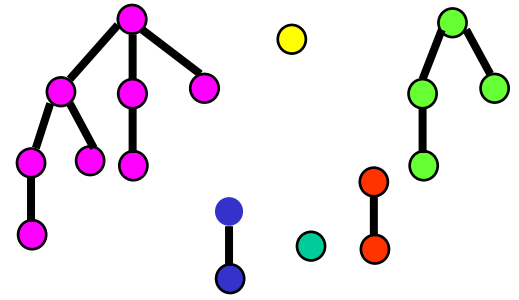
- Guessed amortized cost = 2.
- Use 1 unit to pay for the actual cost of the insert.
- Keep the remaining 1 unit as a credit.
- Keep this credit with the min tree that is created by the insert operation.
- Potential increases by 1, because there is an overcharge of 1.

# Meld



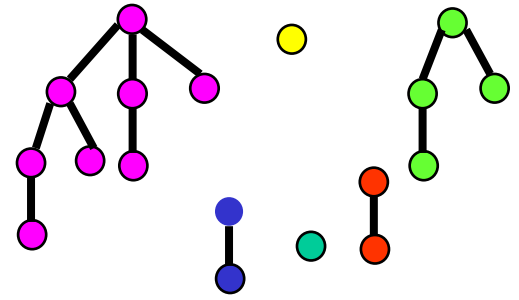
- Guessed amortized cost = 1.
- Use 1 unit to pay for the actual cost of the meld.
- Potential is unchanged, because actual and amortized costs are the same.

# Remove Min



- Let **MinTrees** be the set of min trees in the binomial heap just before remove min.
- Let **u** be the degree of min tree whose root is removed.
- Let **s** be the number of min trees in binomial heap just before pairwise combining.
  - $s = \text{\#MinTrees} + u - 1$
- Actual cost of remove min is  $\leq \text{MaxDegree} + s$   
 $\leq 2\log_2 n - 1 + \text{\#MinTrees}.$

# Remove Min



- Guessed amortized cost =  $3\log_2 n$ .
- Actual cost  $\leq 2\log_2 n - 1 + \# \text{MinTrees}$ .
- Allocation of amortized cost.
  - Use up to  $2\log_2 n - 1$  to pay part of actual cost.
  - Keep some or all of the remaining amortized cost as a credit.
  - Put 1 unit of credit on each of the at most  $\log_2 n + 1$  min trees left behind by the remove min operation.
  - Discard the remainder (if any).

# Paying Actual Cost Of A Remove Min

- Actual cost  $\leq 2\log_2 n - 1 + \text{\#MinTrees}$
- How is it paid for?
  - $2\log_2 n - 1$  comes from amortized cost of this remove min operation.
  - $\text{\#MinTrees}$  comes from the min trees themselves, at the rate of 1 unit per min tree, using up their credits.
  - Potential may increase or decrease but remains nonnegative as each remaining tree has a credit.

# Potential Method

- Guess a suitable potential function for which  $P(i) - P(0) \geq 0$  for all  $i$ .
- Derive amortized cost of  $i$ th operation using
$$\Delta P = P(i) - P(i - 1)$$
$$= \text{amortized cost} - \text{actual cost}$$
- $\text{amortized cost} = \text{actual cost} + \Delta P$

# Potential Function

- $P(i) = \sum \# \text{MinTrees}(j)$ 
  - $\# \text{MinTrees}(j)$  is  $\# \text{MinTrees}$  for binomial heap  $j$ .
  - When binomial heaps  $A$  and  $B$  are melded,  $A$  and  $B$  are no longer included in the sum.
- $P(0) = 0$
- $P(i) \geq 0$  for all  $i$ .
- $i$ th operation is an insert.
  - Actual cost of insert = 1
  - $\Delta P = P(i) - P(i - 1) = 1$
  - Amortized cost of insert = actual cost +  $\Delta P$   
 $= 2$



# **i**th Operation Is A Meld

- Actual cost of meld = 1
- $P(i) = \sum \# \text{MinTrees}(j)$
- $\Delta P = P(i) - P(i - 1) = 0$
- Amortized cost of meld = actual cost +  $\Delta P$   
= 1

# **i**th Operation Is A Remove Min

- **old**  $\Rightarrow$  value just before the remove min
- **new**  $\Rightarrow$  value just after the remove min.
- $\#MinTrees^{old}(j)$   $\Rightarrow$  value of **#MinTrees** in **j**th binomial heap just before this remove min.
- Assume remove min is done in **k**th binomial heap.

# ith Operation Is A Remove Min

- Actual cost of remove min from binomial heap **k**  
 $\leq 2\log_2 n - 1 + \#MinTrees^{old}(k)$
- $\Delta P = P(i) - P(i - 1)$   
 $= \sum [\#MinTrees^{new}(j) - \#MinTrees^{old}(j)]$   
 $= \#MinTrees^{new}(k) - \#MinTrees^{old}(k).$
- Amortized cost of remove min = actual cost +  $\Delta P$   
 $\leq 2\log_2 n - 1 + \#MinTrees^{new}(k)$   
 $\leq 3\log_2 n.$