- Horn-SAT

Proof: If algorithm outputs a solution. by design of algorithm, the solution must satisfy all clauses.
$(x_1, x_2, x_3, \cdots, x_n)$

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

let $i_1, i_2, \cdots, i_k$ be the ordering in which the algorithm sets the variables to true.

case ① if $x_{i_1}', x_{i_2}', \cdots, x_{i_k}'$ are all true.

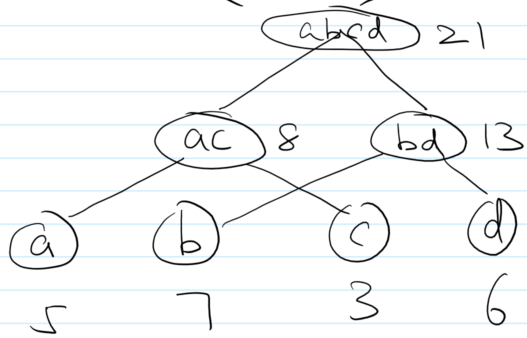
let C be the type 3 clause that assignment $(x_1, \cdots, x_n)$ violates,
the variables in C must be in $\underline{x_{i_1}, x_{i_2}, \cdots, x_{i_k}}$
since x_{i_j}' is also true for $j = 1, 2, \cdots, k$
C must be violated by (x_i'), contradiction.

case ② let i_j be the first variable where
$x_{i_j} = \text{true}$, $\underline{x_{i_j}' = \text{false}}$
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'), contradiction.

- Hoffman tree
  - cost of merging two characters = sum of their

- cost of the tree = sum of merge costs (frequencies)

- form a tree $\Longleftrightarrow$ do $n-1$ merge operations



- running time

  naive implementation

  $n-1$ iteration (every iteration reduces # char. by 1)

  $O(n)$ for each iteration

  $O(n^2)$

  use priority queue/heap

  - support: finding min element, add, delete $O(\log n)$

  $O(n \log n)$

- Proof of correctness.

  we use induction.

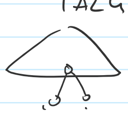
  Induction Hypothesis: Hoffman Tree algorithm finds an optimal encoding for all alphabets of size at most n.

  Base Case: when $n=1$, there is only one solution with cost 0.

  Induction Step: Assume IH 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_{ALG} be the tree found by algorithm T_{OPT} be the tree found by OPT, and i, j be the first two characters that the algorithm merged.

T_{ALG}



  if i, j are not children of the same node in T_{OPT}

  let i', j' be two nodes at the highest depth in T_{OPT} 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 highest depth 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 $d_j, d_{i'}, d_{j'}$), we have

$$\begin{aligned} cost(T'_{OPT}) &= cost(T_{OPT}) - (w_i \cdot d_i + w_j \cdot d_j + w_{i'} d_{i'} + w_{j'} d_{j'}) \\ &\quad + (w_i d_{i'} + w_j d_{j'} + w_{i'} d_i + w_{j'} d_j) \\ &= cost(T_{OPT}) - (w_{i'} - w_i)(d_{i'} - d_i) - (w_{j'} - w_j)(d_{j'} - d_j) \\ &\le cost(T_{OPT}) \end{aligned}$$

here the last inequality is because

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

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

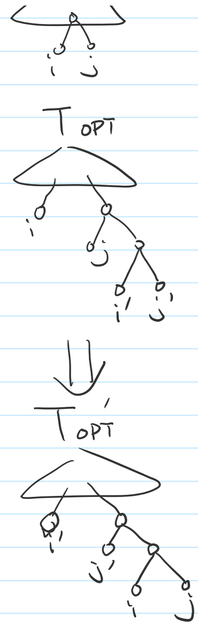
therefore, T'_{OPT} is also an optimal solution.

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

the problem reduces to an alphabet of size n

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

therefore $cost(T_{ALG}) \le cost(T'_{OPT}) \le cost(T_{OPT})$, this contradicts with the assumption that T_{ALG} is not optimal.



Now we know T_{ALG} is always optimal even for alphabet of size $n+1$, this finishes the induction. $\square$