CSL 356 Aug 16

Binomial Heaps

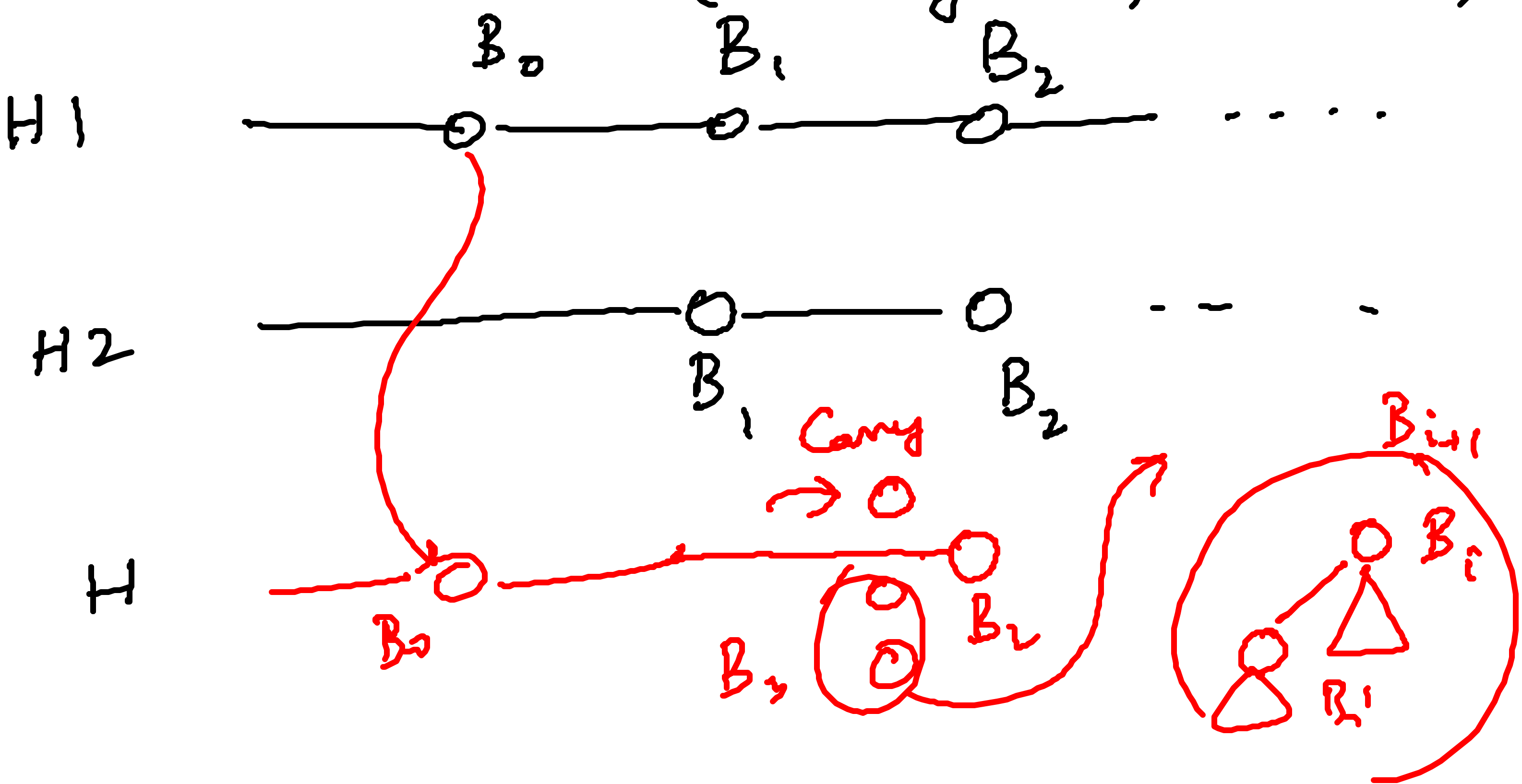
to support fast union of heaps in time $O(\log(n))$ where

n_1 = # elements in Heap 1
n_2 = # elements in Heap 2

Set of Binomial Trees

B_i : contains 2^i nodes

Any set (any number n) can be expressed as a union of $\log(n)$ Binomial Trees (binary representation)



The total number of operations (cutting and joining of Binomial trees)

is $O(\max(\log(n_1), \log(n_2)))$

which is $O(\log(n_1+n_2))$

Total cost is logarithmic for creating a union of two Binomial Heaps / Melding of Heaps

<u>Make heap</u> (S) : First figure out which order of trees will be required and partition the elements accordingly (in any arbitrary order) : $O(n)$

For any specific B_i create B_{i-1} (two)
- then make the larger root the leftmost child of the smaller root

Total $O(|S|)$ cost $O(|B_i|)$

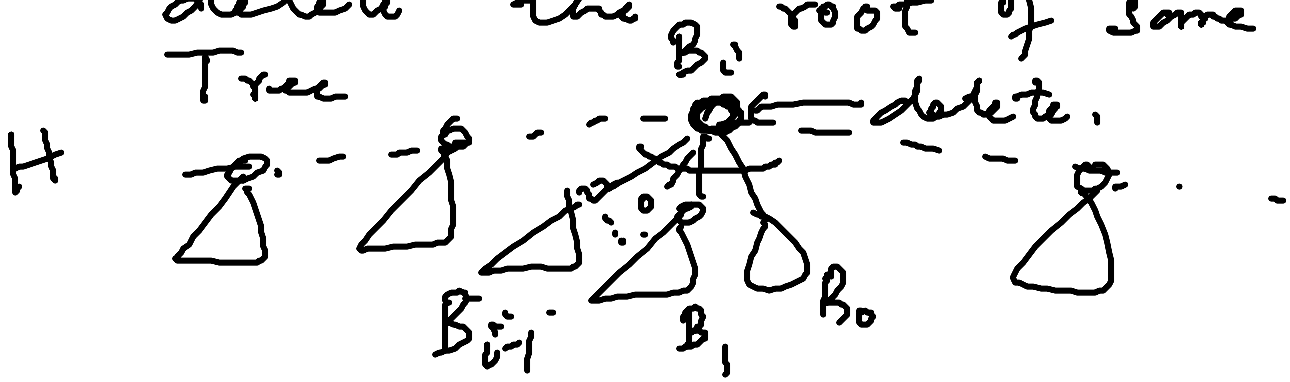
Insertion : Create a Binomial Heap with the node to be inserted

Union the two heaps

Time $O(\log(n))$

Deletion : A simple case for deletion happens when we delete the root of some Binomial Tree

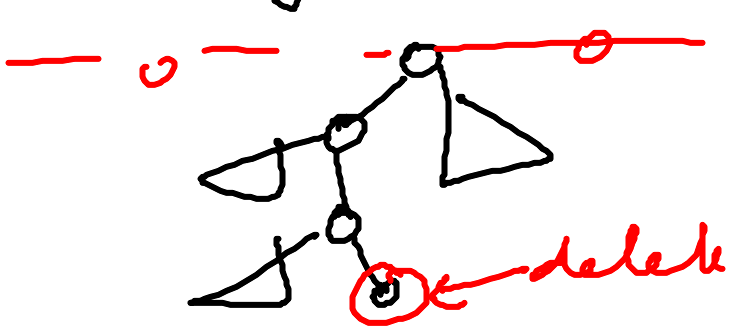
↗ a pointer to the node is given



Create another heap with the children of the deleted node, say H'

Union (H, H') Time : $O(\log n)$

Case : Arbitrary node :



Decrease the value to say $-\infty$

Decrease key takes $O(\log n)$
(by swapping with parent as long as necessary).

Total time: length of path + delete root

$= O(\log n)$

Dictionary Data structure

Supports search, updates (insert, ~~delete~~)

Semi-dynamic

Suppose we have at some juncture n elements, say $n = 11$

$n = 2^3 + 2^1 + 2^0$

We maintain 3 sorted arrays of elements, say A_3, A_1, A_0

$|A_3| = 2^3 \quad |A_1| = 2^1 \quad |A_0| = 2^0$

Within each array we store the elements in sorted order.

Search (x) : Search for x in A_3, A_1, A_0

A_3 : $[50 \ldots\ldots\ldots 5]$

A_1 . $[100 \ldots 2]$

$x = 10$ In general for a set with n elements, we may have to search $\log n$ sorted arrays

$\Rightarrow$ Time : $O(\log^2 n) = (\log n)^2$

<u>Insertion</u> . $n \to n+1$

$11 \to 12$

$A_3 \ A_1 \ A_0$ $\qquad$ $A_3 \ A_2$

In general it would lead to creation/ destruction of sorted arrays

$$\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

destroyed

created

not affected

$$0\underbrace{111111}_{j\text{ bits}} \to \underset{2^j}{1}\overbrace{\underbrace{0000}_{j\text{ bits}}}^{2^j-1}$$

What is Time to create A_j ?

If we sort $\Rightarrow O\left(2^j \log(2^j)\right) = j \cdot 2^j$

Claim : Can be done in $O(2^j)$

Quiz (in class) on 22nd Aug (Th)

20 mins