CS3000
5/24 - Weas

Admin

- Long Hw3 out tomorrow, due 5/31
- no OH memorize day
- exam tomorrow!

Agenda

1. Greedy Huffman
2. Data structure overview
3. Exam summary

Recap

- Huffman - what to optimize for?
  min. # of bits
- Huffman - what was greedy choice?
  two lowest-frequencies
- what to look for in the tree?
  high freq at the top
  low freq ~~how~~ at the bottom

11. Greedy Huffman

- create a tree w/ frequencies, characters
- left = 0, right = 1
- goal: min # bits when tree is used for compression

Lingering questions

- Is 2 lowest-freq an optimal choice? (us)
- How do we actually implement? (you)

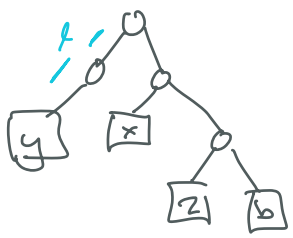
→ choose 2 lowest freqs → x, y
go on the bottom

define
$z = x.freq + y.freq$

what if they're not on the bottom?

- 2 other ones, a and b, are on the bottom
- $c = a.freq + b.freq$

$z \leq c$



→ total # bits
rest of tree + $l \cdot z + (l+1) \cdot c$

$lz + lc + c$

what if x, y are swapped with a, b?

total # of bits
rest of tree + $l \cdot c + (l+1) z$

$lc + lz + z$

$z \leq c$

2. Data Structures

naive → D+C → DP → Greedy

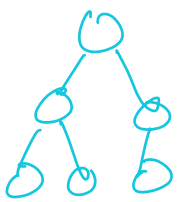
trying to do better with each type

Sometimes, can we do better? it's with the way we organize our data

sometimes, we need/want our data organized differently just to make our algo work

→ Binary Search

ex Binary Search Trees



left = = smaller

right = = bigger

- assume balanced (not default)

operations:

- search for a key O(lgn)
- insert a node
- remove a node

Binary Tree → Huffman

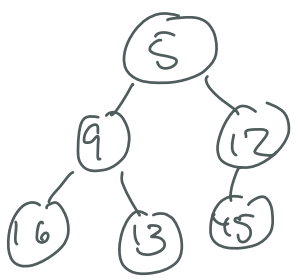
↳ Actually a heap!

- looks like a Binary Tree
- has an underlying array

Heap property

- every node is smaller than everything below

Ex Freqs: 5 9 12 16 13 45



operations

- extract-min (H)
  - remove the root/min
  - returns it
  - rearrange remaining elements
- insert (H, e)
  - insert element e into heap
  - maintain heap property

Trees in general

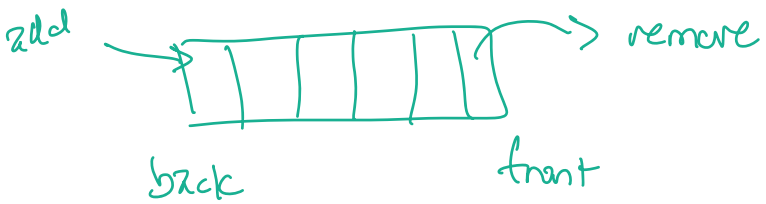
↳ If you have the root, you have the whole tree

Sets

- no dups
- order doesn't matter
- operations: union, intersection, complement

Queue

- FIFO (first in, first out)



operations

- enqueue(Q, e)
  - put element e in back of Q
- dequeue(Q)
  - remove + return element from front of Q

10:53

Exam ~ Tomorrow!

- in class
- on paper → solutions in designated areas; back pages just for scratch paper
- extra scratch paper
- 8.5 x 11 inch cheat sheet (one sided)
- no other notes/devices
- 90 minute exam → end at 11:30

On the exam...

- new algorithm → prove correctness
- familiar algorithm → use in a different way
- Run time →
  - given iterative also, count up # steps

    t_i = # times loop test is executed when $i = 1, 2, \ldots$

    (we'll want a summation)
  - sum up all the steps, get total
  - put a bound on
  - given D+C algo, give a recurrence

    (probably not solve the recurrence)

    - state the bound
- DP →
  - given a recursive formula, write the table (and/or) give pseudocode

- given a description of a D&C algo, give pseudocode