CSCI 2300: Introduction to Algorithms

Homework 7

Lucien Brule
Prof. Bulent Yener
April 27, 2023

1 Problem 1

Problem: Given an undirected, weighted graph G = (V, E). Assume that no two edges have the same weight. Prove that there is a unique minimum spanning tree (MST) for such a graph.

Proof:

To prove the uniqueness of the MST, we'll use proof by contradiction. Assume that there are two distinct minimum spanning trees, MST1 and MST2, for the given graph G with the same total weight. Let's consider the edge with the smallest weight that is in MST1 but not in MST2. We denote this edge as e. Since MST1 and MST2 are spanning trees, they are connected and acyclic.

Now, let's add edge e to MST2. Adding e to MST2 will create a cycle since MST2 is a tree. In this cycle, there must be at least one edge that is not in MST1, as MST1 and MST2 are distinct. We denote this edge as f. Since e was the edge with the smallest weight not in MST2, the weight of e must be less than the weight of f.

We can now create a new tree, MST3, by removing edge f from MST2 and adding edge e. The total weight of MST3 will be less than the total weight of MST2 because the weight of e is less than the weight of f. However, this contradicts our assumption that MST2 is a minimum spanning tree, as we have found another tree with a smaller total weight.

Thus, our assumption that there are two distinct minimum spanning trees is incorrect, and there must be a unique minimum spanning tree for the given graph G.

2 Problem 2

Problem: Given a graph G=(V,E), we want to find the Minimum Spanning Tree (MST) using the following algorithm: Sort the edges in decreasing order of their weights. For each edge e in sorted order, if e creates a cycle in G, remove it from G. Prove that this method is correct, and analyze its running time.

Solution:

The given algorithm is called the Reverse-Kruskal Algorithm. To prove its correctness, we'll demonstrate that it is equivalent to Kruskal's algorithm but in reverse order. Here is the proof:

1 Let's assume we have a graph G and its MST T. Let T' be the result of the Reverse-Kruskal algorithm.

2 Let's consider the edges of G in increasing order of their weights. Let's denote these edges as e1, e2, ..., en.

3 Let's consider the edges of G in decreasing order of their weights. Let's denote these edges as en, en-1, ..., e1.

4 In Kruskal's algorithm, we add the edges in increasing order of their weights. In Reverse-Kruskal, we add the edges in decreasing order of their weights.

5 In Kruskal's algorithm, we add an edge to the MST if it doesn't create a cycle. In Reverse-Kruskal, we remove an edge from the MST if it creates a cycle.

6 Thus, Kruskal's algorithm and Reverse-Kruskal are equivalent except for the order in which they add edges to the MST.

7 Since Kruskal's algorithm is correct, Reverse-Kruskal must also be correct.

Now, let's analyze its running time by considering the following steps:

1 Sorting edges: $O(|E|log|E|)$

2 Checking for cycles and removing edges: $O(|E|\alpha(|V|))$ using Union-Find data structure with path compression and union by rank

The overall running time is $O(|E|log|E|)+O(|E|\alpha(|V|)) = O(|E|log|E|)$, where $\alpha(|V|)$ is the inverse Ackermann function, which grows extremely slowly and can be considered almost constant for practical purposes.

3 Problem 3

Problem: Given an alphabet with n characters and their respective frequencies, answer the following questions:

1. Show that the frequencies of all characters sum to 1 (as they should) for any n.

2. Show what the Huffman encoding is for each character.

3. What is the expected number of bits per character?

Solution:

a) We have the frequencies $f_i = \frac{1}{2^i}$ for $i = 1, 2, \ldots, n-1$ and $f_n = \frac{1}{2^{n-1}}$. Let's sum all frequencies:

$$\sum_{i=1}^{n} f_i = \sum_{i=1}^{n-1} \frac{1}{2^i} + \frac{1}{2^{n-1}}$$

Since this is a geometric series, we can apply the geometric series formula:

$$\sum_{i=1}^{n-1} \frac{1}{2^i} = \frac{1 - \frac{1}{2^{n-1}}}{1 - \frac{1}{2}} = 1 - \frac{1}{2^{n-2}}$$

Now, adding the last frequency term:

$$1 - \frac{1}{2^{n-2}} + \frac{1}{2^{n-1}} = 1 - \frac{1}{2^{n-2}} + \frac{1}{2(2^{n-2})} = 1$$

This shows that the sum of all frequencies is indeed 1.

b) To build the Huffman encoding tree, we start with the lowest frequencies and build up:

1. Combine the two lowest frequencies (in this case, f_n and f_{n-1}) to form a new subtree with the combined frequency. 2. Remove the original two nodes from the list and insert the new subtree back into the list, maintaining the frequency order. 3. Repeat steps 1 and 2 until the list contains only one tree.

In this case, we'll end up with a tree where the left child always has a higher frequency than the right child. The Huffman encoding for each character is the path from the root to that character, where going left is a '0' and going right is a '1'.

c) To find the expected number of bits per character, we can use the formula $\sum l_i f_i$. In this tree, the encoding length l_i is equal to the depth of character i in the tree. Since the tree is built in ascending order of frequency, the depth of character i is just i.

Thus, the expected number of bits per character is:

$$\sum_{i=1}^{n} l_i f_i = \sum_{i=1}^{n} i \cdot \frac{1}{2^i}$$

To find a closed-form expression for this, we can use the following trick:

$$S = \sum_{i=1}^{n} i \cdot \frac{1}{2^i}$$
$$2S = \sum_{i=1}^{n} i \cdot \frac{1}{2^{i-1}}$$

Now, subtract the second equation from the first:

$$S - 2S = -S = \sum_{i=1}^{n} \frac{1}{2^i} - \sum_{i=1}^{n} \frac{i}{2^{i-1}}$$
$$-S = \sum_{i=1}^{n} \frac{1-i}{2^i}$$

Now, notice that for $i \geq 2$, we can write the summand as a telescoping sum:

$$\frac{1-i}{2^i} = -\frac{i-1}{2^i} + \frac{i}{2^i} = \frac{1}{2^{i-1}} - \frac{1}{2^i}$$

Thus, we have:

$$-S = \left(\frac{1}{2}\right) + \sum_{i=2}^{n} \left(\frac{1}{2^{i-1}} - \frac{1}{2^i}\right)$$
$$-S = \frac{1}{2} + \left(\frac{1}{2} - \frac{1}{4}\right) + \left(\frac{1}{4} - \frac{1}{8}\right) + \cdots + \left(\frac{1}{2^{n-1}} - \frac{1}{2^n}\right)$$

Notice that most terms cancel each other out, so we are left with:

$$-S = \frac{1}{2} + \frac{1}{2} - \frac{1}{2^n}$$

Now, we can find the expected number of bits per character S:

$$S = 1 - \frac{1}{2^{n-1}}$$