CS3000
5/23 - Tue.

Admin

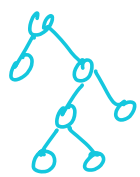
- Long HW2 due 9pm
- today's recitation: unofficial
- Solns to practice problems are on piazza
- next HW goes out thurs
- exam thurs in class

Agenda

1. Greedy Strategy
2. Huffman codes
3. Greedy Huffman

Recap

Binary tree every node has ≤ 2 children



Binary Search tree left == smaller
right == bigger

What does a greedy algo do?
makes locally best choice

Greedy choice for scheduling problem
earliest finish time

II. Greedy Strategy

- make a sequence of choices
- at each decision point, pick the choice that seems best at the time

Greedy vs. DP

- DP also makes a sequence of choice
- current choice depends on earlier choices

(ex: $c[i,j] = c[i-1,j] + 1$)

Risky Pieces

- Pick DP solution when greedy could work
  - ↳ maybe greedy would work too, maybe more efficient (time/space)
- Pick greedy when DP is needed
  - ↳ greedy might not give optimal solution !!

(ex) scheduling problem

greedy choice: earliest finish time

optimizing for: # activities

↳ is this right?

↳ reminder :)

S_k — subproblem

$a_m \in S_k$ ~ earliest finish time

there is an optimal solution A_k

- is $a_m \in A_k$? If so, great!
- if not, can we swap a_m into A_k?

  $a_j \in A_k$ with earliest finish time

  $f[j] \geq f[m]$

  so, safe to swap in a_j, a_m

$\boxed{\text{But}}$ If we picked earliest start time?

a_m has earliest start time

a_j has earliest start time in optimal

- can we swap a_m, a_j? no :(

  b/c a_m might finish too late

2. Huffman Codes

- used to compress data
- In general, we want to rep the same information, but ideally in less space

ex: Laney Is Awesome Laney Is Awesome

file.txt

LaneyIsAwesome x 1000

total characters: (5+2+7) x 1000

14,000 chars

Compress - replace something with something shorter

Laney - A

Is - B

Awesome - C

File ABC ABC ABC

ABC x 1000

total chars: 3000

File compression as optimization problem

- size of file (smaller is better)
- replace a character (a - z) with something smaller
  - one char stored as one byte (8 bits)
  - goal: use ≤ 8 bits for most/some chars
    - ↳ 0/1

Huffman goal:

- replace char with bits
- based on frequency of characters

- for every char in file, we know its frequency

$\boxed{\text{ex}}$ Alphabet

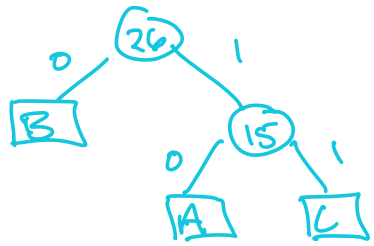
A - 7
B - 11
C - 8

one has 1 bit (0)
two ~~ha~~ have 2 bits (10)
(11)

B should have 1 bit ⇝ most frequent!

A, C can have 2

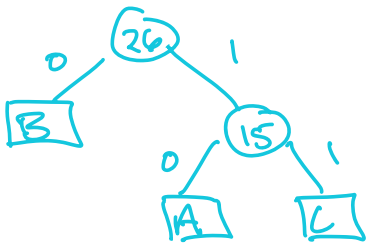
↳ ~~tea~~ create a binary tree
characters are leaves
left = 0
right = 1



B : 0 → code
A : 10
C : 11

Prefix-free code:

- no code is a prefix of another code
- 110 / 111 not valid



ex 1 1 1 0 0 0 1 1 → compressed file
C A B B C

(no ambiguity :))

10:47

3. Greedy Huffman

- given: list of chars with frequencies
- goal: build a tree to rep. an optimal prefix-free code
- the tree can be used to build a new, smaller file

greedy choice: pick the two chars with lowest freqs, build tree from bottom up

High-level Pseudocode

- start with chars/freqs
- remove the two lowest-freq chars (x, y)
- merge x, y together into a subtree
- x.freq + y.freq added back into char/freq list

ex Alphabet: a-f

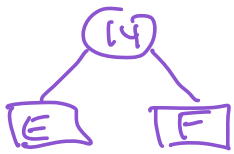
Freqs:

A	B	C	D	E	F
45	13	12	16	~~9~~	~~5~~

! bit !

goze

(1) remove 2 lowest-freq chars
merge into subtree



Put 14 into list

Freq:

A	B	C	D	EF
45	~~13~~	~~12~~	16	14

(2) Remove 2 lowest-freq chars

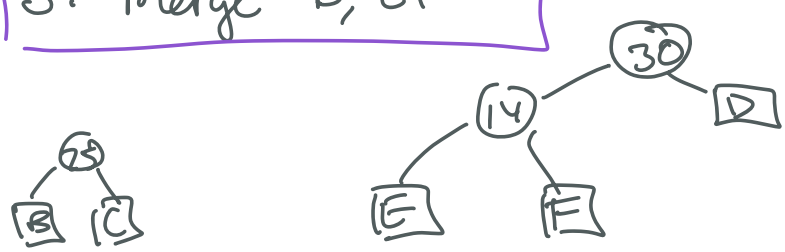
Put 25 into list

A	BC	D	EF
45	25	16	14

3. merge D, EF

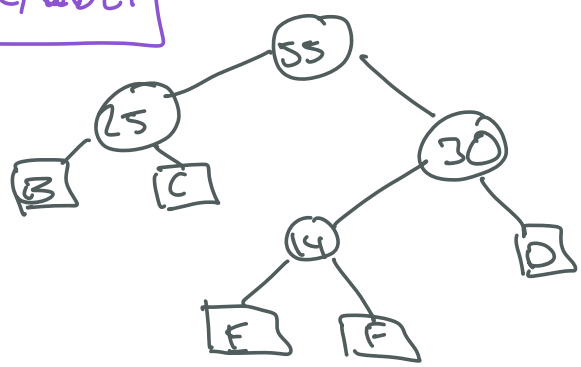
Freqs:

A	BC	DEF
45	25	30

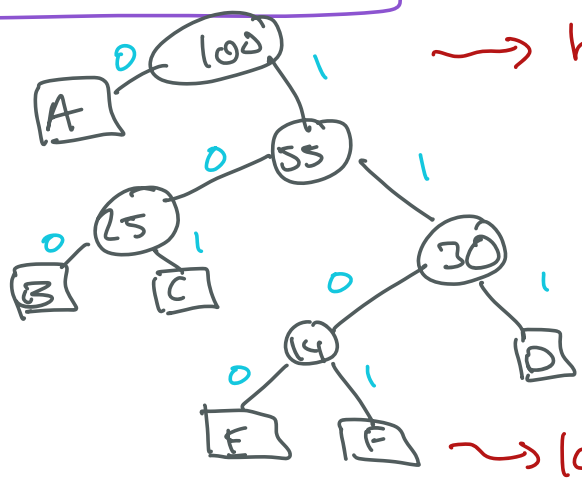


4. merge BC, DEF

Freqs: A 45 | BCDEF 55



5. merge A with rest of tree



→ highest freq (fewest bits)

→ lowest freq (most bits)

ex encode ABC 0100101

ex decode 111100 DB

How much space did we save?

orig: 100 chars, 8 bits each = 800

compressed: $1 \cdot 45 + 3 \cdot 13 + 3 \cdot 12 + 3 \cdot 16 + 4 \cdot 9 + 4 \cdot 5$

A B C D E F

= 224 bits

800 → 224