Greedy Algorithms (continued)

Last Time

- The student's Problem.
- MST
- The Cut Property
- Kruskal's Algorithm

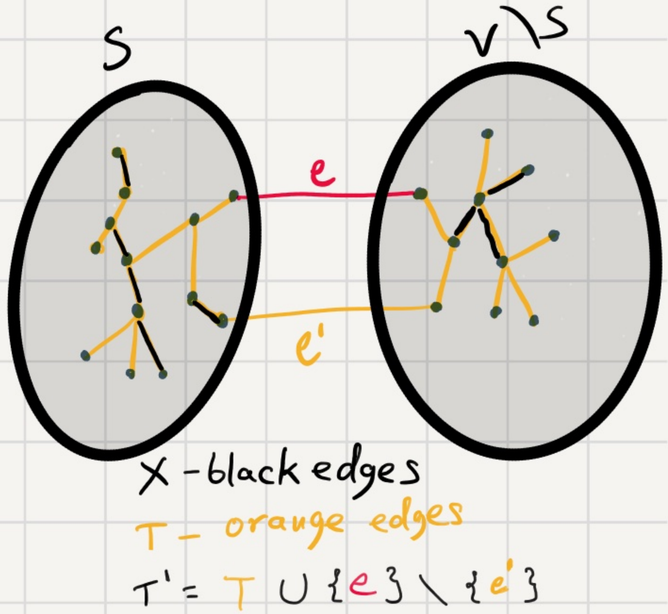
Today

- Finish Kruskal.
- Prim's Algorithm.
- Compression - Huffman Coding.
- Set Cover.

Recap: **MST:** Given an undirected graph $G=(V,E)$ with weights $w: E \to \mathbb{N}$, find a tree T that connects all vertices in V and minimizes $\sum_{e \in E(T)} w(e)$.

"The Cut Property":

- The cheapest edge crossing a cut $(S, V \setminus S)$ appears in some MST.
- More refined: Let $X \subseteq E$ and an MST T such that $X \subseteq E(T)$.
  Suppose that X doesn't cross a cut $(S, V \setminus S)$.
  If e is the cheapest edge crossing $(S, V \setminus S)$ then $X \cup \{e\}$ is contained in some MST T'.



Meta Algorithm

- $X \leftarrow \emptyset$
- For $i = 1, \dots, |V|-1$:
  - Find a cut $(S, V \setminus S)$ s.t. X doesn't cross it.
  - Add cheapest edge in the cut to X.

Kruskal(G, w):

1. $X \leftarrow \emptyset$,
2. Sort edges by their weight. $O(|E| \cdot \log |E|)$.
2.1 For every $v \in V$ makeset($\{v\}$).
3. For all edges $e \in E$, ordered by weight: $e = (u, v)$

   If adding e to X creates a cycle, then skip

   ↳ Else, $X \leftarrow X \cup \{e\}$. union$(u, v)$.

find(u) = find(v)

Last Time: We saw the PoC for Kruskal, based on the Cut Property.

Implementation / Runtime? Naive Implementation $O(|E| \log |E| + |E||V|)$.

Runtime:

n	makesets	$O(1)$
m	finds	$O(\log v)$
$n-1$	unions	$O(\log v$

$\Rightarrow$ Total runtime: $O(|E| \log |E|)$.

Recall

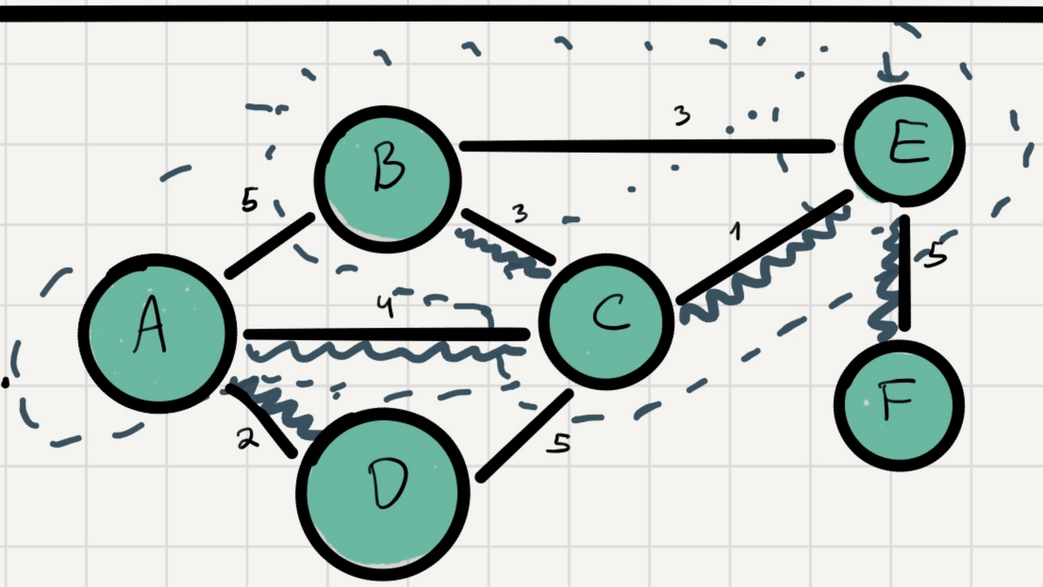
Meta Algorithm

- $X \leftarrow \emptyset$
- For $i = 1, \dots, |V|-1$:
  - Find a cut $(S, V \setminus S)$ s.t. X doesn't cross it.
  - Add cheapest edge in the cut to X.

Prim's Algorithm:

Idea: At each point in time, X would be a subtree.

- At each step, pick the minimal edge between $S = \{v : X \text{ touches } v\}$ and its complement.



- How to implement? similar to Dijkstra's Alg.

Maintain $\forall v \notin S$

$\text{prev}(v) = \underset{u \in S}{\text{argmin}}\ w(u,v)$.

$\text{cost}(v) = \min_{u \in S} w(u,v)$.

Pick $v \notin S$ with smallest cost.

add $(v, \text{prev}(v))$ to X.

Data Compression

We have an alphabet of 32 characters & frequencies $f_1, \dots, f_{32}$.
Say we have a text with N characters and we want to encode it as efficiently as possible in binary.

Naive: Every character → 5 bits overall: $5N$ bits

Example:

Freq		Encoding 1	cost	Encoding 2	Encoding 3	cost
0.4	A	00	$0.4N \cdot 2$	0	0	$0.4N \cdot 1$
0.3	B	01	$0.3N \cdot 2$	1	10	$0.3N \cdot 2$
0.2	C	10	$0.2N \cdot 2$	00	110	$0.2N \cdot 3$
0.1	D	11	$0.1N \cdot 2$	01	111	$0.1N \cdot 3$
			$2N$.			$1.9N!$

00|10|11|10
A C D C.

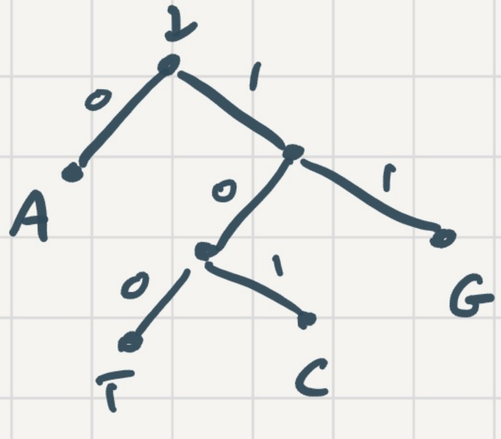
0|110|111|110
A C D C

Prefix Free Codes & Trees

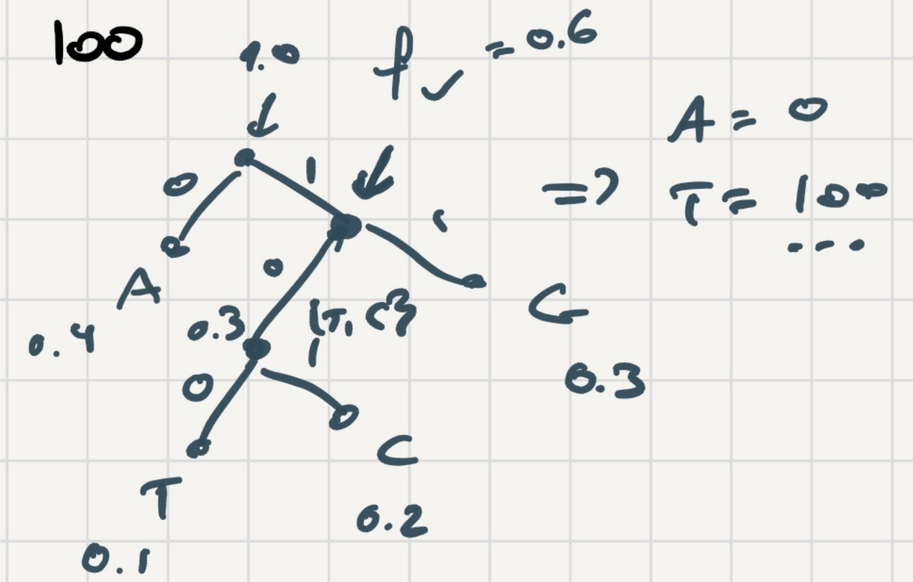
Any prefix-free code corresponds to a binary tree & vice versa.

	0.4	0.2	0.3	0.1
Example:	A	C	G	T
	0	101	11	100

Prefix-free encoding $\Rightarrow$



tree.



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

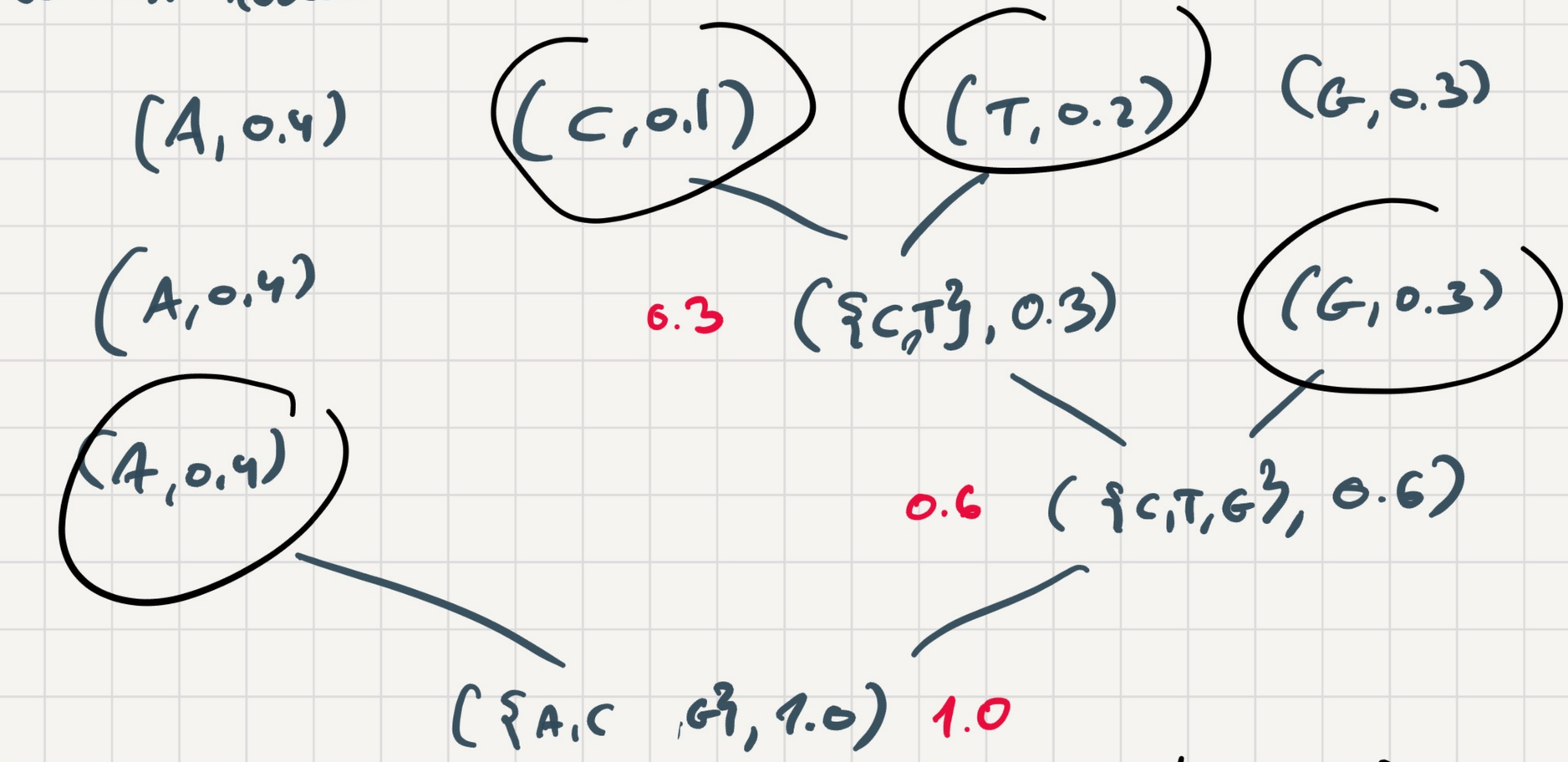
$$\text{cost(tree)} = \sum_{i=1}^{n} f_i \cdot (\text{depth of symbol } i \text{ in the tree}).$$

Cost: Associate with every internal node v, $f_v = \sum_{u \text{ descendants of } v} f_u$.

$$\text{cost of prefix tree} = \sum_{v \text{ internal node}} f_v$$

Greedy: which nodes to merge first.

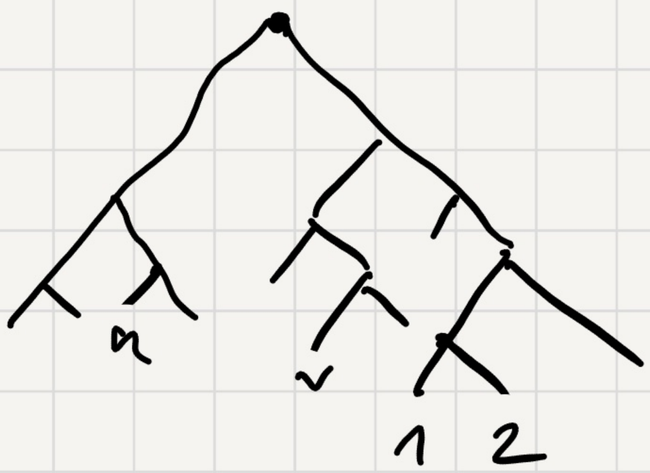
Example :



Huffman: There's an optimal tr where the two lowest freq are siblings.

Proof:

WLOG an optimal tree is a full-binary tree.



$f_1 \leq f_2 \leq f_3 \cdots \leq f_n$.

swap(1, u)
swap(2, v)

} either improves value or maintains the same value.

Claim: Huffman's strategy gives an optimal prefix-free tree.

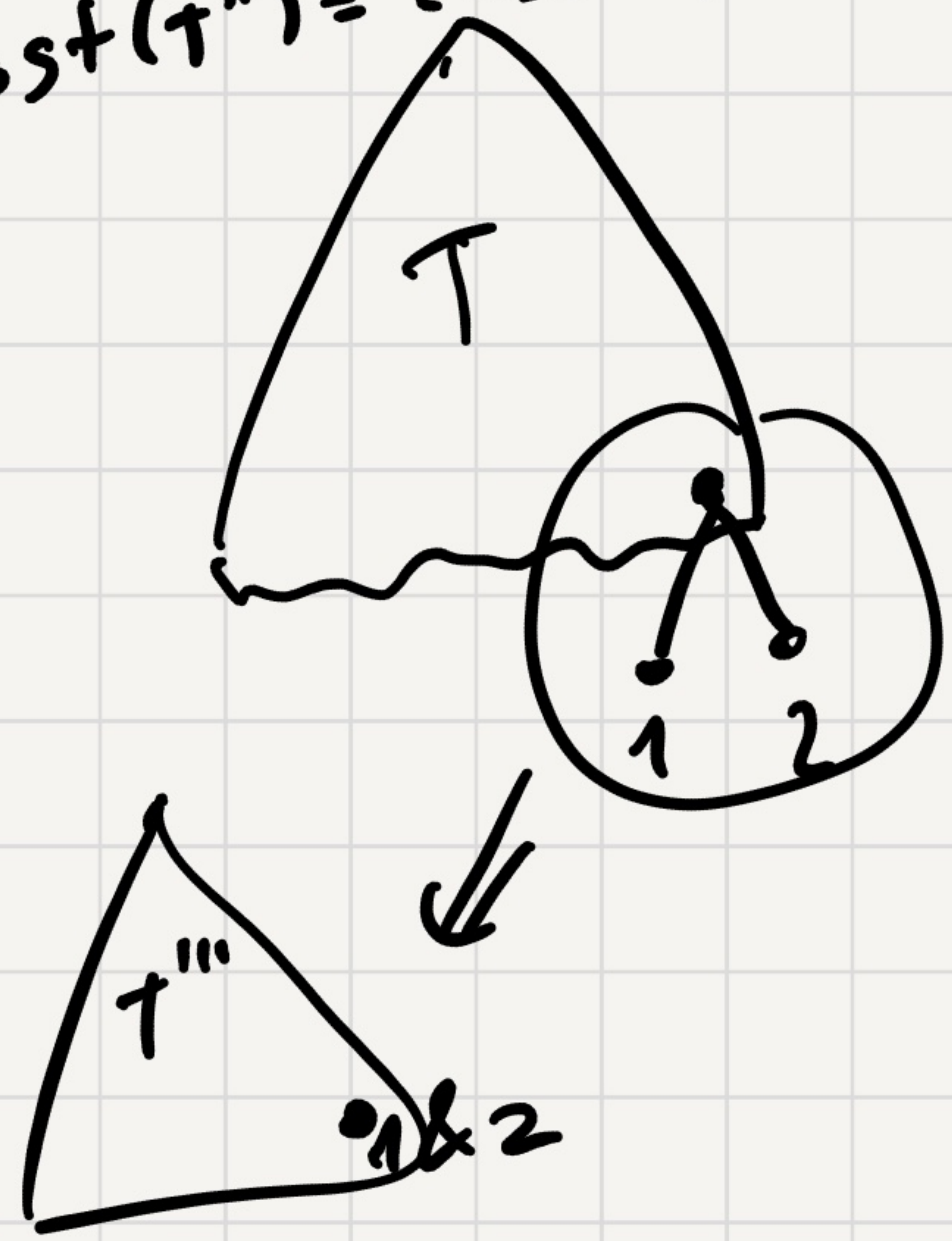
Proof: By induction on n.

Base case: $n=1$, $n=2$ ✓

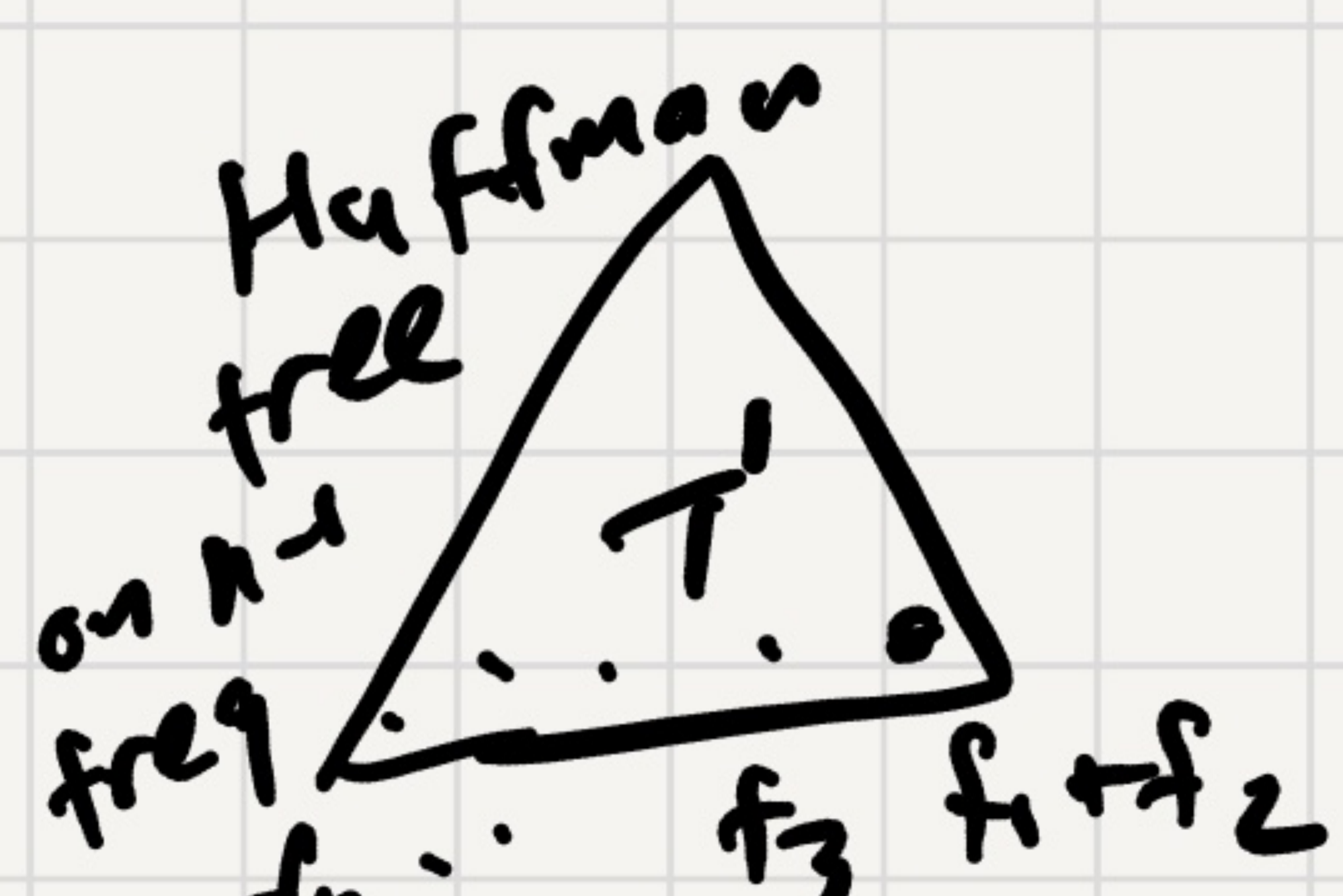
Induction step: Let $\hat{T}$ be an optimal tree s.t. f_1 & f_2 are siblings. (Assuming $f_1 \le f_2 \le f_3 \dots$)

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

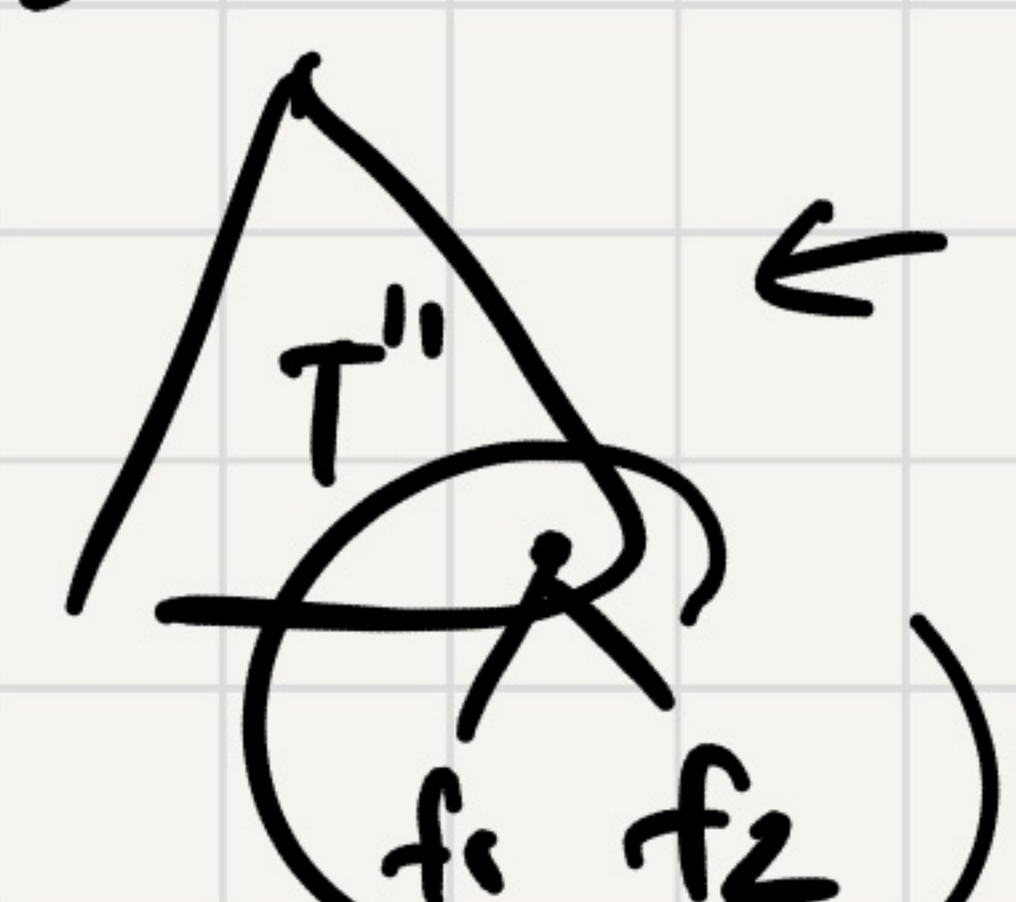
$\text{cost}(T'') = \text{cost}(T') + f_1 + f_2$.



Huffman merge f_1 & f_2 and then solves recursively on $\underbrace{f_1+f_2, f_3, \dots, f_n}$.



⇒



← Huffman on n freq.

optimal $\text{cost}(T''') \ge \text{cost}(T')$

$\Rightarrow \text{cost}(T'') = \text{cost}(T)$. □

Set Cover

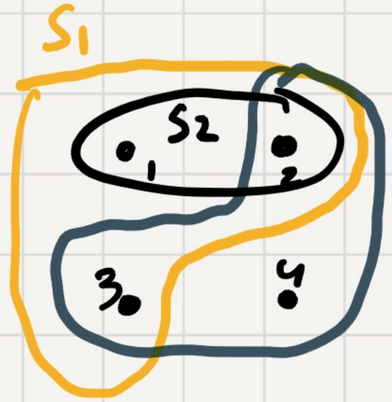
Input: Universe $U = \{1, 2, 3, \dots, n\}$

Collection of subsets $S_1, S_2, S_3, \dots, S_m \subseteq U$ $\quad$ (s.t. $S_1 \cup S_2 \cup \dots \cup S_m = U$.)

Output: Minimal subcollection that covers U.

Minimal size $J \subseteq [m]$ s.t. $\bigcup_{j \in J} S_j = U$

For Example:



$S_1 \cup S_2 \cup S_3 = \{1, \dots 4\}$

$J = \{1,3\}$ $S_1 \cup S_3 = \{1, \dots 4\}$

$J = \{2,3\}$ $S_2 \cup S_3 = \{1 \dots 4\}$

Greedy strategy? Pick at any time the set that covers most new points.

Greedy strategy:

Algorithm:

1. $J \leftarrow \emptyset$.
2. While $S_J \neq U$: ($S_J = \bigcup_{j \in J} S_j$)

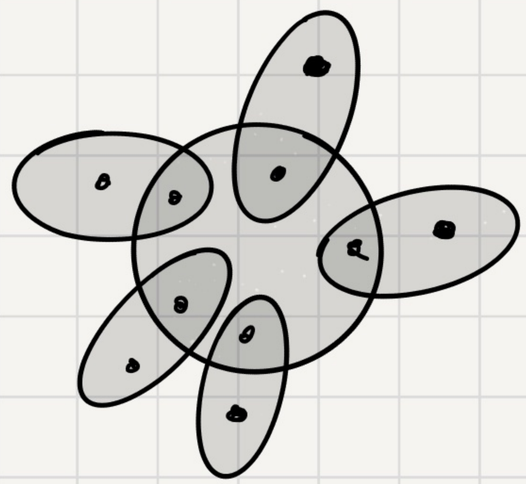
   Pick $i \notin J$ with largest $|S_i \setminus S_J|$

   (covers the most new points)

   Add $i \in J$.
3. output J.

Is it correct? No.

Counterexample:



Greedy: will pick all sets
$\Rightarrow |J| = 6$.

Optimal solution: 5.

Claim: "Greedy solution is not too bad":

If optimal solution uses k sets, then greedy finds J with $|J| \leq k \cdot \ln(n) + 1$.