ESO207A: Data Structures and Algorithms

Practice Set 7: *Greedy* –

Problem 1. A problem from automatic program analysis: For a set of variables $x_1, x_2, \ldots, x_n$, you are given some *equality* constraints, of the form "$x_i = x_j$" and some *not-equality* constraints, of the form "$x_i \neq x_j$". You have to answer whether it is possible to satisfy all of them? For instance, the constraints

$$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$$

cannot be satisfied. Given an efficient algorithm that takes as input m constraints over n variables and decides whether all the constraints can be satisfied.

Problem 2. *SJF* A server has n customers waiting to be served. The service time required by the ith customer is say t_i. The server serves customers one at a time, in some sequence. If a customer i is served after customers $j_1, j_2, \ldots, j_k$, then, the *waiting time* of customer i is the total service time of the customers served before it, namely, $t_{j_1} + t_{j_2} + \ldots + t_{j_k}$. We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i) \ .$$

Give an efficient algorithm (strategy) for computing the optimal order in which to process the customers. *Note:* The algorithm is obvious and intuitive. Do the correctness proof carefully.

Problem 3. *Ternary Huffman.* Suppose you can encode strings using sequence of 0,1, and 2, instead of just 0 and 1. Design a modified version of Huffman's algorithm to compress a sequence of characters from an alphabet of size n, where, the characters occur with frequencies $f_1, f_2, \ldots, f_n$. Your algorithm should encode each character with a variable length codeword over the values $0, 1, 2$, such that the coding is prefix-free, and obtains maximum possible compression. Analyze the complexity of your algorithm and prove its correctness.

Problem 4. In this problem, you will show that the approximation ratio derived for the greedy set cover algorithm is tight (up to constant factors) as follows. For every integer n that is a power of 2, create an instance of the set cover problem with the following properties.

1. There are n elements in the universe set.
2. The optimal cover uses two sets.
3. The greedy algorithm picks up $\approx \log n$ sets.