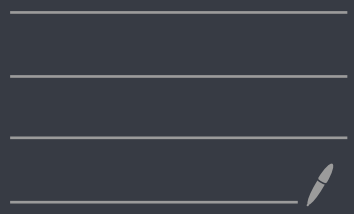
CSCE 310H - Fall 2021

3.0 - Data Structures



$3x + 2y = 35$ → $3(27 - 3y) + 2y = 35$

$x + 3y = 27$

$81 - 7y = 35$

$\frac{7y}{7} = \frac{46}{2}$

$x = 27 - 3y$

$y = 46/7$

3

$$\begin{aligned} 2x + 5y + 2z &= -38 \\ 3x - 2y + 4z &= 17 \\ -6x + y - 7z &= -12 \end{aligned}$$

3

$6z = 4 \quad z = 4/6$

↓ Augmented Matrix

$$\left[\begin{array}{ccc|c} 2 & 5 & 2 & -38 \\ 3 & -2 & 4 & 17 \\ -6 & 1 & -7 & -12 \end{array}\right]$$

want → upper triangular matrix

$$\left[\begin{array}{ccc|c} ? & ? & ? & ? \\ 0 & ? & ? & ? \\ 0 & 0 & ? & ? \end{array}\right]$$

$$\begin{bmatrix} 2 & 5 & 2 & -38 \\ 3 & -2 & 4 & 17 \\ -6 & 1 & -7 & -12 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 2 & 5 & 2 & -38 \\ 0 & -9.5 & 1 & 74 \\ 0 & 16 & -1 & -126 \end{bmatrix}$$

eliminate values in the 1st column below the 1st variable

- what can I multiply row 1 by so that I can subtract it from row 2 to eliminate the 3?

multiply row 1 by -3 and subtract it from row 3 to eliminate

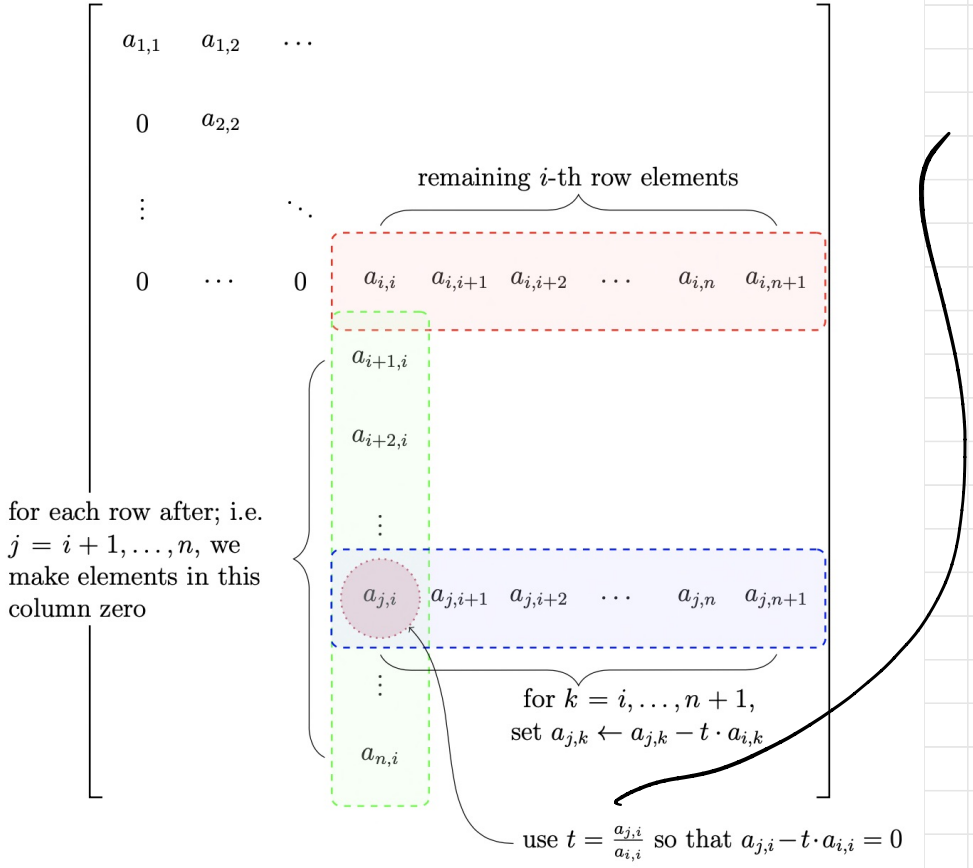
$$\begin{array}{rrrrl} & -6 & & & \\ \hline -6 & 1 & -7 & -12 & \text{(row 3)} \\ -(-6 & -15 & -6 & 114) & (-3 \times \text{row 1}) \\ \hline 0 & 16 & -1 & -126 & \end{array}$$

$$\begin{bmatrix} 2 & 5 & 2 & -38 \\ 0 & -9.5 & 1 & 74 \\ 0 & 16 & -1 & -126 \\ \vdots & & & \end{bmatrix}$$

- what x row 2's -9.5 when subtracted from 16 gives 0?

$$c \cdot -9.5 = 16 \qquad c = -\frac{32}{19}$$

$$\begin{bmatrix} 2 & 5 & 2 & -38 \\ 0 & -9.5 & 1 & 74 \\ 0 & 0 & z & k \end{bmatrix}$$



$$t = \frac{a_{ji}}{a_{ii}}$$

$$a_{ji} - t \cdot a_{ii} = 0$$

$$a_{ji} = 0$$

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ \qquad a_{22}x_2 + a_{23}x_3 = b_2 \\ \qquad\qquad a_{33}x_3 = b_3 \end{array}$$

$$x_3 = b_3 / a_{33}$$

$$x_2 = \frac{b_2 - a_{23} \cdot x_3}{a_{22}}$$

$$x_1 = \frac{b_1 - a_{13}x_3 - a_{12}x_2}{a_{11}}$$

$$x_i = \frac{b_i - \overbrace{a_{i,i+1}x_{i+1} - \cdots - a_{in}x_n}^{n-i \text{ Things}}}{a_{ii}}$$

Algorithm 21: (Better) Gaussian Elimination

INPUT : An $n \times n$ matrix A and a $1 \times n$ vector b

OUTPUT : An augmented, upper triangluar matrix A' preserving the solution to $A \cdot \vec{x} = \vec{b}$

```
1  A' ← A|b⃗ //augment
2  FOR i = 1, ... n − 1 DO
3      pivotrow ← i
4      FOR j = (i + 1), ..., n DO
5          IF |A'[j, i]| > |A'[pivotrow, i]| THEN
6              pivotrow ← j
7      FOR k = i, ..., (n + 1) DO
8          swap(A'[i, k], A'[pivotrow, k])
9      FOR j = (i + 1), ..., n DO
10         t ← A'[j,i] / A'[i,i]
11         FOR k = i, ..., (n + 1) DO
12             A'[j, k] ← A'[j, k] − A'[i, k] · t
13 output A'
```

$n \times (n+1)$

for rows $1 \ldots n-1$

finds the "best" pivot row

absolute value

swapping pivot row / ith row

for each row below ith row...

"factor" to multiply row i by to subtract it from row j

subtracts $t \cdot$ row i from row j

$O(n^3)$

Back Solve: given an $n \times (n+1)$ upper triangular matrix A

```
for i = n ... 1
    t ← A[i, n+1]        // b_i
    for j = (i+1) ... n
        t ← t − x_j · A[i, j]
    x_i ← t / A[i, i]
output (x_1 ... x_n)
```

scalar on the RHS

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} 1$$

... solve

$O(n^2)$

Problems:

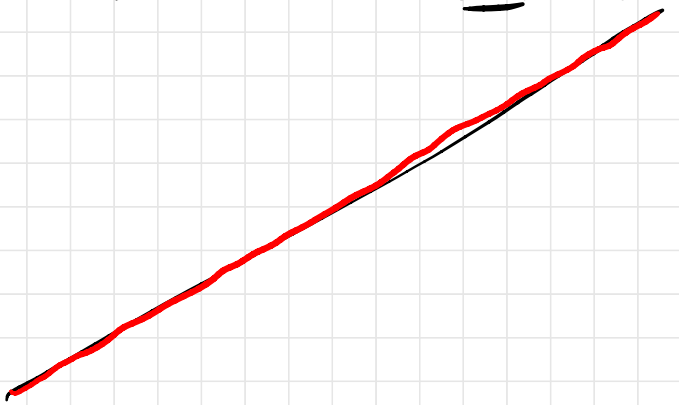
$$2x + 3y = 14$$
$$4x + 6y = 28$$

↓

$$2x + 3y = 14$$
$$0 + 0 = 0$$

indeterminant system.

- infinite # of solutions
- last row is <u>all</u> zeros



$$x + 7y = 8$$

$$-2x \quad 2x + 14y = 20$$

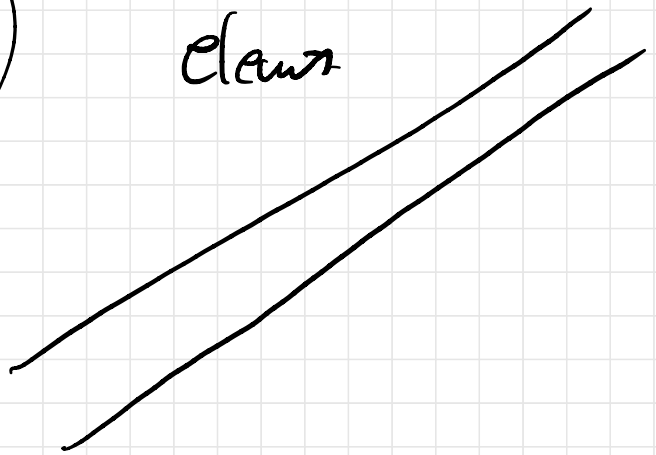
Inconsistent system
- no solutions exist

- $0 = 1$

$$\begin{matrix} 1 & 7 & 8 \\ 0 & 0 & 4 \end{matrix}$$

$$0z = 4$$

- last row is all zeros except the last column



$$0 = 4$$

Huffman Coding

$n \to \infty \quad p^{n/2} \to 0$

$\to 010$ (codeword)

$p^2 < p$

0 → 00000

1 → 11111

encode

$0 \to 000 \to$ [noisy channel] $\to 010 \to 0$

Majority vote

- detect an error

a bit is flipped w/ prob. $p < 1/2$

n bits + log(n) bits of redundant info

detect errors and/or

correct errors

1 more bit: 0 if even # of 1s

0000

1 if odd # of 1s

$A = 65 =$ 01000001

all codewords length 8

$Z = 91 =$?

8 bits

Variable length encoding:

A B C D

$\{0, 01, 101, 010\}$

4 code words

not prefix-free

- some codewords appear as prefixes of others

receive → 010 → Decode → D

B A

B, A

(4) (3)

0 is a prefix 01

01 (B) « 010 (D)

Want: a prefix free variable length encoding

A B C D

{10, 010, 110, 0110}

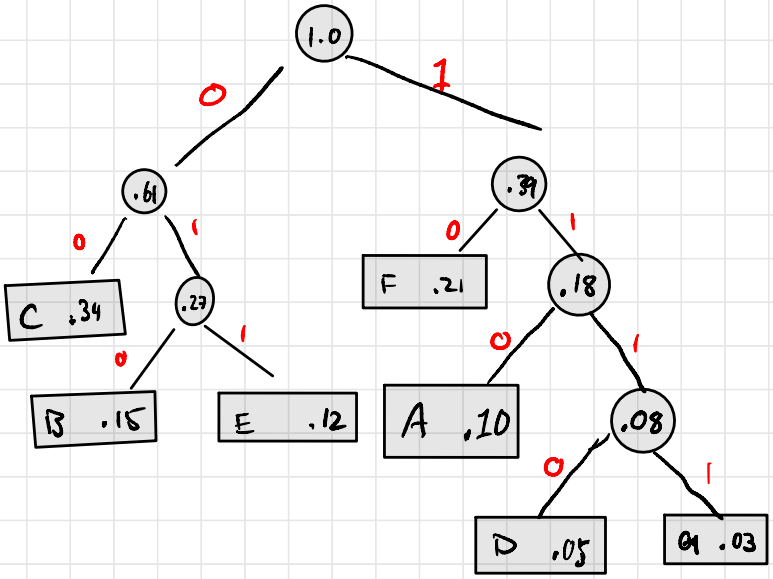
10: most common

0110: least common

Average Codeword length:

suppose a symbol has a "frequency"

$\sum_{\text{all codewords}}$ freq × code word length = compression ratio



Symbol	codeword	length
A	110	3
B	010	3
C	00	2
D	1110	4
E	011	3
F	10	2
G	1111	4

Symbol	codeword	length			Weighted codeword length
A	110	3	x	.10	.3
B	010	3	x	.15	.45
C	00	2	x	.34	.68
D	1110	4	x	.05	?
E	011	3	x	.12	
F	10	2	x	.21	
G	1111	4	x	.03	

sum + 2.53

Avg codeword length 2.53 bits

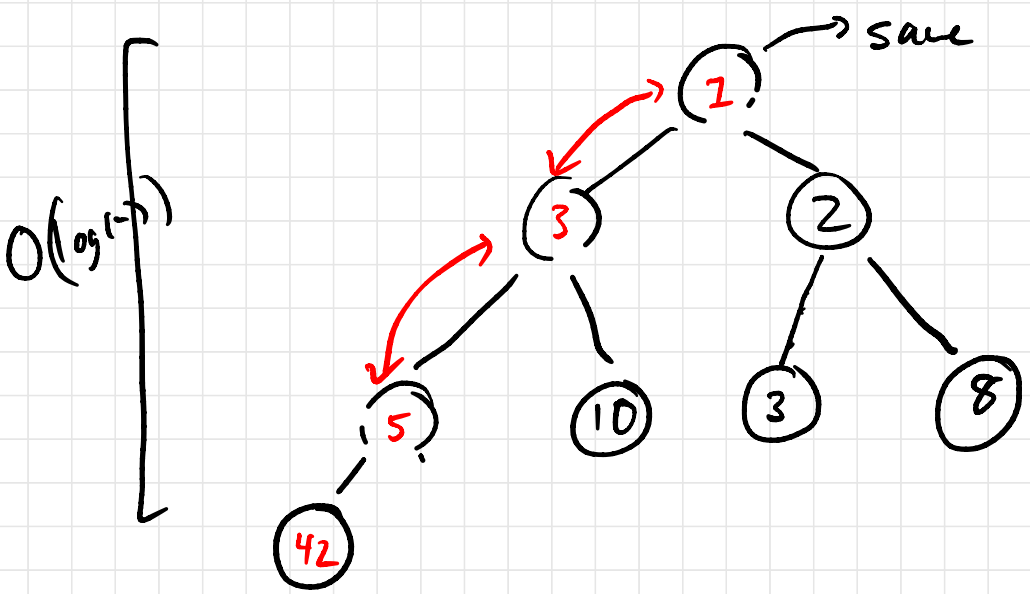
# of bits under ASCII: 8

#bits in a fixed length encode:

$$\lceil \log_2(7) \rceil = 3 \qquad \frac{2.53}{3} = 85\%$$

$$\frac{3 - 2.53}{3} \approx 15\% \text{ compression ratio}$$

- nodes: { letter or no letter (internal); frequency; parent/left/right }
- table/map of encodings
- "walking" algo
- heap: heapq

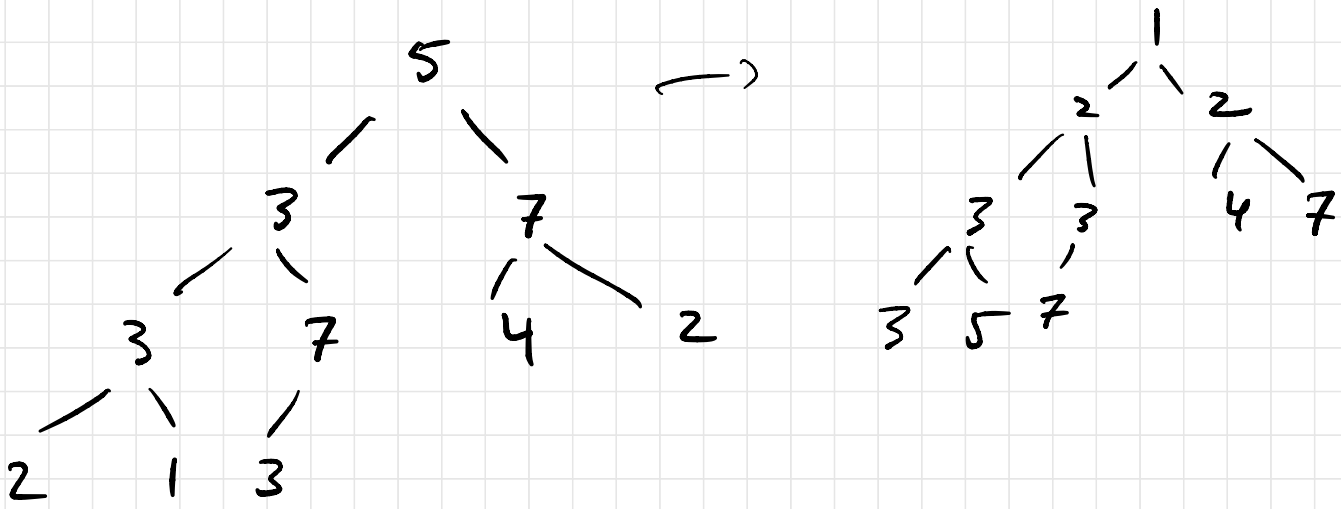


insert: 5

heapify or fix

insert: -1

getMin()


5
3
7
3
7
4
2
2
1
3
1
2
2
3
3
4
7
3
5
7

Input: A collection of letters $\Sigma = \{x_1, \dots x_n\}$, with frequencies $freq(x)$

Output: A Huffman tree

//init:

- create n single node trees, each with:
  1) symbol x_i
  2) $wt = freq(x)$
- place all trees into a min-heap H using the wt.

for $j = 1 \dots n-1$

left ← H.pop()

right ← H.pop()

u ← new node with wt = left.wt + right.wt

u.left = left, u.right = right.

H.insert(u)

return H.pop()

Complexity:

Elem. op: heap push/pop

$$\underbrace{2 \cdot O(\log(n))}_{\text{pop} \times 2} + \underbrace{1 \cdot O(\log(n))}_{\text{push}} = \underbrace{O(\log(n))}_{1 \text{ iteration.}}$$

$n-1$ iterations $\Rightarrow O(n \log(n))$