**Group Assignment 1**

Jing Huang

933-039-277

Xinwei Li

933-666-663

# Presudo-Code

## Enumeration

**ENUMERATION(**a[1,...,n]**)**

define a result variable

for i = 1:n

for j = i:n

define a current variable

for k = j:n

add two numbers together and store it in current

take the max of current and result and store it in result

return max sum found so far

## Better Enumeration

**BETTER\_ENUMERATION**(a[1, ..., n])

for i = 1:n

sum <- 0

for j = i:n

current += a[j]

if sum < current OR i <- 0

sum <- current

keep max sum found so far

return max sum found so far

## Dynamic Programming

**DYNAMIC(**a[1,...,n]**)**

define a variable named current

define a variable named result

for i = 1:n

current += a[i]

if current less than 0 then set current to 0

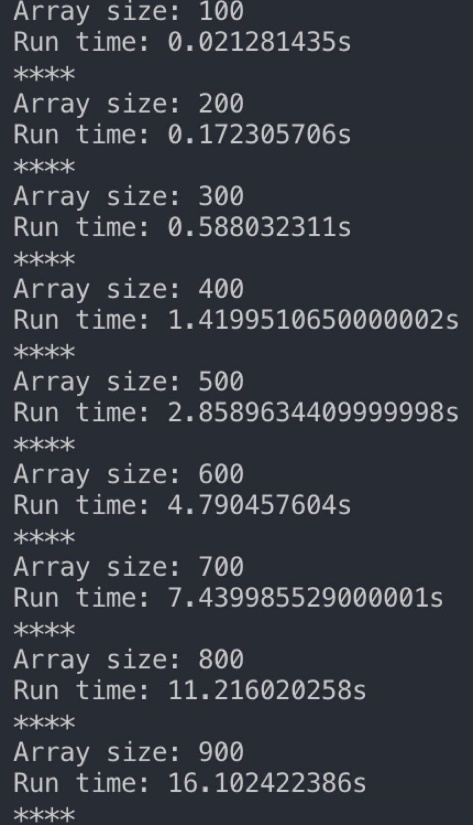
take the max of current and result and store it in result

return max sum found so far

# Run-time Analysis

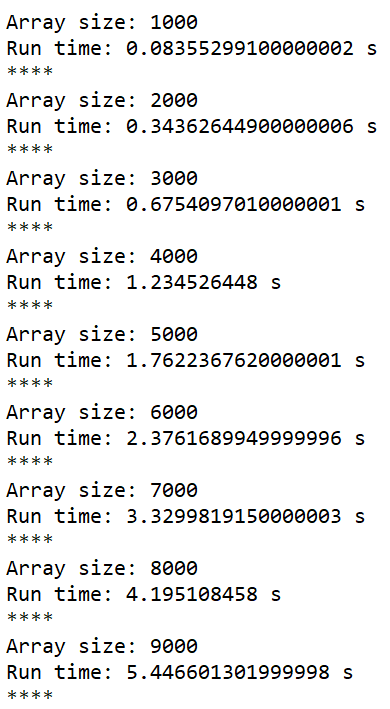
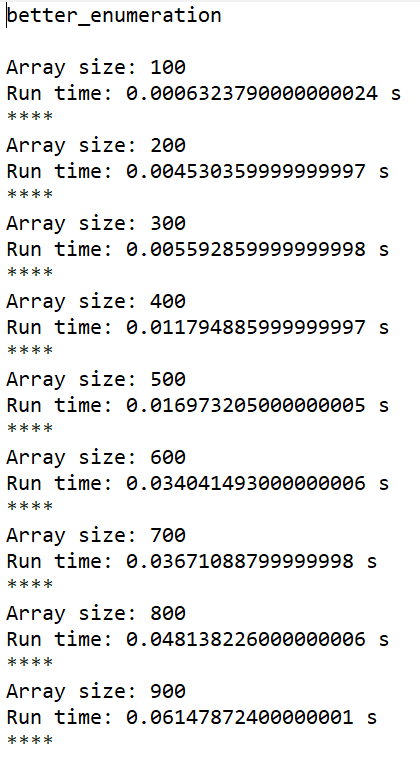
## Enumeration

The run-time of Enumeration Algorithm is O(n^3), and the sum of it is , which means that running every size of input array is much slower, the output of runtime is below. As the picture shows that this algorithm is not an optimized one, it increases faster when it comes to 400 array size, which is nearly three times of 300 array size.



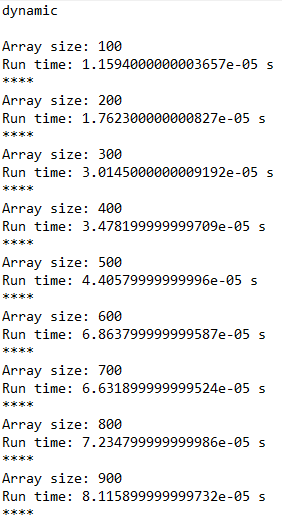
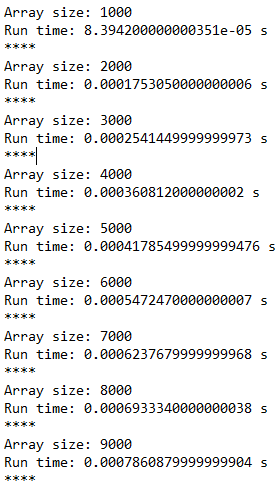
## Better Enumeration

The run-time of Better Enumeration Algorithm is O(n^2). It means that running every size of input array is slower than O(n), which is the best run-time of maximum subarray, and the sum of it is , the output of runtime is below. As the picture shows, run time increases rapid when it comes to 4000 array size, which is nearly twice of 3000 array size.



## Dynamic Programming

The run-time of Dynamic Programming Algorithm is O(n), which means that this is the best solution for maximum array problem among these three algorithms, and the sum of it is The output of runtime is below, as the picture shows that it increases gradually, and the time is much better than the Better Enumeration Algorithm.

# Experimental Run-time Analysis

We can know from the plot that the actual run-time of our implementation is similar to the theoretical run-time. And the most efficient algorithm implementation is the dynamic one, the second is the better enumeration, and the last one is the enumeration. There are some deviations because the compiler needs extra cost for processing.