

# Maximum Subarray Problem

<http://software.intel.com/fr-fr/articles/Acceler8EN/>

## Problem :

Given a 2-dimensional array of natural integers (between -32000 and +32000), the Maximum Subarray Problem asks for the rectangular area within the array that maximizes the sum of the array elements found in the area.

## Input-output :

Input format : Natural integers (between -32000 and +32000) separated by spaces, ended by a line break. All the lines have the same number of natural integers.

Output format : 4 integers delimiting the rectangle (x0 y0 x1 y1), separated by a space, ended by a line break. Array index starts at 0.

Example input 1 :

```
-10 -5 2 1 6  
-5 10 50 60 -9
```

Example corresponding output 1 :

```
1 1 3 1
```

Because the maximum subarray, highlighted in red, is in our case :

```
-10 -5 2 1 6  
-5 10 50 60 -9
```

If the software is given **several files as input**, the output should be the results of each file, separated by a line break, with output lines respecting the order of the input files.

## Build and run automated procedure :

The build and benchmark procedure is **automated**.

You will deliver C/C++/Fortran/Java source **code**, ready for compilation with a Makefile, in a zip file called "solution.zip" and **password protected (\*) with the password "secret"**. Binaries are not accepted.

We will unzip the file with your password, clean the build environment, build the binary/bytecode from your source and run it with a variable number of worker threads (in the following example 40) and a variable set of input files. Our script is very simple :

```
unzip -P secret solution.zip  
make clean  
make  
time ./run 40 input1.txt input2.txt input3.txt > output.txt
```

We expect you to have everything ready in your solution to be built with "make", and run with "./run". We will then check the output for correctness, and keep the time result as benchmark.

## Reference machine :

The **reference** machine for building and benchmarking is the **ManyCore Testing Lab**. The build environment and libraries available are described in the attached "Getting Started Guide.pdf". It's a typical 64bit Linux OS with gcc and Intel tools installed, easy to replicate at home : You don't have to work on the MTL to participate, but it is our reference system. You can develop and benchmark on the MTL, but keep the original copy of your code outside the ManyCore Testing Lab.

**Advice** : Design your algorithm and code to make it **scale** on **a very large number of cores**.

*\* The zip file password helps your zip file go through email servers antiviruses without being blocked.*