Theory of Algorithms Homework 1

Evan Dreher

August 2022

1 Problem 1

Consider the following problem: You have a number line, on which an arbitrary number of points have been placed; each point can be identified by its coordinate location (i.e {1.6, 2.3, -19000}. You can place covers of length 1 onto the number line. For instance, you might place a cover that stretches over the range [1.5, 2.3], which would ocver two points in the list above. Your goal is to cover all of the points in your list using the fewest number of covers possible. Describe an optimal greedy strategy for doing so and explain *why* you this your strategy is optimal.

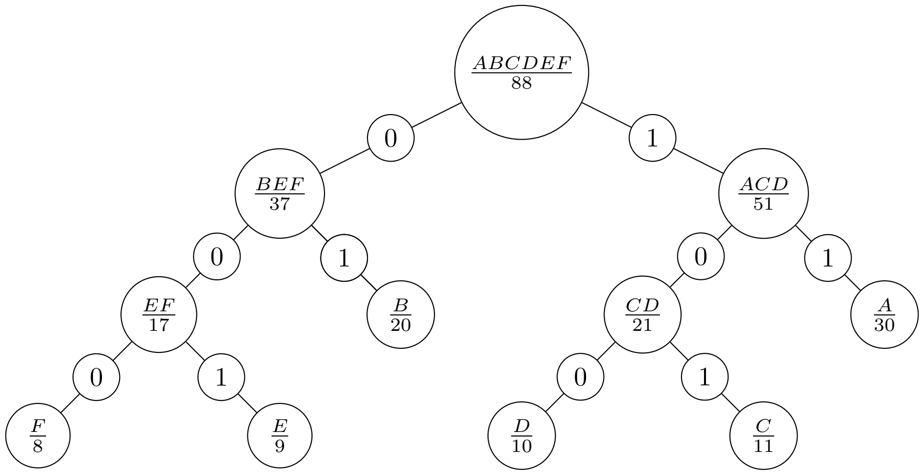
This solution assumes the input list of numbers is finite, and given to us in unsorted order.

1. Sort the input list with some efficient sorting algorithm such as quicksort (Note: counting sort is risky to implement here since we don't know the range of values).
2. Iterate over the sorted list once placing the left edge of a cover over the point with the smallest value that is not yet covered

Explanation: For any finite set S of points that can be on a number line, there exists a smallest point of that set, let us call that point x. At some point we have to cover x, but since there are no points to the left of it, we want to place the left-edge or our cover exactly on top of x (i.e covering the interval [x, x+1]). This is in order to maximize the number of points we cover on the right, thus this cover is part of the optimal solution. Once we place this tile, our remaining points S' are a subset of the original points S. We can repeat this process with S' by placing the left-most edge of our next tile on the smallest element of S'. We repeat this process until S' is the empty set and we have coverd all points. Since each tile was part of the optimal solution, the algorithm gives an optimal solution.

2 Problem 2

Draw the Huffman code for the following frequencies: $A = 30, B = 20, C = 11, D = 10, E = 9, F = 8$. When given a choice, always assign the smallest value to the left "0" branch.



- A: 11
- B: 01
- C: 101
- D: 100
- E: 001
- F: 000

3 Problem 3

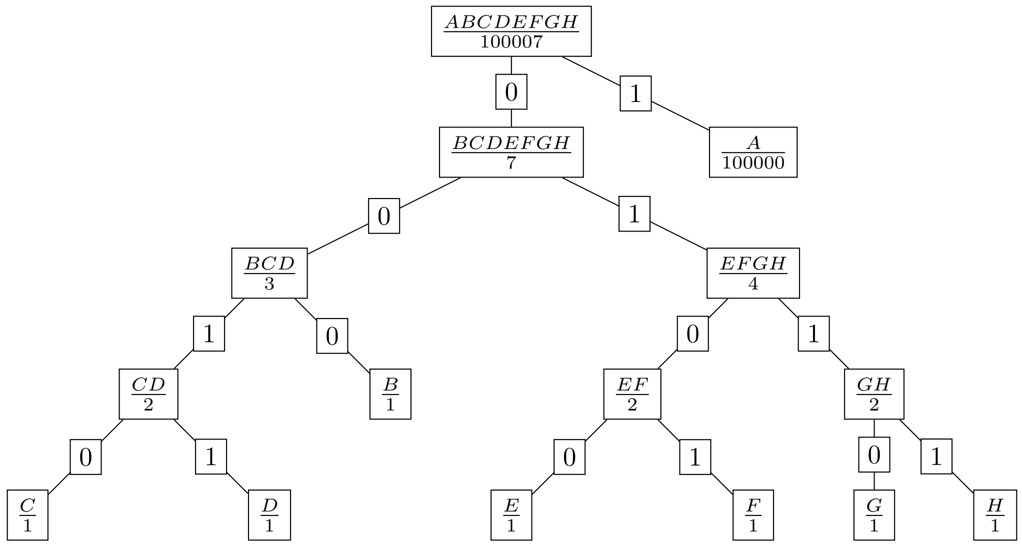
Imagine a Huffman coding problem where you have some number of letters, each with a frequency with which it appears in the message. Show frequencies such that these two requirements are simultaneously met.

(a) The number of unique letters is a power of 2.

(b) the Huffman code uses no more than half as many bits as the naive solution.

Show both the code that results, and the number of bits used by both solutions.

Consider the following frequencies: A=100,000, B=1, C=2, D=1, E=1, F=1, G=1, H=1.



Letter	Code	Bits
A	1	1
B	000	3
C	0010	4
D	0011	4
E	0100	4
F	0101	4
G	0110	4
H	0111	4

The Naive solution uses 3 bits per character for a total of 300,021 bits (100,007 characters). The Huffman encoding does 1 bit for all the A's, 3 bits for the B, and 4 bits for the remaining 6 letters for a total of 100,000+3+4(6)=100,027 bits. The huffman encoding uses approzimately a third of the number of bits the naive solution uses.

4 Problem 4

(a) Suppose we modify the set-covering problem so that we are only allowed to cover each element once. (So, if our target is $\{1, 2, 3\}$, we chould choose source sets of $\{1, 2\}$ or $\{2, 3\}$ but not both.) Your objective is to cover as much of the target as possible, regardless of how many souce sets it takes. Show a situation in which the following greedy strategy is guaranteed to be sub-optimal: of the legal choices (i.e) those not overlapping choices you've already made), always choose a set that overlaps as few of the remaining legal source sets as possible.

- **Sets:**
- {a, b, c, d, e}
- {e, f}
- {e, a}

{a, b, c, d, e} overlaps both of the sets, {e, a} overlaps both as well but {e, f} overlaps only 1, so the algorithm will pick that one which then locks itself out of picking the other two later on. This is suboptimal because picking {a, b, c, d, e} is the optimal choice.

(b) Same problem, but now choosing the set that coverse as much of the target set as possible.

- **Sets:**
- {a, b}
- {c, d}
- {a, b, c}

{a, b, c} will be chosen since it covers three elements but it prevents us from picking the other two sets which is the optimal solution.

5 Problem 5

You're laying out sticks on a number line again, just like in problem 1. This time, however, each stick covers only a specific range of numbers, which might be of any length. (For instance, a stick might cover everything in the range [2,11] or [9,9.1]). You'd like to lay down as many sticks as you can, but you can't lay any two sticks that overlayp - you can't lay the two in the example above, for instance. Again, you have "points" you want to cover on the number line - assume that it is *possible* to do so, given the set of sticks you have. Show an example in which the following greedy strategy is guaranteed to be sub-optimal: always lay a (legal) stick that covers as small of an area as possible.

Let S be a set of two sticks with the ranges [-100,-1], [-2, 2]. Let P be the set of points we are trying to cover where $P = \{-50\}$. The greedy algorithm places down the stick [-2, 2] since it has the smallest area and then terminates since there are no more legal sticks in S. This doesn't cover any of the points in P and the optimal solution was clearly to lay down the larger stick meaning this solution is suboptimal.

6 Problem 6

Consider the following problem: you're attempting to pave a sidewalk that is three feet wide, and a variable number of feet long. To do the job, you have to lay a number of paving tiles; each tile is a rectangle two feet long and one foot wide. Paving tiles can be laid "longways" or "widthways", but they can't overlap, and you need to cover the entire surface of the sidewalk in paving tiles. You want to know the number of possible arrangements of tiles to do the job. After some experimentation, you determine that no odd-length sidewalk will be an odd number, and can be covered at all - which makes sense, because the square footage of the sidewalk would be odd and each tile covers 2 square feet. For even lengths, you derive the following values for different lengths of sidewalk:

Length	2	4	6	8
Permutations	3	11	41	153

... then you decide experimenting is a bad idea. You have ten feet of sidewalk to cover; how many ways can you tile it? Describe a dynamic programming solution to this problem, and show how it calculates your final answer.

The smallest chunk of width 3 that we can fit our pieces is a 2×3. When we try to find how many arrangements there are in a $n \times 3$ (when n is even) we will solve it dynamically by splitting it into multiples of a 2×3 chunk.

for a 2×3 chunk there are no splits to make so there are only 3 ways to arrange it.

for a 4×3 chunk we can split it into 2 2×3 chunks resulting in 9 arrangements. But we can also arrange any $n \times 3$ chunk where $n > 2$ is even in an additional 2 ways so if we make no splits we add an additional 2 arrangements for a total of 11.

for a 6×3 we start by making a split at 2 resulting in a 2×3 and a 4×3. The top chunk has 3 arrangements and the bottom chunk can be arrange in 11 ways for a total of 33. We then make a split at 4 resulting in a 4×3 and a 2×3. The top chunk can be arranged 2 ways without making any cuts and 3 for the bottom resulting in 6 additional arrangements. Lastly by making no splits we get an additional 2 arrangements for a total of 41.

for an 8×3 we once again start by making no splits for 2 arrangements. Then we make a split at 2 for a 2×3 and a 6×4 which can be arranged in 3 and 4 ways respectively for a total of 123. From here we can see that the first step in calculating the total number of arrangements will be L[i] = 3*L[i-1] + 2. Next we make a cut at 4 for two 4×3. The upper chunk can be arrange 2 ways with no cuts and the lower can be arrange in 11 ways for a total of 22. We then make a cut at for a 6×3 and a 2×3. The upper can be arranged in 2 ways without cuts and the lower can be arranged in 3 for a total of 6. This is for a total of 123 + 22 + 6 + 2 = 153.

From here we can see that $L[i] = 3 \times L[i-1] + 2(L[i-2]...L[1]) + 2$. Where L[1] is the number of arrangements for length 2, L[2] is the number of arrangements for length 4 and so on so forth. This can be simplified to $L[i] = 3 \times L[i-1] + 2(L[i-2]...L[1] + 1)$. By that equation $L[10] = 3 \times 153 + 2(41 + 11 + 3 + 1) = 459 + 82 + 22 + 6 + 2 = 571$