CS3000
6/2 - Mon.

Admin

- HW4 due thurs 9pm
- APP6 due 11:30 tmw
- thu 6/12 exam #2,
  - ↳ practice problems out 6/5,
    review in rec on 6/10

Agenda

1. File compression
2. Greedy Huffman
3. APP6

0. Greedy Strategy

; can we do better? ;

brute force → (data structure) → D+C → DP → Greedy

What is greedy?

- makes optimal choice at each step
- make it easy to make the optimal choice
- need to prove correctness

What kinds of problems?

- optimization problems
- optimal substructure

Pros/Cons vs. DP?

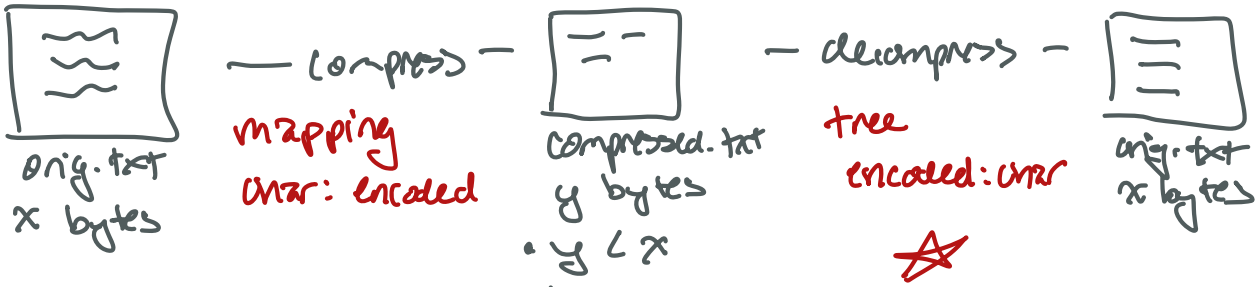
- easier to implement
- careful about correctness
- (usually) faster

Both DP, Greedy:

- need optimal substructure
- make local decisions at each step

1. File compression

- text file of characters
  - ↳ flat file
  - ↳ one byte per character



- y < x
- lossless

(ex) File: LaneyIsAwesome 1000 times

total space: $(5+2+7)\times 1000 = 14{,}000 = 14\text{KB}$

compress: Laney - L
Is - I
Awesome - A

LIA 1000 times

$(1+1+1)\times 1000 = 3{,}000 = 3\text{KB}$

↳ valid! But, what if another word starts with L?
What about optimization?

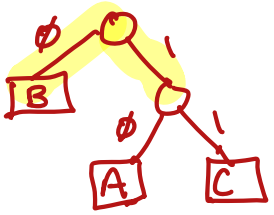
- valid solution: compress
  decompress
  lossless
- optimal solution: minimize space!

Huffman: compress characters, not words
- char = 1 byte (8 bits)
- if we can replace with ≤ 7 bits that's a win!

(ex) File: AABBBBBBC orig. txt
space $9\times 8 = 72$ bits

three chars to encode: A, B, C
- one in one bit ~ most frequent (B)
- two in two bits ~ A, C

Huffman Tree



Building the tree:
- highest freq character = closest to root
- characters are leaves
- 0 on left edges
- 1 on right edges

mapping char → encoded:

B: 0

A: 10

C: 11

decompress (ex)

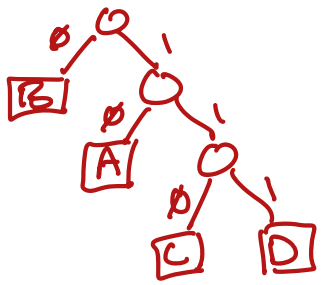
1 0 0 0 1 0

A B B A

Huffman gives a prefix-free code
- no way to encode a char that is a prefix for something else ⇝ 01 bad :(
- only one way to decode any bit sequence

Where would D go?
- assume D.freq = 1

010 111 → BAD!



2. Huffman Algorithm

- goal: build the tree!
- optimization goal: tree creates the smallest encoding

For the algo:
- C, set of chars
- |C| # of characters
- priority queue Q
  - extractmin
  - insert
- every char c has c.freq

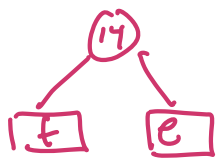
Chars to use:

a	b	c	d	e	f
45	13	12	16	9	5

Qs to consider:
- keys in nodes vs. leaves?
- optimal solution? 900
  - ↳ How many bits do we start with?
- what is greedy choice? ⇝ 2 lowest frequencies
- what is in Q each step?

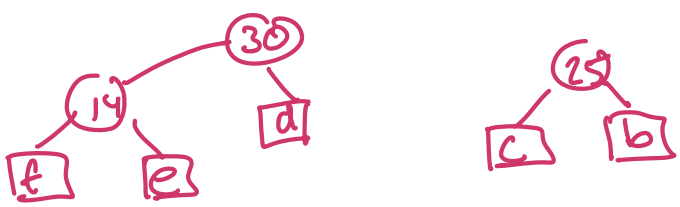
```
HUFFMAN(C)
1  n = |C|
2  Q = C
3  for i = 1 to n − 1
4       allocate a new node z
5       x = EXTRACT-MIN(Q)
6       y = EXTRACT-MIN(Q)
7       z.left = x
8       z.right = y
9       z.freq = x.freq + y.freq
10      INSERT(Q, z)
11 return EXTRACT-MIN(Q)
```

↳ Q: 45 13 12 16 9 5

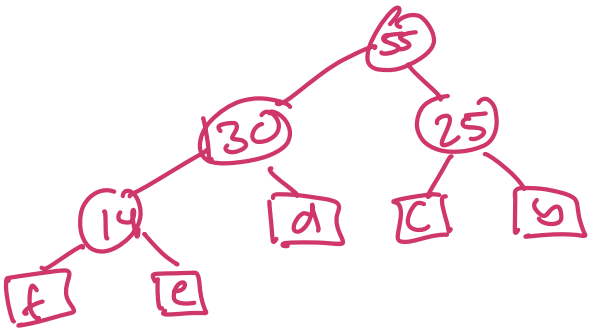


↳ Q: 45 13 12 16 14

↳ Q: 45 25 16 14

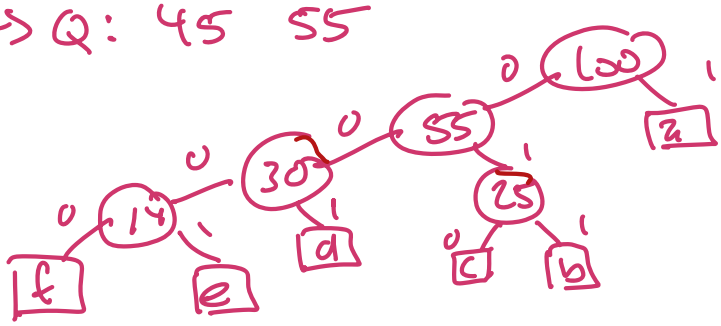


↳ Q: 45 25 30



00101100111
D B D A

↳ Q: 45 55



bits now! a = 1 (1) d = 3 (001)
b = 3 (011) e = 4 (0001)
c = 3 (010) f = 4 (0000)

(compressed bits: 1·45 + 3·13 + 3·12 + 3·16 + 4·9 + 4

= 224 bits !!

not prefix free:
0110 101
0101
0100

00	a e f
01	c b
1	a

APPG