<u>Week 7 Day 2 Lecture Notes</u>

Prep:

-Implementation 2 due tonight @midnight.

- 1 Day Grace period with no point reduction.

- Read DPV Chapter 5.2

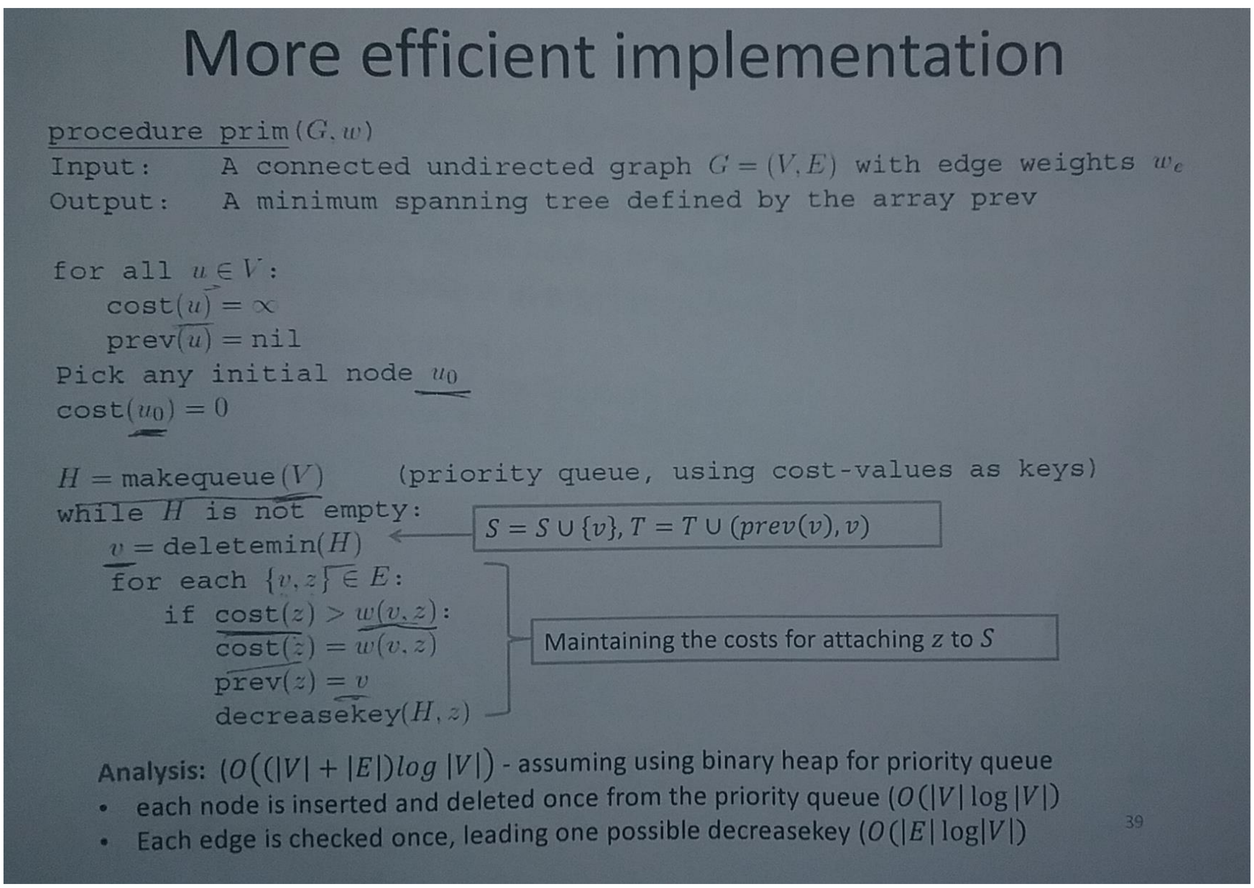
Minimum Spanning Tree and Hoffman Coding.

- Review Prims and Kruskal's Algorithm. Every edge can be constructed with a cut that the edge is cheapest across that cut. Used to prove correctness.

Prims Algorithm:

Prims algorithm Review: It randomly picks a node to start with.

Prims Algorithm Pseudocode:



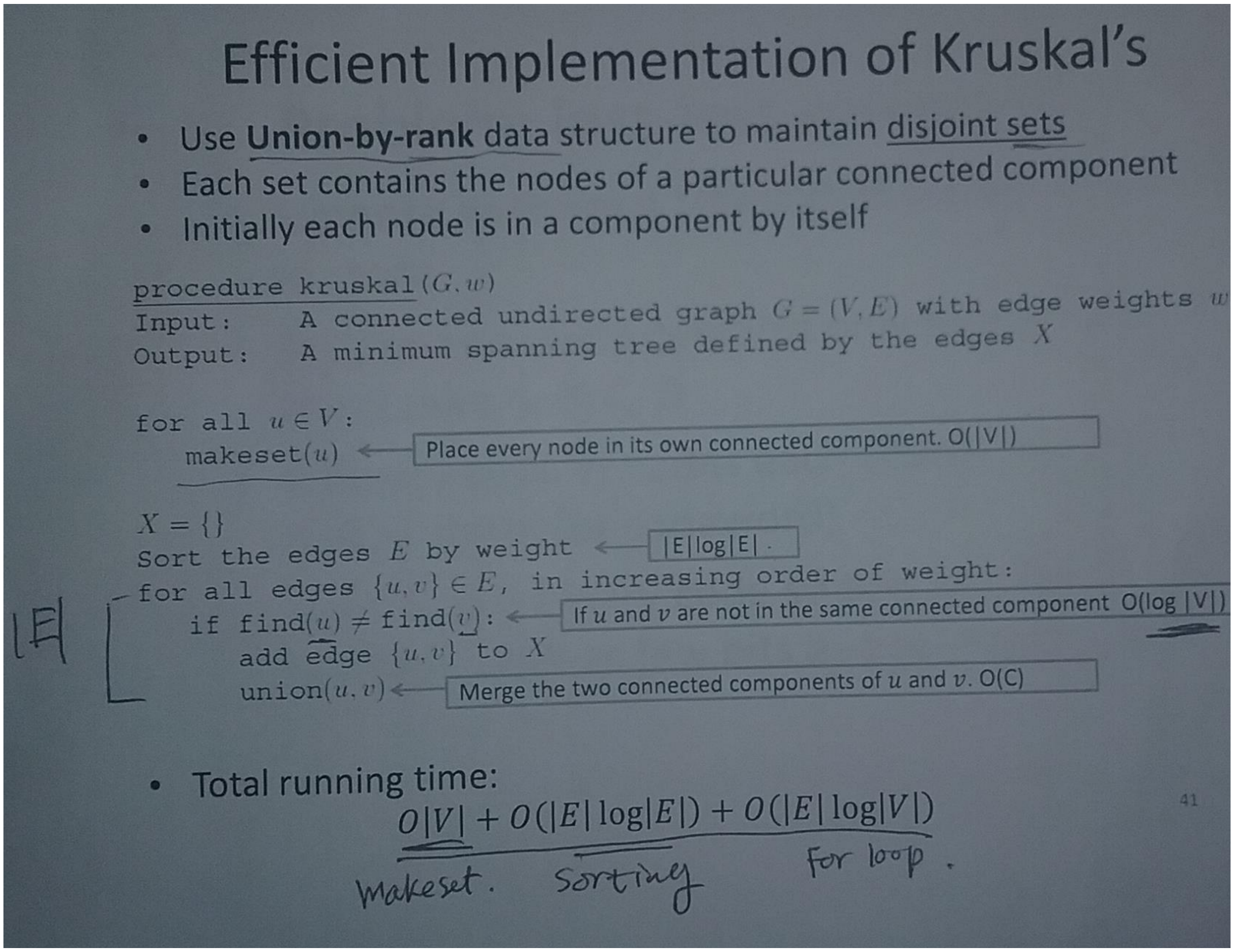
Can we improve the runtime?

Binary Priority queue. stores the cost of reaching every node from the original node u

Kruskal's Algorithm: Sorts edges in increasing weight order.

- Adds the cheapest edges one at a time (as long as the edge doesn't create a cycle)
- To accomplish this we check if there is already a path between the nodes that the edge we are checking connects.
- Runtime: Sorting time + cycle through all the edges. Simplified as $O(|E|^2)$

Union-By-Rank Efficient Kruskal:



- Compared to the original we maintain a tree structure.

- Maintain disjoint sets.
- **Runtime:** O(V) + O(|E|log|E|) + O(|E|log|V|)
- **Much faster than E^2.**

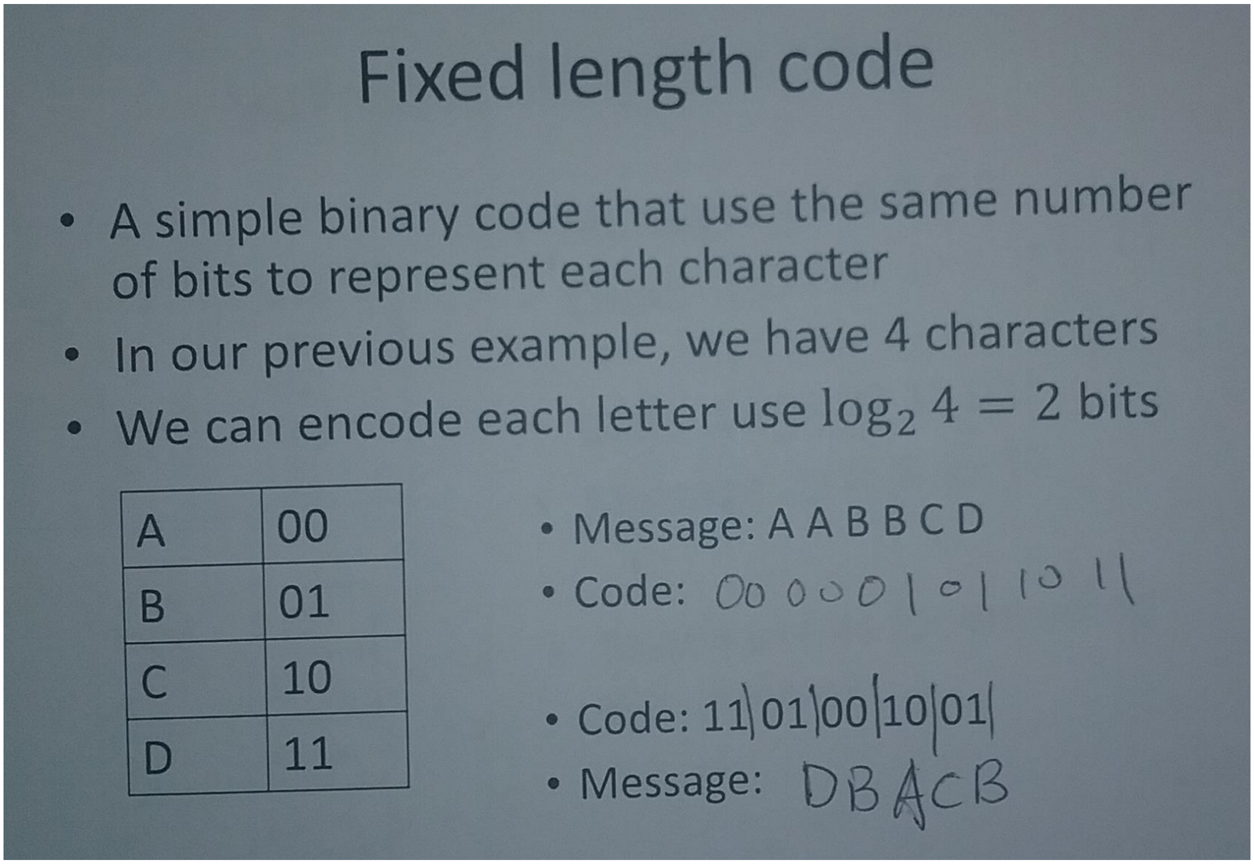
<u> The information above is not required material for final review. It's to help enhance your knowledge about the class and Algorithm improvement.**</u>**

Huffman Coding

Loss-Less Compression:

- Assume we have 100,000 characters
- How can we encode the data without any loss while storing data in a binary form which is much cheaper than storing the asci characters?

Fixed Length Code:



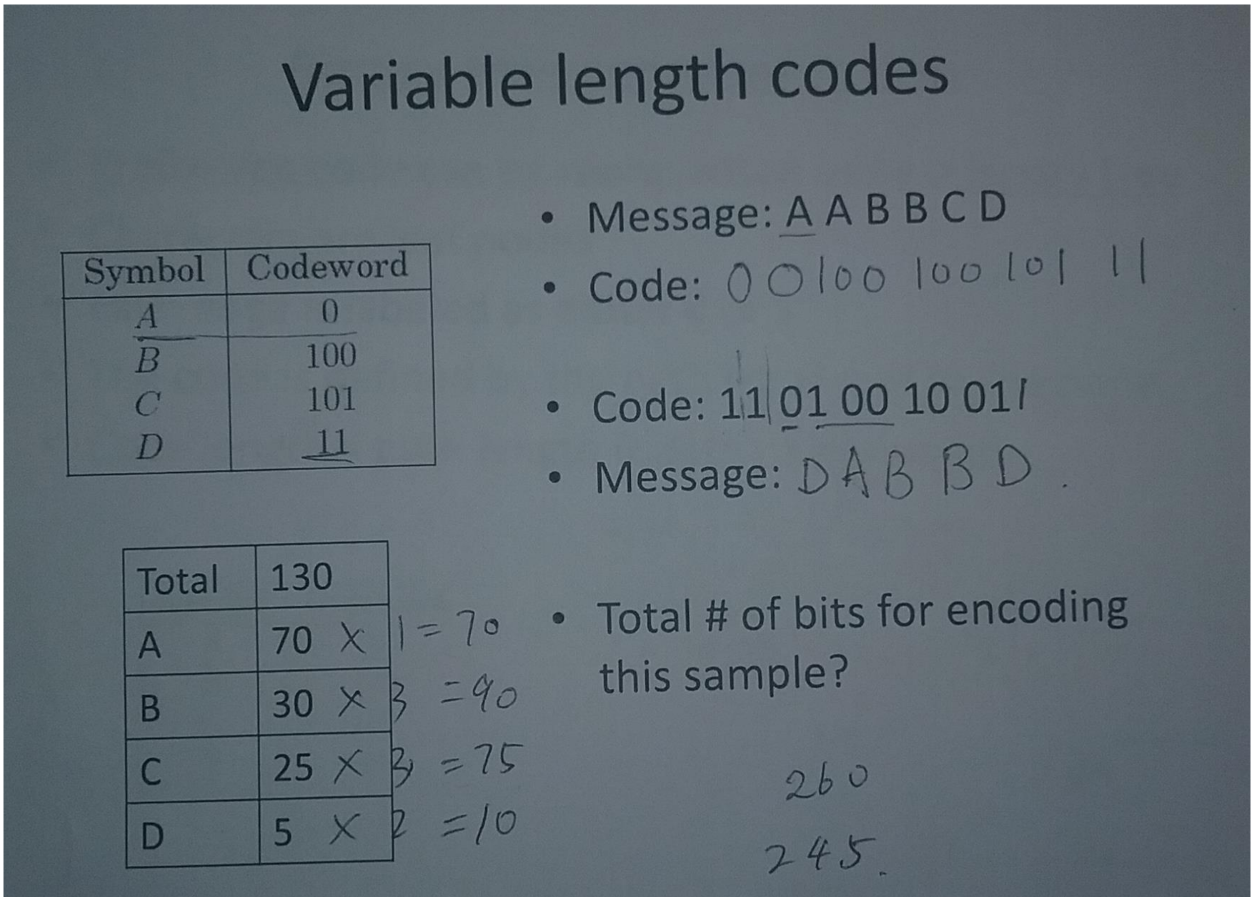
- For simple code we can represent 4 characters in a binary bit.
- A = 00, B = 01, C = 10, D = 11.

- Because we have 2 bits for every character we compress the data from original 8 bits down to 2 bits.

Coding Efficiency:

Data sample where the frequency of each amount is different.

- Make higher frequency characters use shorter representation. (Similar to how English words that are used allot are usually very short words)
- **But how can we encode and decode if we have variable bit length?**



How do we avoid Ambiguity? The code must be pre-fix Free. (No code could be the prefix of another code)

Example: If we have the code A = 00. B = 001. C = 101 and C = 110.

Since A=0 and B starts with the Prefix of 00.

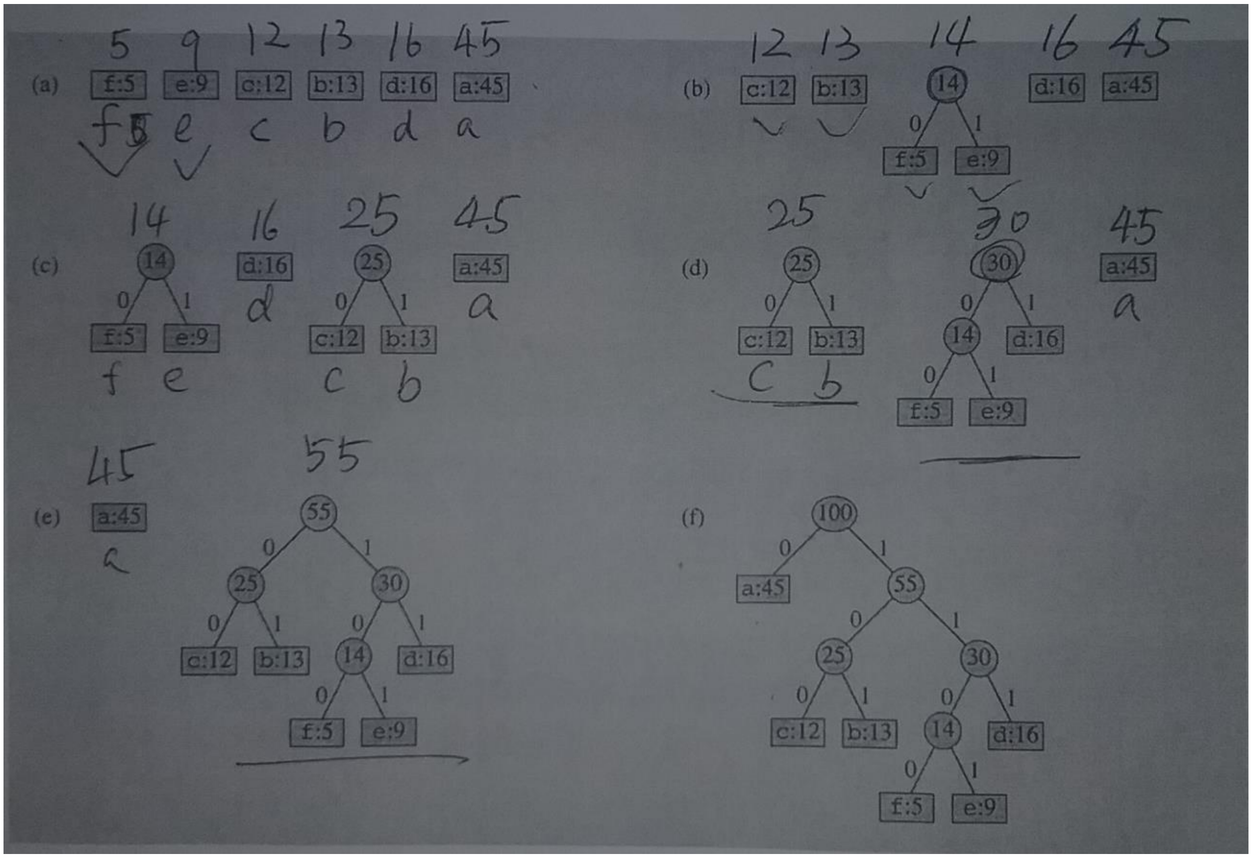
Solve Ambiguity:

****NO LETTER CAN HAVE THE PREFIX OF ANY OTHER LETTER.****

Binary Tree Representation:

- Every Character must be a leaf node.
- If a character is not a leaf node than it would be the pre-fix of another character.
- So every character has to be a leaf node to prevent Prefix Ambiguity.

Decoding is easy with the prefix tree: See picture below for Tree Traversal



Optimal Prefix Coding Problem:

How do we determine the optimal Binary Tree to minimize the average code length?

Optimal Tree <u>Must</u> be a Full Binary Tree:

Full Binary Tree: Every Node (except leaf nodes) have two children.

Think Greedily.

Start by creating leaves for the least-frequent characters.

- See Picture of Tree* showing the most expensive leaves first.
- For each combination add a parent node that contains both.

Ended on Recursive view for solving*

Next time:

- Next Lecture will start with proving the correctness of Huffman's Algorithm
- Implementation 2 Due tonight.
- Continue Reviewing lectures
- Read DPV Chapter 7.1

<u>End of Week 7 Day Notes</u>

~Information composed by Notetaker Scott Russell for CS 325 **DAS** student