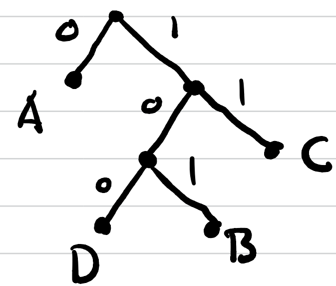
Recap: data compression

Text consisting of n letters $1, \dots, n$ with frequencies $f_1, \dots, f_n$
Encode letters with prefix-free code

Prefix-free code

	A	B	C	D
	0	101	11	100
f:	0.4	0.2	0.3	0.1

Binary tree



Given symbol frequencies $f_1, \dots, f_n$, find the best prefix-free code

$$\text{cost(code)} = \sum_{i=1}^{n} f_i \cdot (\#\text{ bits for letter } i \text{ in code})$$

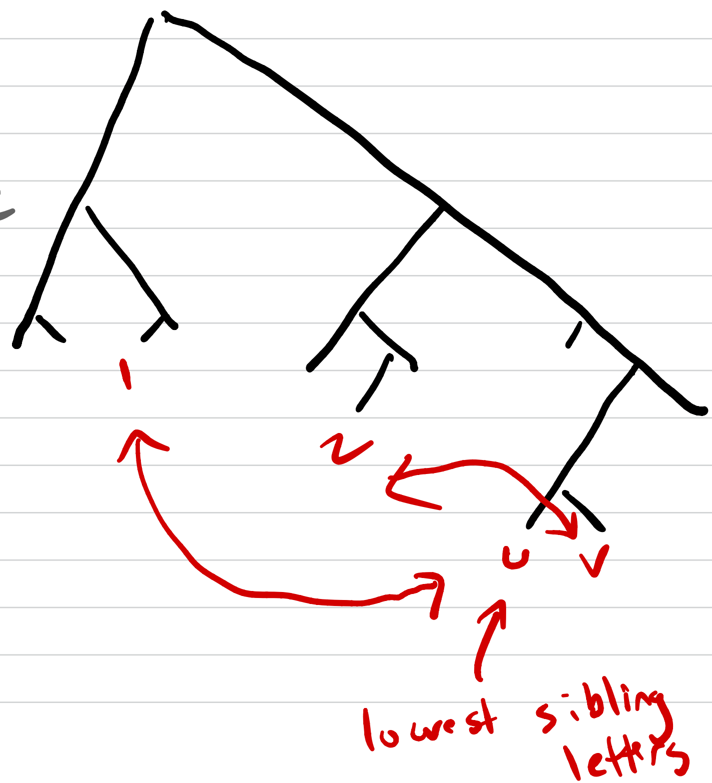
$$\text{cost(tree)} = \sum_{i=1}^{n} f_i \cdot (\text{depth of symbol } i \text{ in tree})$$

Claim: There is an optimal prefix-free tree where the two lowest frequency letters are siblings.

$f_1 \le f_2 \le \dots$

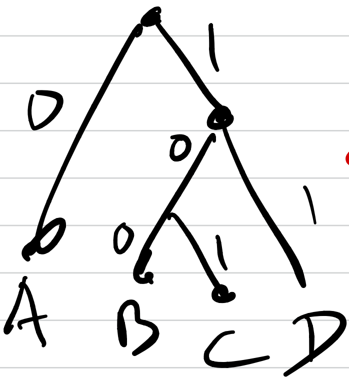
Pf:

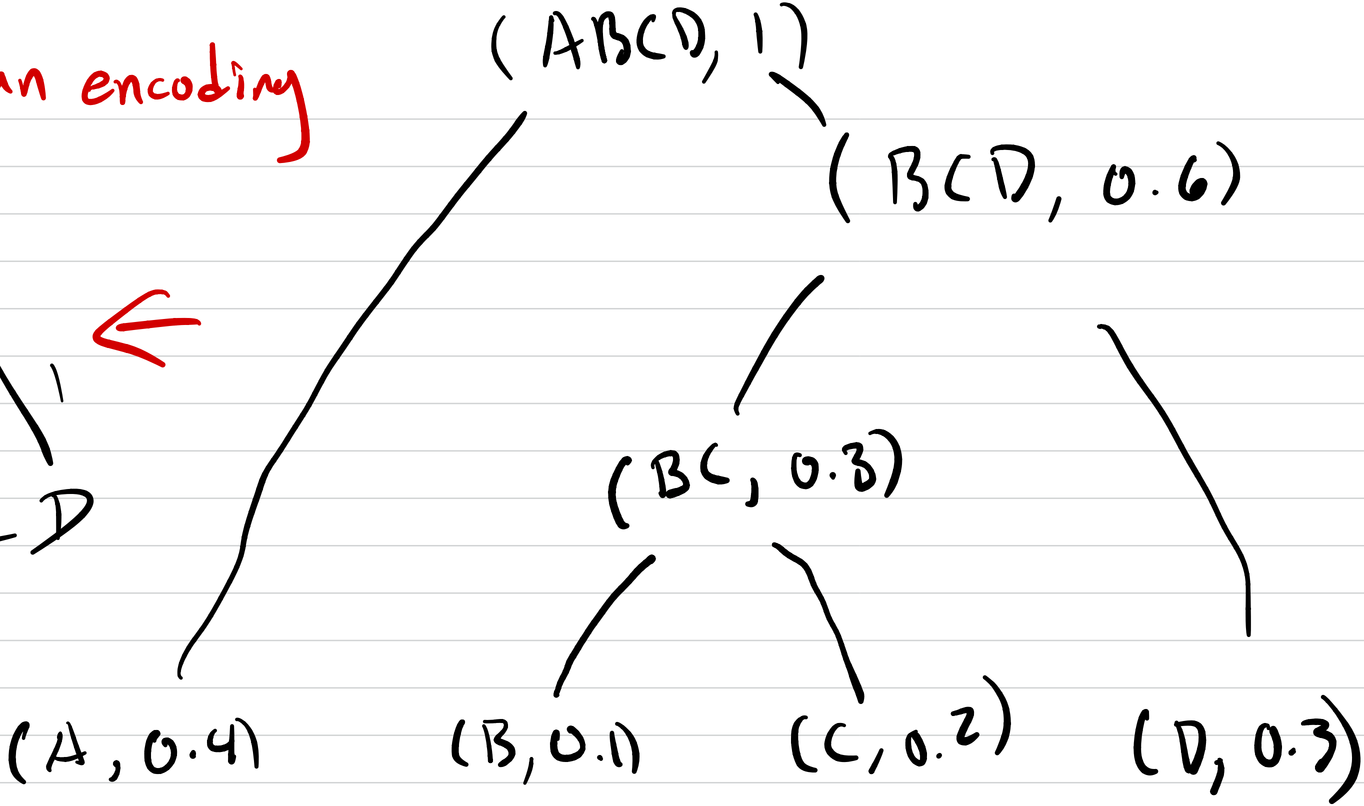
WLOG a full binary tree



Swapping 1 & 2 with u and v cannot increase cost of tree

Huffman encoding

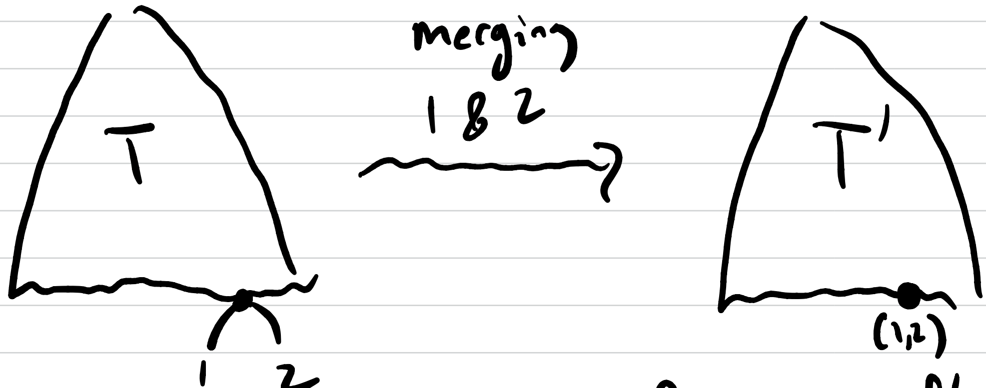




Runtime: $O(n \log n)$

Merging nodes

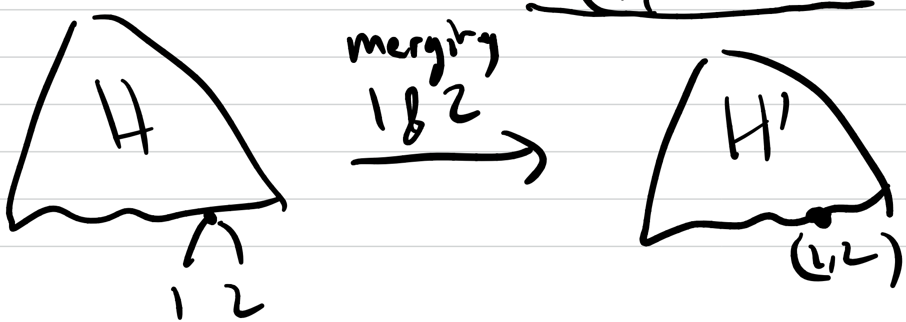
Suppose frequencies $f = (f_1, \ldots, f_n)$, $f_1 \le f_2 \le \cdots \le f_n$



frequencies $f' = (f_1 + f_2, f_3, \ldots, f_n)$

$$\text{cost}(T) = \text{cost}(T') + \underline{(f_1 + f_2)}$$

Huffman code for f



Huffman code for f'

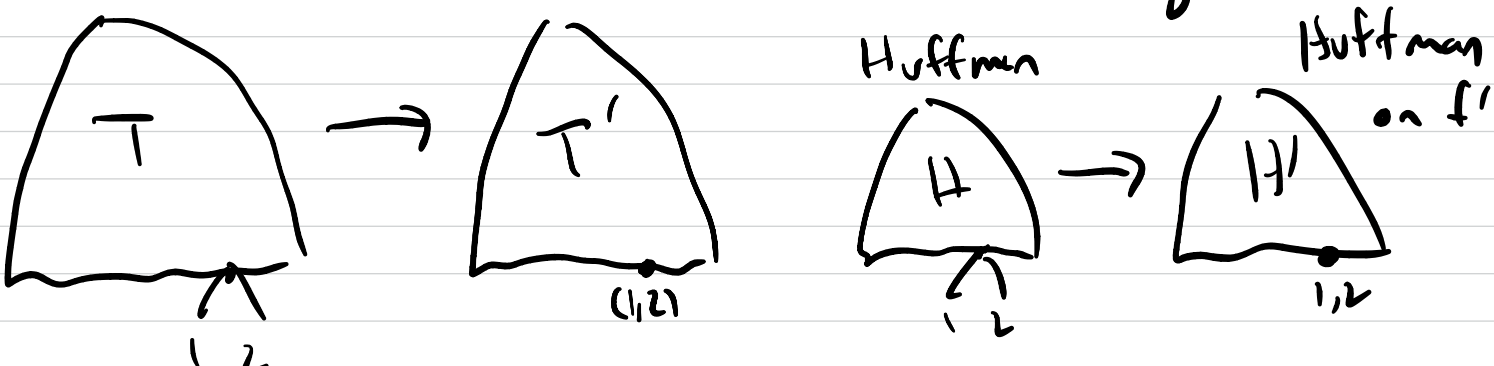
Claim: Huffman coding gives an optimal prefix-free tree

Pf: By induction on n.

Base case: $n=2$ ✓

Induction step: Assume $f_1 \le f_2 \le \cdots$

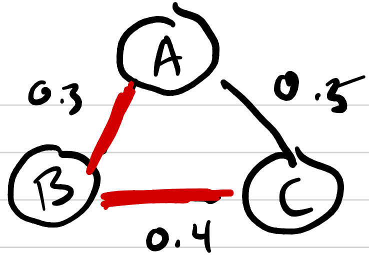
Let T be an optimal prefix-free tree w/ f_1 & f_2 siblings



$$\text{cost}(T) = \text{cost}(T') + (f_1+f_2) \qquad \text{cost}(H) = \text{cost}(H') + (f_1+f_2)$$

$$\text{cost}(H') \le \text{cost}(T') \Rightarrow \text{cost}(H) \le \text{cost}(T) \qquad \square$$

T: –

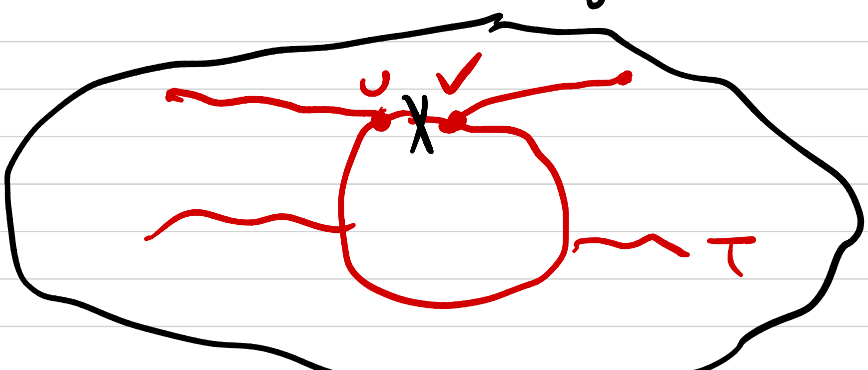


$G = (V, E)$

edge weights w_e (positive)

Goal: Find the subset of edges T with the minimum total weight $\sum_{e \in T} w_e$ which keeps G connected

Claim: This T has no cycles. This T is a tree.



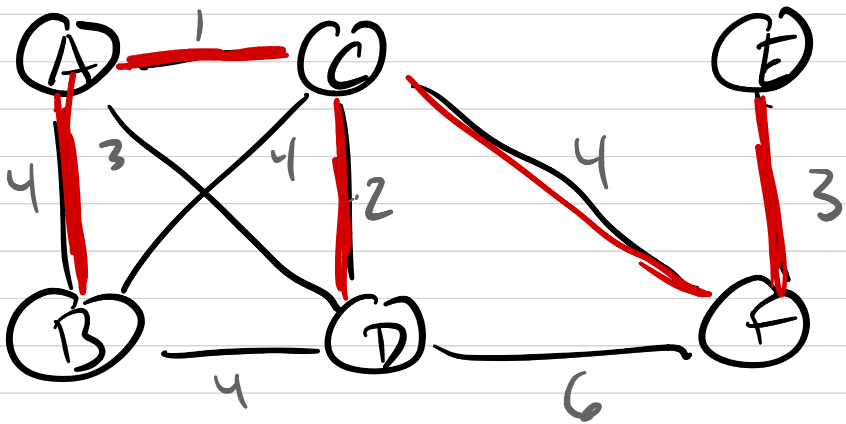
T is a **Minimum Spanning Tree (MST)**

A tree is any acyclic connected graph

Equivalently:
- G is connected
- G has $n-1$ edges

Kruskal's algorithm for MST

Repeatedly add the next lightest edge that doesn't produce a cycle

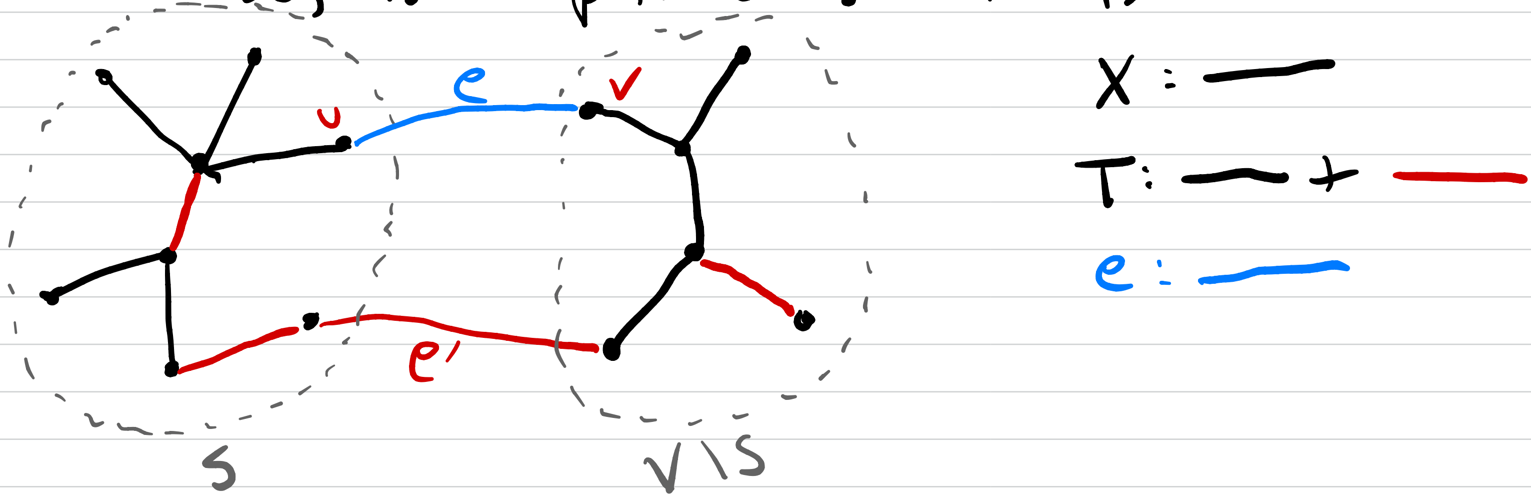


T: ——

Two questions: 1. Why does this work?,
2. How do you implement it?

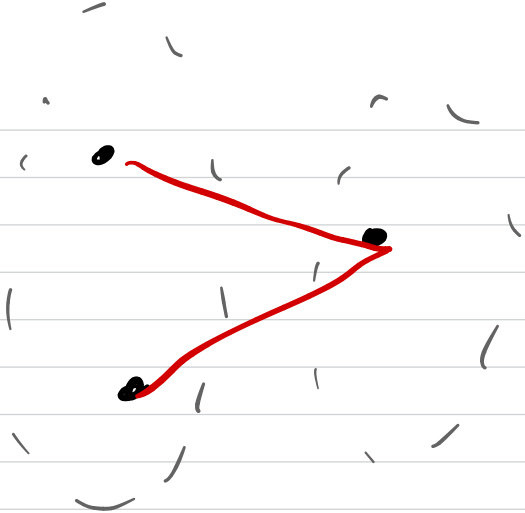
The cut Property

Suppose $X \subseteq E$ is a part of MST for G.
Let $S \subseteq V$ s.t. X has no edge from S to $V \setminus S$.
Let e be the lightest edge from S to $V \setminus S$.
Then $X \cup \{e\}$ is a part of some MST.



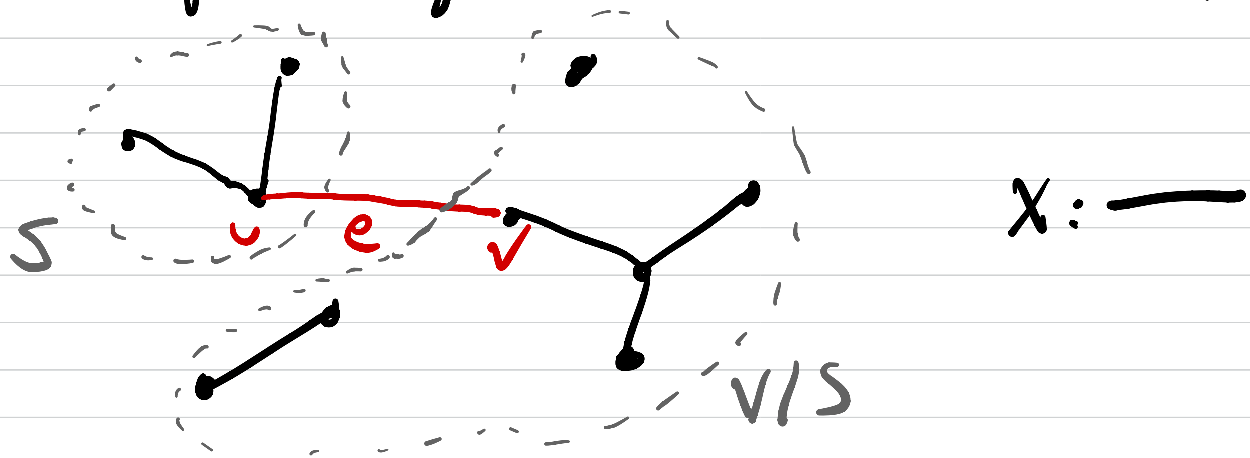
Pf: Suppose e is not in T.
There is an edge e' on path from u to v which goes from S to $V \setminus S$.
Remove e', add e.

$T \cup \{e\} \setminus e'$ is also MST, contains $X \cup \{e\}$. $\square$



Correctness of Kruskal

Add the next lightest edge that doesn't produce a cycle



If X is part of an MST, then so is $X \cup \{e\}$
by cut property

Claim: At all time steps,
Kruskal's set of edges X is part of some MST

Pf: By induction.

Implementing Kruskal

for each edge (u,v), check if u's connected component $= v$'s CC

Disjoint set data structure:
(union-find data structure)

- makeset(x)
  (makes a singleton set containing x)
- find(x) (which set does x belong to?)
- union(x, y)
  (merges sets containing x and y)

Kruskal(G, w)

```
for all u ∈ V, makeset(u)
X = { }
sort edges in E by weights

for all edges (u,v) ∈ E in increasing order of weight
    if find(u) ≠ find(v):
        add (u,v) to X
        union(u,v)
return X
```

Runtime of Kruskal's

$n = |V|$, $m = |E|$

n makeset, $2m$ finds, $n-1$ unions

+ sorting edges $O(m \log m) = O(m \log n^2)$

$= O(m \log n)$

Union find

makeset $O(1)$, find, unions $O(\log(n))$ time

total: $O((m+n) \log n)$

$= O(m \log n)$ (for reasonable graphs)

Meta-algorithm

$X = \{\}$

repeat until $|X| = |V| - 1$

pick $S \subseteq V$ s.t. X has no edges from S to V/S

let $e \in E$ be the lightest edge from S to $V \setminus S$

$X = X \cup \{e\}$