CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

HW 10

Due November 8st (Friday), 11:59 pm

Formatting: Each problem should begin on a new page. When submitting in Gradescope, try to assign pages to problems from the rubric as much as you can. Make sure you write all your group members' names. For the full policy on assignments, consult the syllabus.

1. (15 pts.)
Suppose we have an optimal prefix code on a set $C = \{0, 1, \ldots, n-1\}$ of characters and we wish to transmit this code using as few bits as possible. Show how to represent any optimal prefix code on C using only $2n - 1 + n\lceil \log n \rceil$ bits. (Hint: Use $2n - 1$ bits to specify the structure of the tree, as discovered by a walk of the tree.)

   Solution:

   First, observe that any full binary tree has exactly $2n - 1$ nodes. We can encode the structure of our full binary tree by performing a preorder traversal of T. For each node that we record in the traversal, write a 0 if it is an internal node and a 1 if it is a leaf node. Since we know the tree to be full, this uniquely determines its structure. Next, note that we can encode any character of C in $\lceil \log n \rceil$ bits. Since there are n characters, we can encode them in order of appearance in our preorder traversal using $n\lceil \log n \rceil$ bits.

2. (15 pts.)
Generalize Huffman's algorithm to ternary codewords (i.e., codewords using the symbols 0, 1, and 2), and prove that it yields optimal ternary codes.

   Solution:

   Instead of grouping together the two with lowest frequency into pairs that have the smallest total frequency, we will group together the three with lowest frequency in order to have a final result that is a ternary tree. The analysis of optimality is almost identical to the binary case. We are placing the symbols of lowest frequency lower down in the final tree and so they will have longer codewords than the more frequently occurring symbols.

3. (20 pts.)
You are consulting for a trucking company that does a large amount of business shipping packages between New York and State College. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit of W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight w_i.The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after he made it to State College faster. At the moment the company is using a simple greedy algorithm for packing; they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometime sending off a truck that was less fill, and in this way allow the next few trucks to be better packed.

Prove that, for a given set of boxes with specified weights, the greedy the algorithm currently in use actually minimizes the number of trucks that are needed. Your proof should follow the type of analysis we used for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" of all other solutions.

Solution:

Let n boxes arrive in order $b_1, b_2, \ldots, b_n$. Each box b_i has weight w_i and the maximum capacity of each truck is W. We pack the boxes into N trucks while preserving the following conditions.

- No truck is overloaded: the total weight of all the boxes in each truck is at most W.
- The order of arrival is preserved: Box b_i is sent before box b_j iff b_i arrived before b_j.

We prove that the current greedy algorithm uses the fewest possible trucks by showing that it "stays ahead" of any other solution. Specifically we consider any other solution and show the following. If the greedy algorithm fits boxes $b_1, \ldots b_j$ into first k trucks and the other solution fits boxes $b_1, \ldots b_i$ into first k trucks then $i \leq j$. Note this implies the optimality of the greedy algorithm by setting k to be the total number of trucks used by the greedy algorithm.

We will prove this claim by induction on k. Base Case: for $k = 1$, this trivially holds as the first truck fits as many boxes as possible. Induction Step: Assume this holds for $k - 1$. Thus if the greedy algorithm fits j' boxes into the first $k-1$ trucks and the other algorithm fits i' boxes into the first $k - 1$ trucks then $i' \leq j'$. Now for the kth truck let the alternate solution packs $b_{i'+1}, \ldots, b_i$. Thus since $j' \geq i'$, the greedy algorithm is also able to fit at least all the boxes $b_{i'+1}, \ldots, b_i$ (that are not already carried by the $k - 1$th truck) into the kth truck and potentially it can fit more. This completes the induction argument and thus proving the optimality of the greedy algorithm.