- **Horn-SAT**

  Proof: If algorithm outputs a solution, by design of algorithm, the solution must satisfy all clauses 
  \[(x_1, x_2, x_3, \ldots, x_n)\]

  If algorithm outputs no, assume towards contradiction that there is a satisfying assignment 
  \[(x_1', x_2', \ldots, x_n')\]

  let \(i_1, i_2, \ldots, i_k\) be the order in which the algorithm sets the variables to true.

  **Case 1** if \(x_{i_1}', x_{i_2}', \ldots, x_{i_k}'\) are all true

  let \(C\) be the type 3 clause that assignment \((x_1, \ldots, x_n)\) violates,

  the variables in \(C\) must be in \(x_{i_1}, x_{i_2}, \ldots, x_{i_k}\)

  since \(x_{i_j}'\) is also true for \(j = 1, 2, \ldots, k\)

  \(C\) must be violated by \((x_{i_j}')\), contradiction

  **Case 2** let \(i_j\) be the first variable where

  \(x_{i_j} = \text{true} \Rightarrow x_{i_j}' = \text{false}\)

  when \(x_{i_j}\) were set to true

  **Case 2.1** \(x_{i_j}\) is set to true by a type 2 clause

  **Case 2.2** \(x_{i_j}\) is set to true by a type 1 clause

  in both subcases this particular clause will be violated by \((x_{i_j}')\), contradiction.

- **Hoffman tree**

  - cost of merging two characters = sum of their
- cost of the tree = sum of merge costs

- form a tree \[ \xrightarrow{\text{do } n-1 \text{ merge operations}} \]

- running time
  - naive implementation
    - \( n-1 \) iteration (every iteration reduces \#char. by 1)
    - \( \mathcal{O}(n) \) for each iteration
    - \( \mathcal{O}(n^2) \) use priority queue/heap
      - support: finding min. element, add, delete \( \mathcal{O}(\log n) \)
  - \( \mathcal{O}(n \log n) \)

- Proof of correctness:
  - we use induction.
  - Induction Hypothesis: Hoffman Tree algorithm finds an optimal encoding for all alphabets of size at most \( N \).
  - Base Case: when \( N = 1 \), there is only one solution with cost 0.
  - Induction Step: Assume I'H is true for \( N \), consider an alphabet of size \( n+1 \).
    - Assume towards contradiction that Hoffman Tree algorithm does not find the optimal solution. Let \( T \) be the tree found by algorithm
    - Let \( T' \) be the tree found by \( \mathcal{OPT} \), and \( i, j \) be the first two characters that the algorithm merged.
    - \( T' \) be two nodes at the highest depth in \( T \) that share the same parent
    - (note: one of \( i, j \) may overlap with one of \( i, j \))
let $i', j'$ be two nodes at the lowest level in $T_{opt}$ that share the same parent (note: one of $i', j'$ may overlap with one of $i, j$)

let $T_{opt}$ be a solution where $i, j$ are swapped with $i', j'$ in $T_{opt}$

let $d_i$ be depth of $i$ in $T_{opt}$ (similarly for $d_{i'}, d_j, d_j'$), we have

\[
\text{cost}(T_{opt}) = \text{cost}(T_{opt}) - (w_i d_i + w_j d_j + \alpha_i \alpha_j w_i w_j d_i d_j) + (w_{i'} d_{i'} + w_{j'} d_{j'} + \alpha_{i'} \alpha_{j'} w_{i'} w_{j'} d_{i'} d_{j'})
\]

\[
= \text{cost}(T_{opt}) - (w_i - w_{i'}) (d_i - d_{i'}) - (w_j - w_{j'}) (d_j - d_{j'})
\]

\[
\leq \text{cost}(T_{opt})
\]

here the last inequality is because

$w_i \leq w_i', w_j \leq w_{j'}$ (ALG has chosen two characters with lowest freq.)

$0 \leq d_i, d_{i'}, d_j, d_{j'}$ (both $i$ and $j$ have highest depth)

Therefore, $T_{opt}$ is also an optimal solution.

Now we know there is always an optimal solution that merges $i$ and $j$.

the problem reduces to an alphabet of size $m$

by induction hypothesis, Huffman tree algorithm is optimal for this instance

therefore $\text{cost}(\text{ALG}) \leq \text{cost}(T_{opt}) \leq \text{cost}(T_{opt})$, this

contradicts with the assumption that $\text{ALG}$ is not optimal.

Now we know $\text{ALG}$ is always optimal even for alphabet of size $m+1$,

this finishes the induction. $\square$