In lab exercises

1. Trace the execution of the weighted interval scheduling solution and confirm that it correctly computes the optimal solution. Also modify the algorithm to save enough state to reconstruct the solution set.

**Algorithm 1** WIS\textunderscore Sol(n):
\begin{verbatim}
If \( n = 0 \): return 0
elif \( M[n] \) is not null: return \( M[n] \)
else: \( M[n] = \max(v_n + \text{WIS}\textunderscore Sol(p(n)), \text{WIS}\textunderscore Sol(n - 1)) \)
return \( M[n] \)
\end{verbatim}

2. Consider the 0-1 Knapsack problem: Given \( n \) items, each with weight \( w_i \) and value \( v_i \), compute the largest subset of items which maximizes the total value, subject to the constraint that the total weight is less than \( W \). Give counter examples that show a greedy solution does not work if we greedily select on any of the following rankings: value per weight, heaviest first, lightest first, highest value first

3. Consider a \( p \times q \) matrix \( A \) and a \( q \times r \) matrix \( B \). The product \( C \) is a \( p \times r \) matrix that can be computed using \( pqr \) multiplication operations. Now consider a chain of matrices \( A_1A_2A_3\ldots A_n \). The total number of multiplications needed to compute this product depends on how we parenthesize the individual multiplications.

   (a) Let \( A_1 \) be \( 10 \times 100 \). Let \( A_2 \) be \( 100 \times 5 \). And let \( A_3 \) be \( 5 \times 50 \). How many multiplications are used computing \((A_1A_2)A_3)\)?

   (b) How many multiplications are used computing \((A_1(A_2A_3))\)?

   (c) For a sequence of \( n \) matrices, how many parenthesizations are there (roughly)?

   (d) Design an algorithm to compute the optimal parenthesization and analyze its runtime.

4. (Problem 6.8 in text) A swarm of robots arrives over the course of \( n \) seconds. In the \( i \)th second, \( x_i \) robots arrive. The sequence \( x_1, x_2, \ldots, x_n \) is known in advance from scouting reports. At your disposal is an electromagnetic pulse weapon, the EMP. If charged for \( j \) seconds, it can destroy up to \( f(j) \) robots in the vicinity. Then you must wait for the EMP to recharge. The function \( f(j) \) is also known from prior defense testing. The number of robots destroyed by discharging the pulse at time \( k \) is given by \( \min(x_k, f(j)) \) where \( j \) is the number of seconds since the last firing of the EMP. The algorithm below is used to defend against the robot invasion.

**Algorithm 2** RoboDefense(\( X \)):
\begin{verbatim}
Let \( j \) be the smallest value for which \( f(j) \geq x_n \)
(If no such \( j \) exists, set \( j = n \))
Activate the EMP in the \( n \)th second.
If \( n - j \geq 1 \):
    Recurse on \( x_1, x_2, \ldots, x_{n - j} \)
\end{verbatim}

   (a) The above “greedy” algorithm does not maximize the number of robots destroyed. Construct an example to show this.

   (b) Design an algorithm that does work.