COL 702 Lecture 5

Binomial Heaps

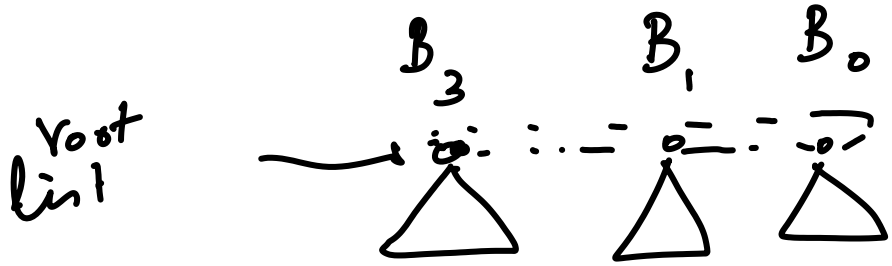
Consists of a set of Binomial Heap ordered trees.

B_i : Binomial tree of order i having 2^i nodes

Given a set of n elements, we can partition n into subsets of size 2^i, s.t. there is no more than one subset of size 2^i

Binary representation of n

Example $n = 11$ 1011



Obs : # Binomial trees is $\leq \lceil \log n \rceil$ since n has a $\log n$ bit binary rep

Min : We can find min by searching
- the root list : $\log n$

By maintaining an explicit reference
to min : $O(1)$ time
(Repointing not recomputing)

Merging of Binomial Heaps :
(Melding)

H_1 : B_1 B_3 B_5 B_6

H_2 : B_0 B_1 B_2 B_5

By simply linking H_1 H_2 we could
isolate

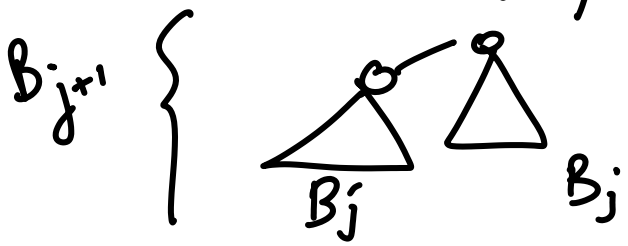
① There may more than one tree of order
i

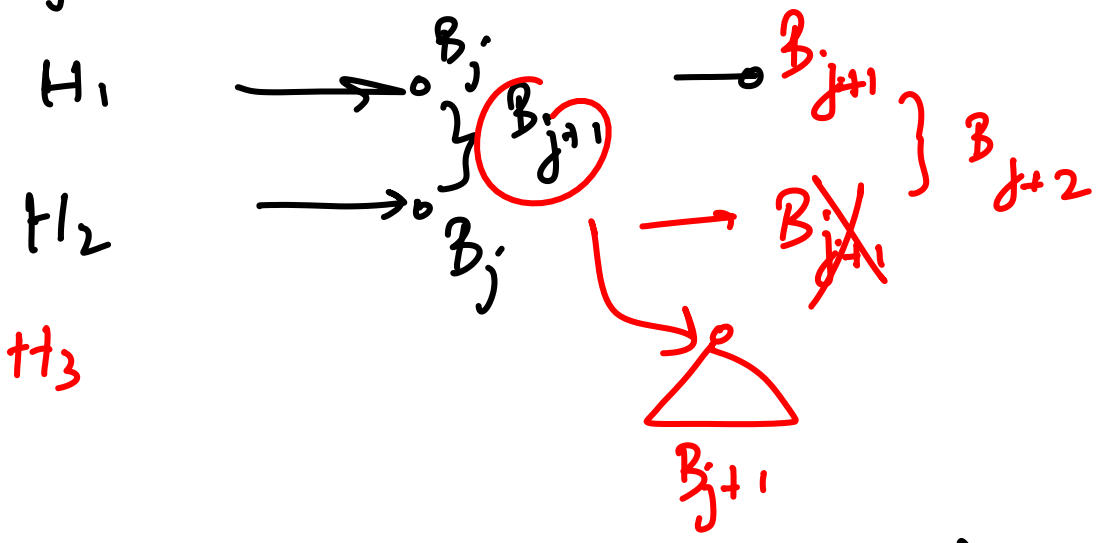
② # trees could be $> \log |H_1| + |H_2|$

Merging two root lists of Binomial Heaps
is similar to binary addition

Procedure

Starting from the lower order trees if there are two trees of order j, we combine them (using the recursive defn) into B_{j+1}





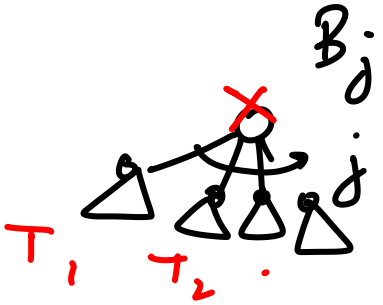
Time : $O(\log |H_1| + \log |H_2|)$

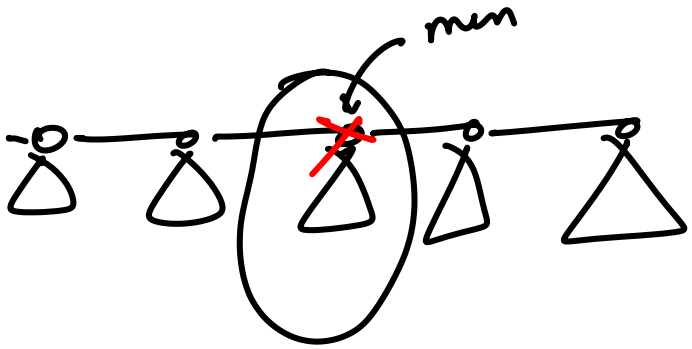
:

: $O(\log n)$

where $n = |H_1| + |H_2|$

Extract min :





① Delete root

② Create a new root list from the children H'

Merge H and H'

③ Find new min

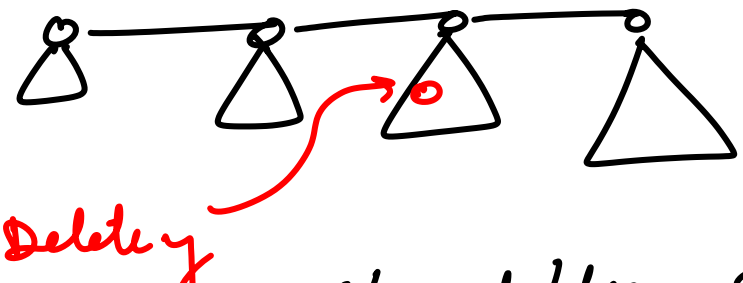
Time $O(\log n)$

Insert : a new element x to H

Create a heap consisting of singleton $\{x\}$

Merge $\{x\}$ with H

Delete :



Delete y, build a root list of its children and merge with the original root list

Decrease Key : 'Bubble' up the element (compare and swap)

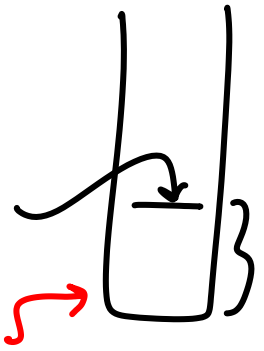
- till it is greater than its parent (It may end at root)

Maintaining Stacks under ops

push, pop, empty. queue

- push: one element — $O(1)$ — P_u
- pop: top of stack — $O(1)$ — P_o
- empty. queue: - Throw out all existing elements — $O(k)$ — E

k is the current # elements in stack



Consider a sequence of ops of the form O_i

P_u P_u P_o P_u P_o E P_u P_u P_u P_o

O_1 O_2 O_3 O_4

What is the total cost of n such operations?

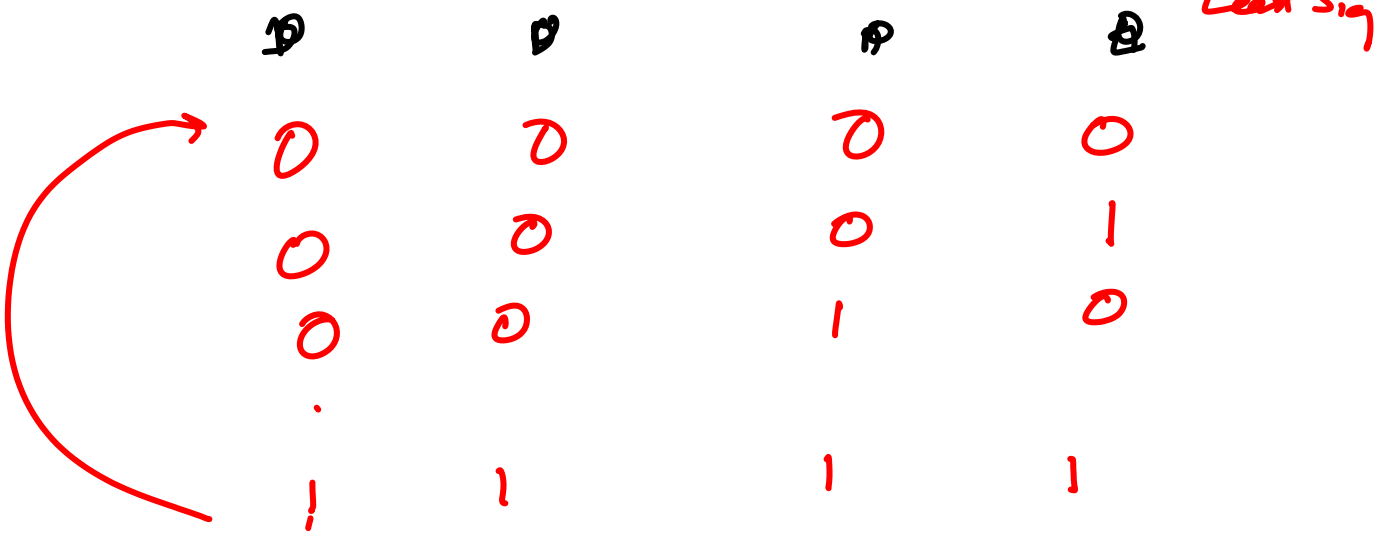
Worst case time for an operation $(E): O(n)$

Total of n operations $\Rightarrow$ $O(n^2)$

The cost of E is k if there are k elements in the stack

Aggregate analysis is known as "amortized analysis"

Example : Consider a n bit counter (binary)



In a single cycle, how many total bit flips happen?

$$\sum_{i=0}^{n-1} \frac{2^n}{2^i} \leq O(2^n)$$ instead the simplistic $n \cdot 2^n$