Andrew McCann

CS350

Assignment #2

1. a) θ(n2) brute force algorithm for finding maximal subarray sum.

**Input**: A one dimensional array, ‘array’, of length ‘n’.

**Output**: the sum of the contiguous subarray of numbers with the largest sum.

func maximal\_sum(array)

max\_subarray = 0

for i = 0 ... n-1

current\_subarray = 0

for j = i ... n-1

current\_subarray += array[j]

if current\_subarray > max\_subarray

max\_subarray = current\_subarray

return max\_subarray

Correctness:

b) θ(nlog n) algorithm for finding maximal subarray sum.

Notes: I use iteration over the array instead of a for loop. The syntax is more easily readable. Just know each value is the value represented within the array, not the index.

**Input**: A one dimensional array, ‘array’, of length ‘n’.

**Output**: the sum of the contiguous subarray of numbers with the largest sum.

func merge\_sum(array)

if n == 1

return array[0], array

mid = n//2

left\_array = array[:mid] # not inclusive

right\_array = array[mid:]

l\_max, l\_result\_array = merge\_sum(left\_array)

r\_max, r\_result\_array = merge\_sum(right\_array)

merged\_array = l\_result\_array + r\_result\_array #Concat

left\_sum = 0

left\_max = LOW\_VAL # Lowest value possible

for value in reversed(l\_result\_array) # note reversed

left\_sum += value

if left\_sum > left\_max

left\_max = left\_sum

right\_sum = 0

right\_max = LOW\_VAL # Lowest value possible

for value in r\_result\_array # not reversed

right\_sum += value

if right\_sum > right\_max

right\_max = right\_sum

spanning\_total = right\_max + left\_max

left\_right\_max = max(l\_max, r\_max)

real\_max = max(left\_right\_max, spanning\_total)

return real\_max, merged\_array