CS 4800: Algorithms & Data

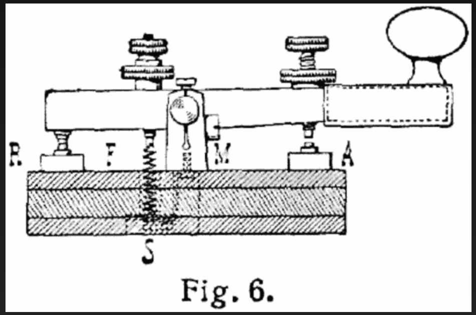
Lecture 13

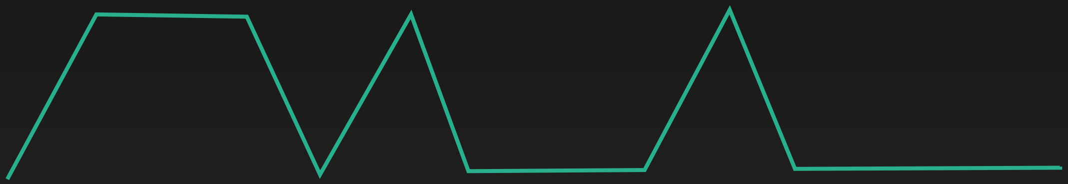
February 23, 2018

Huffman codes

Information transmission

Once upon a time, before Internet and emails,





Texts are transmitted as electrical pulses and silence in between

Long pulses (1) and short pulses (0)

Length of encoding

- Letter c occurs f_c times and its encoding is of length l_c bits
- Encoding length$= \sum_c \ f_c l_c$
- Given a text consisting of n distinct letters, find minimum length encoding

Morse code

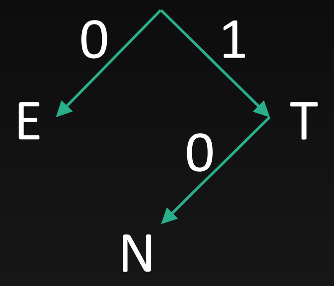
- Encode letters as sequences of dots & dashes (0/1)

Letter	Code
A	01
E	0
I	00
N	10
T	1

What does 01 mean? ET or A?

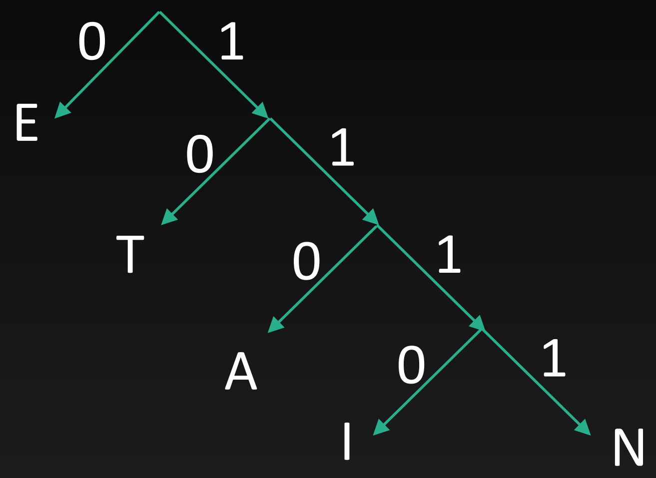
Prefix-free codes

- Problem with Morse code: some encoding is prefix of another
- Prefix-free code: for any two letters $x \neq y$, code(x) is not a prefix of code(y)



Encoding/decoding prefix-free codes

- Text: EATIN
- Encoding:
  - 01101011101111
- Decoding:
  - Start at root
  - Go down until reaching a leaf
    - get a letter
  - Restart from the root



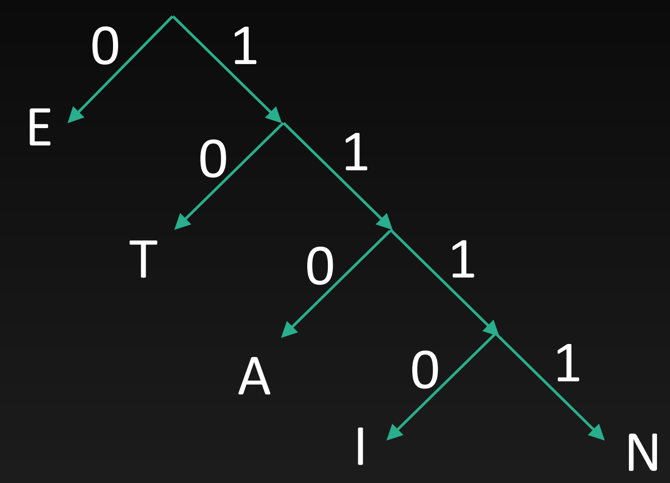
A text for compression

This sentence contains three a’s, three c’s, two d’s, twenty-six e’s, five f’s, three g’s, eight h’s, thirteen i’s, two l’s, sixteen n’s, nine o’s, six r’s, twenty-seven s’s, twenty-two t’s, two u’s, five v’s, eight w’s, four x’s, five y’s, and only one z

-Lee Sallows

Prefix-free code to tree

Letter	Code
A	110
E	0
I	1110
N	1111
T	10

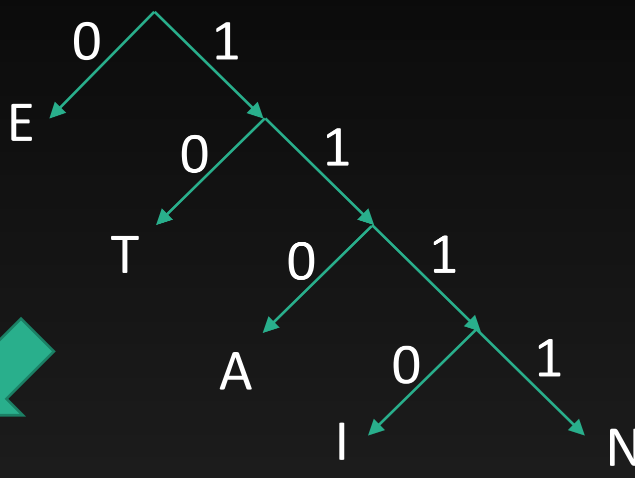


Build tree recursively

- Start with root
- All letters start with 0 go to the left subtree
- All letters start with 1 go to the right subtree
- Recursively build two subtrees

Binary tree to code

- Binary tree with n labeled leaves
- Left branch → 0, right branch → 1
- Encoding of letter c is the path from root to leaf c



Letter	Code
A	110
E	0
I	1110
N	1111
T	10

Which trees give optimal codes?

- Minimize encoding length= $\sum_c f_c l_c$

Optimal tree is full

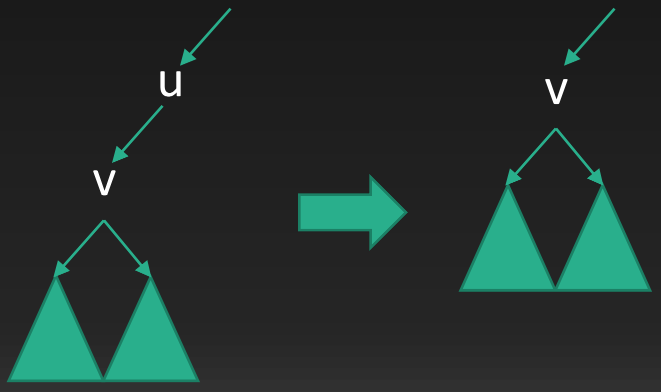
Claim. In optimal tree, non-leaf nodes have 2 children.

Proof. Let T be an optimal tree. Suppose T contains u with one child v.

Remove u and move v into u's location.

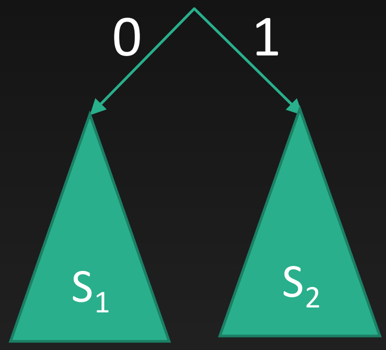
No encoding gets longer.

Encodings of leaves in subtree rooted at v get shorter.



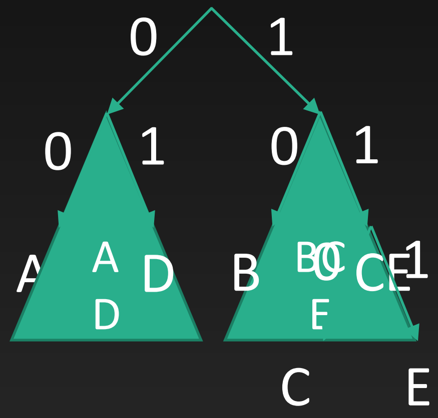
First attempt

- Split alphabet S into S_1, S_2 such that total frequency of S_1 and S_2 are as close as possible.
- Recurse on S_1 and S_2
- Shannon-Fano codes

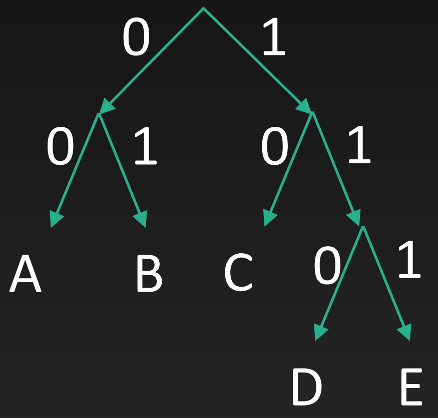


A counter-example

Letter	A	B	C	D	E
Frequency	32	25	20	18	5



Total freq each side: 50



Total cost: 225

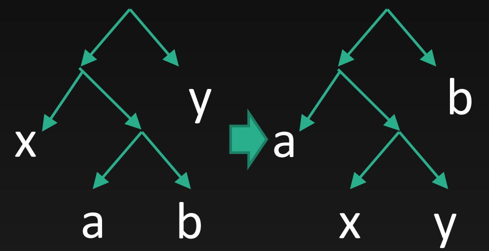
Total cost: 223

Exchange argument

Claim. Let x and y be 2 least frequent characters. There is an optimal code where x and y are siblings and have the max depth of any leaf.

Proof. Let T be optimal tree with max depth d.

T is full so there are 2 sibling leaves at depth d.



Suppose they are a and b, not x and y.

Swap a and x.

Depth of x increases by D, depth of a decreases by same D.

$$New\ cost\ =\ old\ cost\ -\ (f_a - f_x)D$$

x,y are least frequent so $f_a \geq f_x$. Thus, $New\ cost\ \leq\ old\ cost$

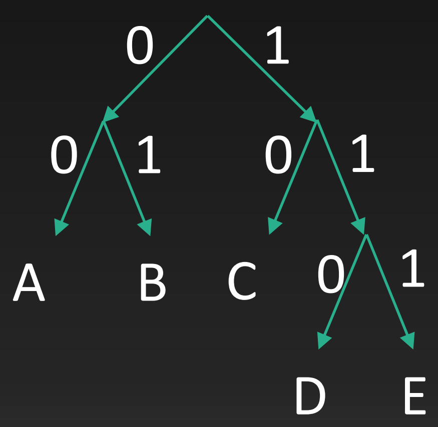
Similarly swapping b and y also decreases cost.

Huffman codes

- Find 2 least frequent letters
- Merge them into a new letter
- Repeat

Example

Letter	A	B	C	D	E
Frequency	32	25	20	18	5



New letter DE: 23

New letter CDE: 43

New letter AB: 57

Total cost: 223

Huffman codes are optimal

- Induction via optimal substructure
- Base case: $n=1$ or $n=2$, optimality is trivial
- Inductive case: assume Huffman codes are optimal for $n<k$, will show it is optimal for $n=k$

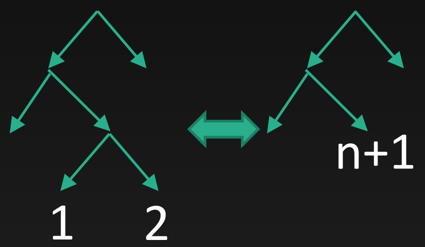
Proof

Let $f_1, f_2, \dots, f_n$ be letter frequencies.

Without loss of generality, assume f_1, f_2 are the smallest.

By lemma, some optimal tree has 1 and 2 as siblings.

Thus, focus only trees with 1 and 2 as siblings.



Let $f_{n+1} = f_1 + f_2$.

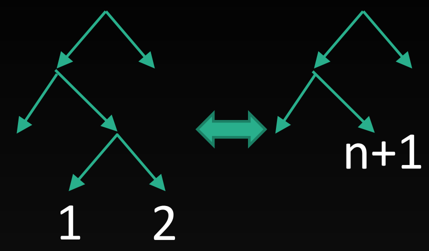
Let T' be Huffman code tree for $f_3, \dots, f_n, f_{n+1}$.

By induction, T' is optimal.

To obtain T, replace the leaf labeled n+1 with an internal node with two children, 1 and 2.

Need to show T is optimal for frequencies $f_1, f_2, \dots, f_n$.

Proof



Let depth(i) = depth of leaf i in either T or T’

$$
\begin{aligned}
cost(T) &= \textstyle\sum_{i=1}^{n} f_i \cdot depth(i) \\
&= \textstyle\sum_{i=3}^{n} f_i \cdot depth(i) + f_1 \cdot depth(1) + f_2 \cdot depth(2) \\
&= \textstyle\sum_{i=3}^{n} f_i \cdot depth(i) + (f_1 + f_2) \cdot (1 + depth(n+1)) \\
&= \textstyle\sum_{i=3}^{n} f_i \cdot depth(i) + (f_1 + f_2) \cdot depth(n+1) + f_1 + f_2 \\
&= \textstyle\sum_{i=3}^{n} f_i \cdot depth(i) + f_{n+1} \cdot depth(n+1) + f_1 + f_2 \\
&= cost(T') + f_1 + f_2
\end{aligned}
$$

$f_1 + f_2$ is fixed so minimizing cost(T) is equivalent to minimizing cost(T’).

T’ is optimal so T is also optimal.

Food for thought

- Take a large text file
- Encode using Huffman code and e.g. zip format
- Compare the sizes
- Usually zip is smaller, why can this happen given that Huffman codes are optimal?