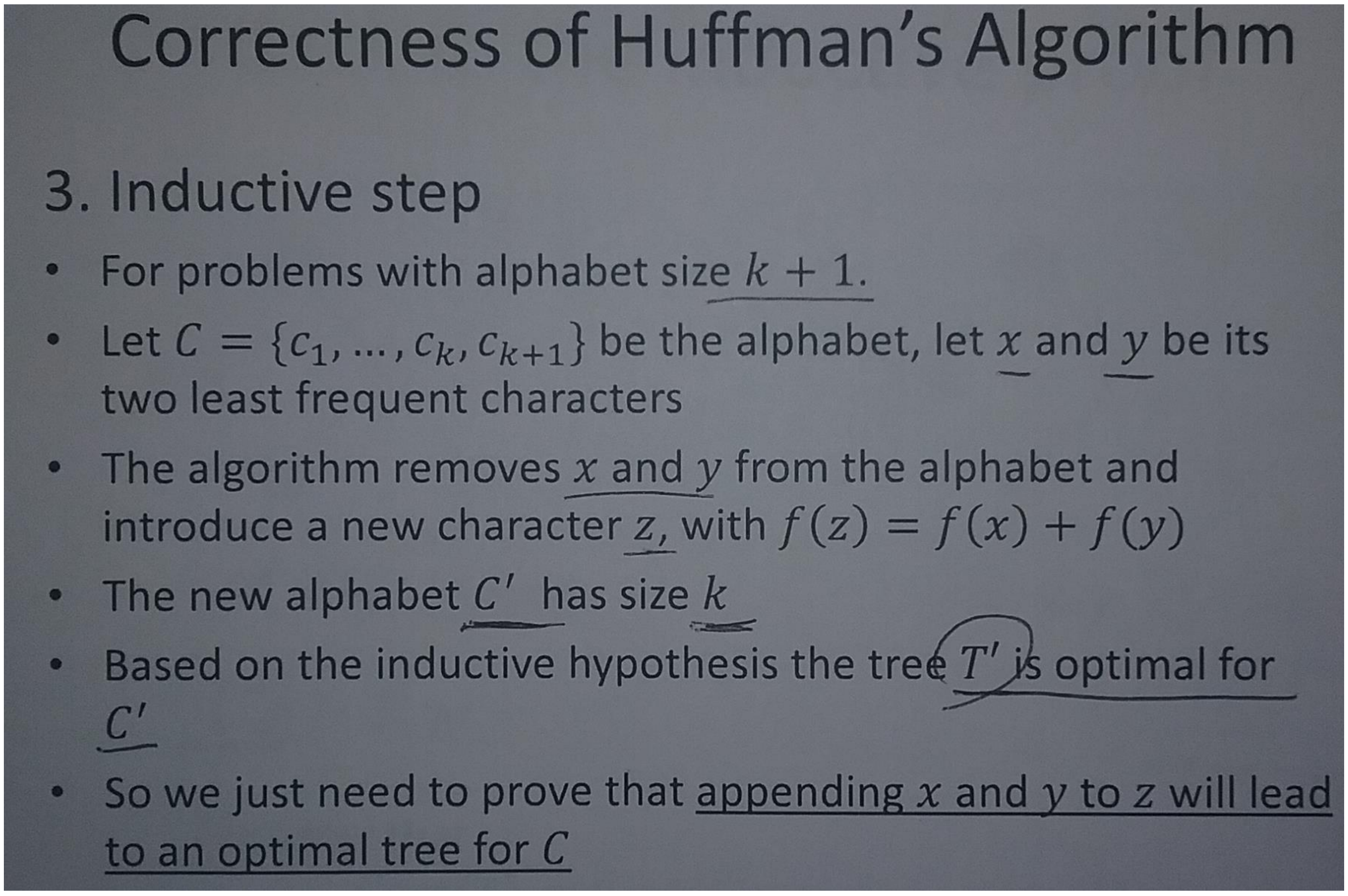
Week 8 Day 1 Lecture Notes

Prep:

- Read DPV 7.1 on Linear Programming
- Review Huffman Algorithm
- Continue reviewing for Quiz 4 and Final Test
- Revisiting Huffman recursion
- TA's are working on Implimentation 2 Grades. Expect them to arrive in a few days.

Proving the correctness of Huffman's Algorithm:

- Using Induction
- 1. Base Case: for n = 2 elements. Both elements are leaves.
- 2. Inductive Hypothesis. Assume optimal for size 2…k
- 3. Inductive Step: Prove for k+1.



Correctness of Huffman's Algorithm

3. Inductive step

- For problems with alphabet size $\underline{k+1}$.
- Let $C = \{c_1, \dots, c_k, c_{k+1}\}$ be the alphabet, let $\underline{x}$ and $\underline{y}$ be its two least frequent characters
- The algorithm removes $\underline{x \text{ and } y}$ from the alphabet and introduce a new character $\underline{z,}$ with $f(z) = f(x) + f(y)$
- The new alphabet $\underline{C'}$ has size $\underline{k}$
- Based on the inductive hypothesis the tree $\underline{T' \text{ is optimal for } C'}$
- So we just need to prove that $\underline{\text{appending } x \text{ and } y \text{ to } z \text{ will lead to an optimal tree for } C}$

In order to prove Huffman we need to prove that appending x and y to z will lead to an optimal tree.

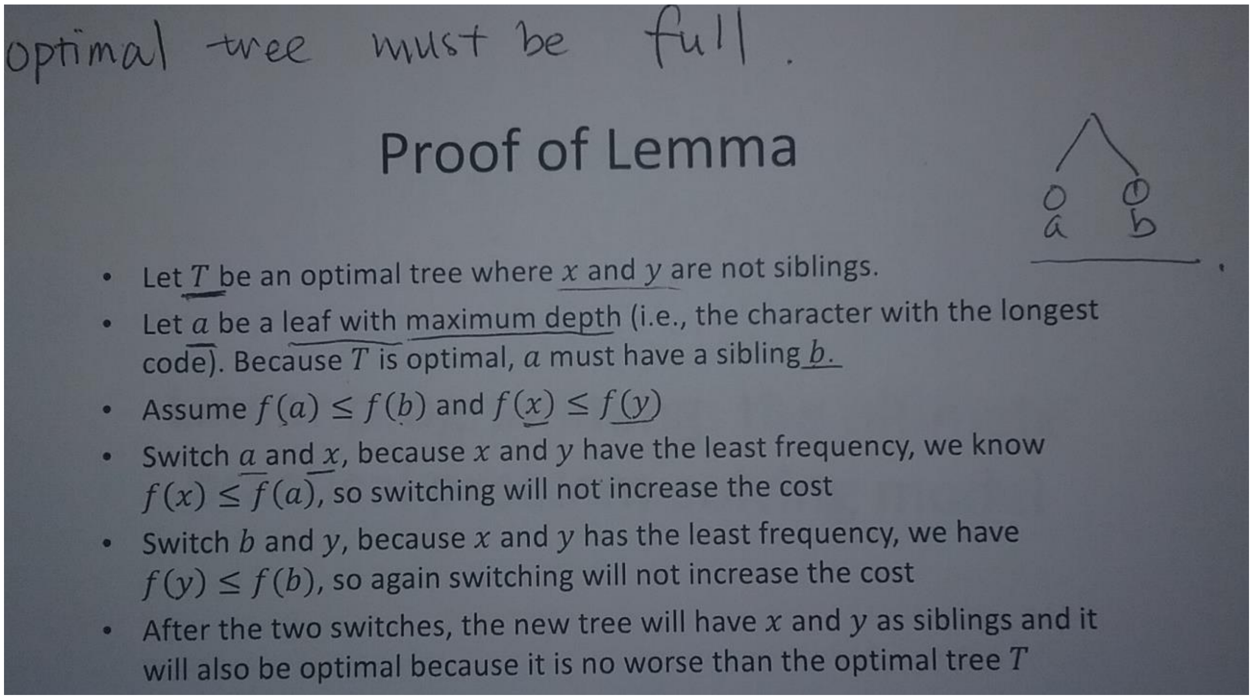
A Useful Lemma:

Let the two least frequent characters x and y, there must be an optimal tree where they are siblings.

Prove this by **the exchange argument**

- Start with Optimal Tree where x and y are NOT siblings.
- Construct the tree into one such that they are siblings.

Proof of Lemma (Picture)



optimal tree must be full.

Proof of Lemma

- Let T be an optimal tree where x and y are not siblings.
- Let a be a leaf with maximum depth (i.e., the character with the longest code). Because T is optimal, a must have a sibling b.
- Assume $f(a) \leq f(b)$ and $f(x) \leq f(y)$
- Switch a and x, because x and y have the least frequency, we know $f(x) \leq f(a)$, so switching will not increase the cost
- Switch b and y, because x and y has the least frequency, we have $f(y) \leq f(b)$, so again switching will not increase the cost
- After the two switches, the new tree will have x and y as siblings and it will also be optimal because it is no worse than the optimal tree T

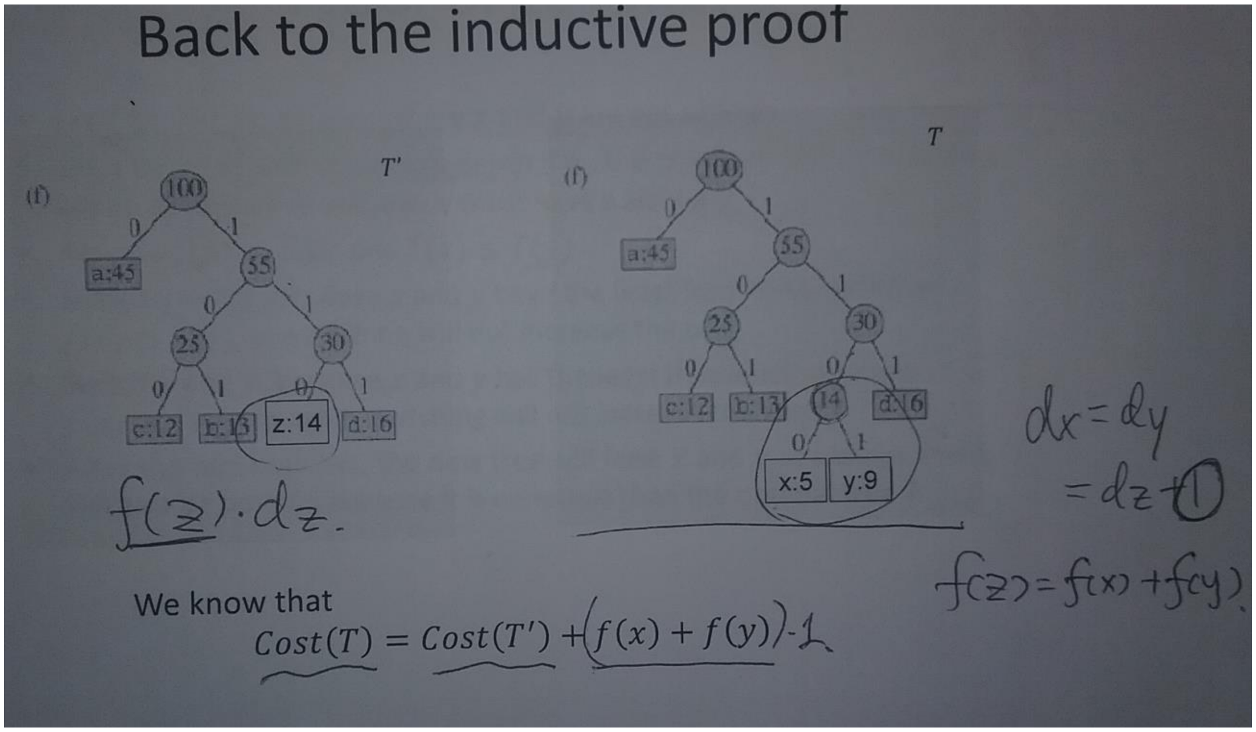
Key Information of the Lemma:

- Consider an optimal tree where x and y are NOT siblings.
- We have already proven that an Optimal Tree Must be Full.
- If nodes x and y are both the least frequent in the tree than they must be at the lowest depth of the tree.

- Since Every node that isn’t a leaf must have 2 children there must be at least 2 nodes at the lowest depth of the tree.
- Therefore we can switch the nodes around so that x and y are siblings. This will guarantee that this switch will not increase the cost because we know that x and y are the least frequent.

When we create the tree make sure to build bottom up. Starting with least frequent characters and going up to most frequent shorter depths. This will create an **Optimal Full Binary Tree.**

Back to the Inductive proof:



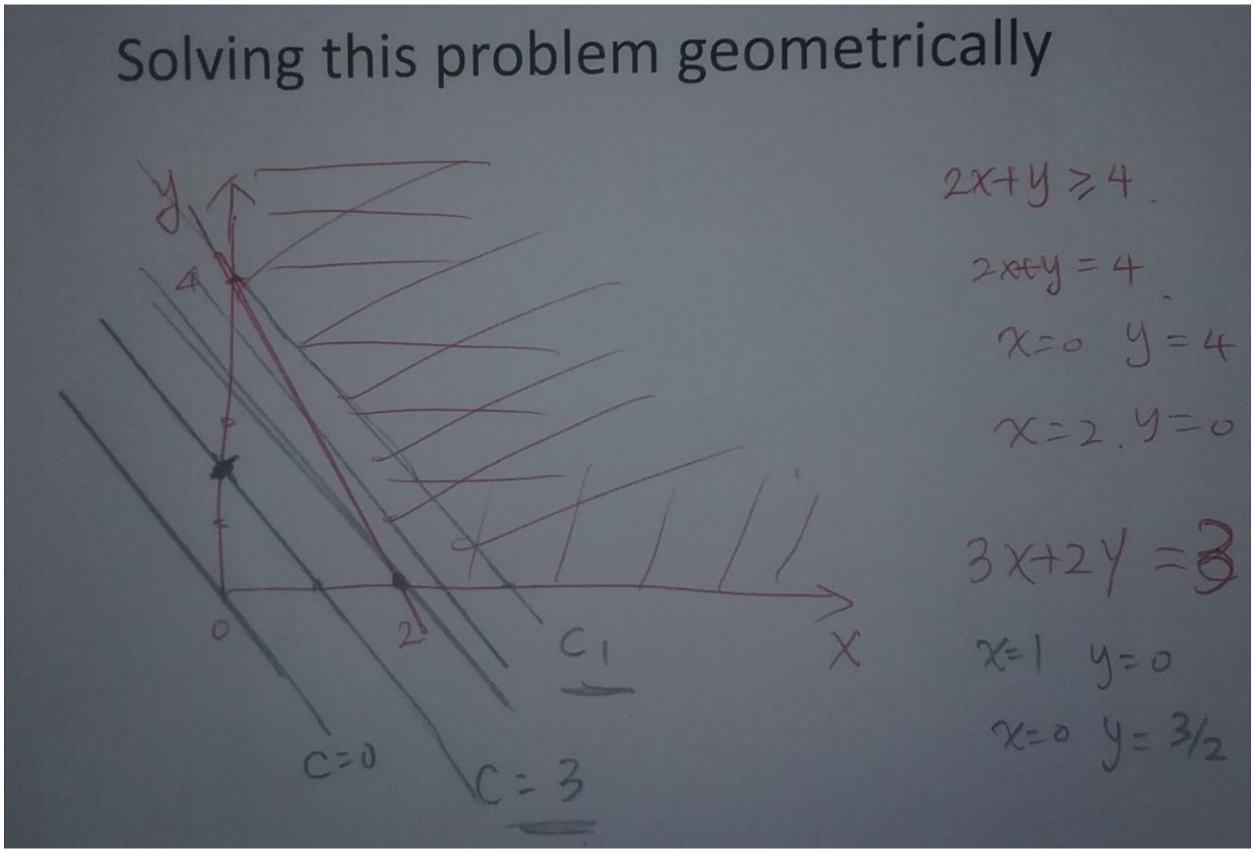
We know that

$$\underset{\sim}{Cost(T)} = \underset{\sim}{Cost(T')} + \underline{(f(x) + f(y))} \cdot 1$$

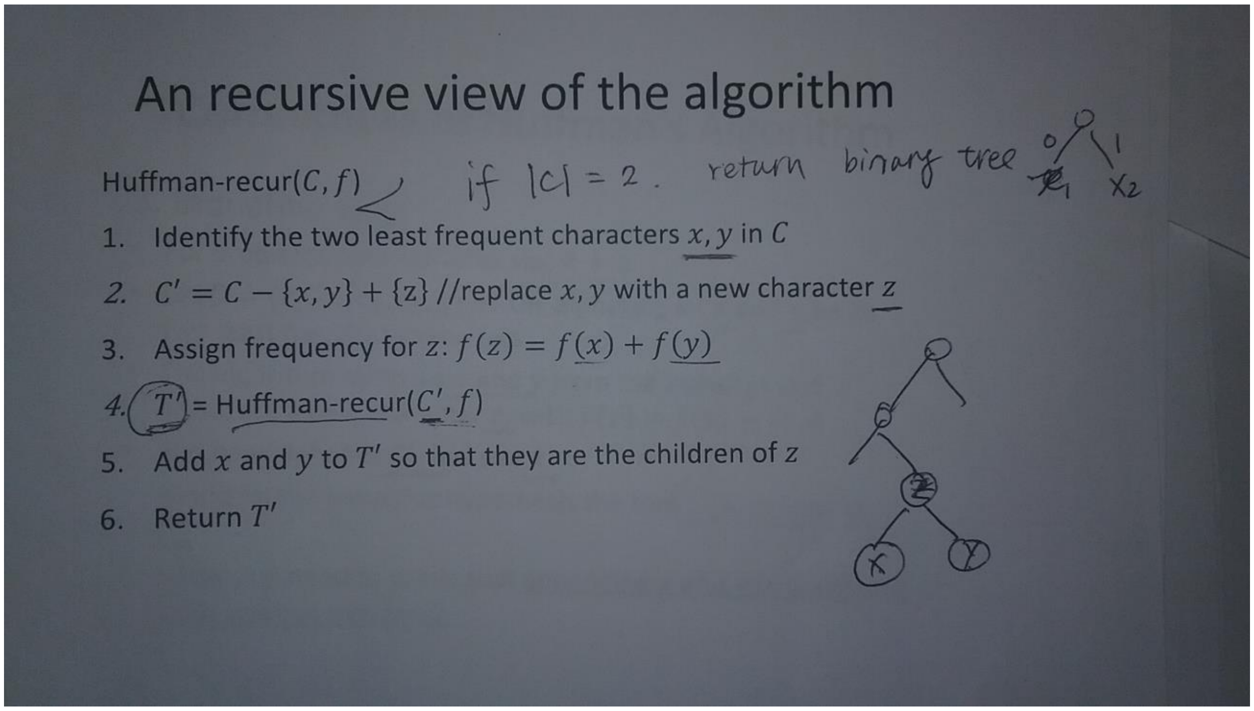
Looking at how depth of the nodes affect their costs.

Since the depth increases by 1 for x and y compared to z the:

- Cost (T) = Cost(T’) + (f(x) + f(y)) * 1
- Increases by the increased depth of T (+1)



Solving this problem geometrically



An recursive view of the algorithm

```
Huffman-recur($C, f$)   if $|C| = 2$. return binary tree
1. Identify the two least frequent characters $x, y$ in $C$
2. $C' = C - \{x, y\} + \{z\}$ //replace $x, y$ with a new character $z$
3. Assign frequency for $z$: $f(z) = f(x) + f(y)$
4. $T'$ = Huffman-recur($C', f$)
5. Add $x$ and $y$ to $T'$ so that they are the children of $z$
6. Return $T'$
```

Now Lets consider an optimal tree T^ with x and y as lowest depth nodes and convert it into T^’ that has z in place of x and y.

*We’re going backwards now, instead of going from z to (x and y) we’re going from (x and y) to z.

Review: Since x and y are the least used characters they must be at the lowest depth in the tree. And the lowest depth must have at least 2 nodes so x and y can be siblings in at least 1 optimal case.

➔End of Huffman Code Lecture

Linear Programming Lecture Start: Practical Problem-Solving Model.

What is Linear Programming?

- Linear objective and constraints.
- Tool for optimal allocation of scares recourses.

Why does it matter so much?

- Dominates Industry efficiency.
- Fast solvers are available.
- Ranked among most important scientific advances of the 20th century.

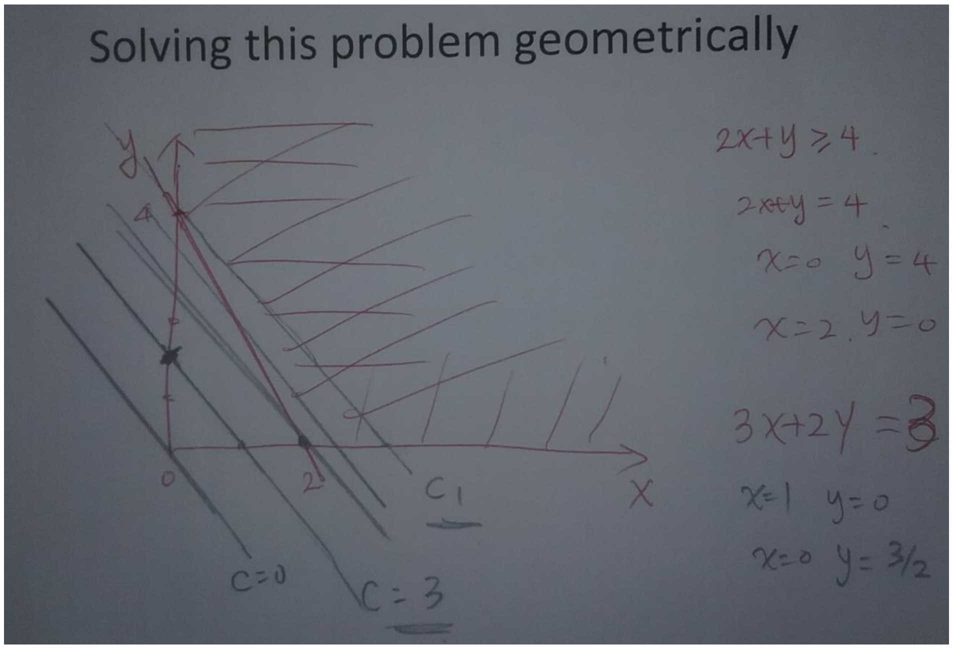
The Diet problem:

Steak: 2 units of steak protein cost 3$/Pound

Peanut Butter: 1 Unit of Peanut Butter protein costs 2$/Pound

Needs 4 units of protein/day.

X = steak Y = Peanut Butter. $2X + Y >= 4$



Formal Definition for Linear Program:

- Define a problem set n with variables x1,x2,….up to xn.
- A linear Objective Function which can be minimized or maximized;
    - Min/Max c1x2 + c2x2 + …. Cnxn
- A set of m linear constraints. A constraint looks like:
- Aix1+ ai2x2 + …. Ainxn <= (or >=,=)bi

Ended on Solution to LP Lecture*

Next Time:

- **Quiz 4** on Linear Programming and Reductions
- **Implementation 3** requirements will be discussed
- Continue to review slides as we near finals

End of Week 8 Day 1 Notes

~Information Composed by Notetaker Scott Russell for CS 325 **DAS** students.