1.

Greedyalgorithm()

For all jobs from j1 to jN

Dollars[ji] = di\*Ti

Sort(dollars)

For every value in dollars[ji]

If ti < Ti #calculated time limit < given time limit

Total = total + dollars[ji]

The time complexity for the for loop that calculates the earnings for each job is O(n)

Time complexity for sorting the values from greatest to least is O(nlogn)

The second for loop will also require a time of O(n)

Total time complexity will be O(nlogn) + O(n), or O(n2)

2.

Example 1

Matrix A: 40 x 20

Matrix B: 20 x 30

Matrix C: 30 x 10

Matrix D: 10 x30

Proof by contradiction

a) Cheapest Multiplication = A((BC)D) = 36,000

Most efficient = (A(BC))D = 26,000

b) Most expensive multiplication = ((AB)C)D = 48,000

Most efficient = (A(BC))D = 26,000

Example 2

Matrix A: 10 x 30

Matrix B: 30 x 70

Matrix C: 70 x 50

Matrix D: 50 x 20

Proof by contradiction

c) Multiplication between Mi and Mi + 1 such that the number of columns in Mi is minimized = A(B(CD)) = 118,000

Most efficient = (((AB)C)D) = 66,000

3.

Knapsack()

//Check if k or n is zero

If n==0 || k==0

Return 0

//Check the weight to see if it is larger than the total weight

If (W[n] > k)

Max value = Knapsack(k,n-1)

If (W[n] <= k)

//take both cases and see which one gives the maximum value

First\_case = Knapsack(k-W[n], n-1)

Second\_case = Knapsack(k,n-1)

Return max(first\_case, second\_case)

4.

LCS()

Array[m+1][n+1]

Int counter

Vector vect

For every value in i to m

For every value in j to n

If i or j = 0

Set array[i][j] = 0

Else if firststring[i-1] == secondstring[j-1]

Increment counter by 1

Array[i][j] = array[i-1][j-1] + 1

Add the found letter to vect

Else

Set array[i][j] = maximum of (array[i-1][j], array[i][j-1])

Return counter or vect depending on if you want the # of characters in the lcs or the lcs string itself