CSE3081-2: Design and Analysis of Algorithms (Fall 2019)
Machine Problem 1: Maximum Sum Subrectangle in a 2D array

Handed out: September 17, Due: October 8, 11:59PM (KST)

1. Goal

The goal of this MP is to understand how different algorithms perform differently while producing the same result for the same problem.

2. Problem Description

You are given a 2D array of integers, such as the following.

1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

Your goal is to find a **subrectangle**, which has the largest sum. For example, sum of the following subrectangle (colored in gray) is 3.

1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

Sum of the following subrectangle is 23.

1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

For this problem, the subrectangle with the largest sum is the following, in which the sum is 29.

1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

In this MP, you should write the program that finds this largest sum. (You do not need to find the position of the subrectangle.)

There is a special case which you need to consider when implementing algorithms.

* If all numbers are negative, the maximum sum subrectangle will be a single largest number. Below is the example. The red region is the answer, and the largest sum is -1.

-2	-5	-9	-3
-7	-8	-6	-5
-12	-3	-1	-7
-4	-8	-13	-19

3. Your task and requirements (Read Carefully!)

(1) You should write a **C/C++** program which takes an array of integers (contained in a file) as input and finds the maximum sum subrectangle.

(2) You should also write a **Makefile**. The TA will build your code by running 'make'. It should create the binary file.

(3) When created, your binary file should be named **mp1_20180001**. The numbers should be your student ID. There should be only a single binary file. It is up to you to make a single or multiple source code files.

(4) Your code should build and run on a Linux machine. Even if you write your code using Microsoft Visual Studio on Windows, your code should compile on Linux unless you use Windows-specific API. Still, you should check and make sure it runs on a Linux machine.

(5) Your program should take two command-line arguments: The first one is the **input file name**, and the second one is the **algorithm index**. An example run is:

```
$ ./mp1_20180001 input00001.txt 2
```

(6) There are two different types of algorithms. Their indices are:

1 – $O(n^6)$ algorithm -- (discussed in class)
2 – $O(n^4)$ algorithm
3 – $O(n^3)$ algorithm

(7) ***IMPORTANT*** The input file format is as follows. Consider the following example.

```
4 5
1 2 -1 -4 -20
-8 -3 4 2 1
3 8 10 1 3
-4 -1 1 7 -6
```

- In the first line, there are always two numbers: **# of rows and # of columns**.
- From the second line, each line has **n** numbers, where **n** is the # of columns.
- Excluding the first line, there are **m** lines, where **m** is the # of rows.
- In each line, numbers are separated by a single space.

(8) You program should produce an output file. The name of the output file must be "result_**inputfilename**". For example, if the input file name is "input00001.txt", the corresponding output file should be named "result_input00001.txt". The output file should have 6 lines, containing the following items:

1st line: input file name
2nd line: algorithm index
3rd line: # of rows in the given 2D array.
4th line: # of columns in the given 2D array.
5th line: sum of the maximum sum subrectangle.
6th line: running time in milliseconds

To measure running time, you can use functions such as `clock()`.

In the above example, the sum of the maximum sum subrectangle was 29, so assuming the input file name is "input00001.txt" and the algorithm index is 3, your output file should look like this:

```
input00001.txt
3
4
5
29
2.5
```

3. Report

In addition to code files, you should also submit a report document. In the document, **you should describe how the running time of each algorithm grows as the input size becomes larger**.

You can use tables, graphs, or any other visualization methods so that the reader could clearly understand the performance difference of the algorithms.

Your document does not need to be long, but you should definitely include the following:

- Experiment environment: The hardware specification of the machine where you did your experiments, such as CPU speed, memory size, and OS type and version. Obviously, you should do all your experiments in one machine for fair comparison.

- Experiment setup: This is basically how you chose parameters for your experiment. What is the metric that you measured, and what is the range of input size, etc.

- Your comments on the experience: This is what you have observed from doing the experiments. (For example, you learned how the algorithms will perform when the input size becomes large. Can you see the trend for yourself? What if the input size is small?) Just write anything you observed, regardless of whether you can explain the phenomena or not.

4. Submission

You should submit the Makefile, the source code(s), and the report document. Make the file into a zip file named cse3081_mp1_**20180001**.zip. The numbers should be your student ID. Where to submit your file will be announced by TA at the cyber campus.

5. Evaluation Criteria

(1) Your implementation: 60%

- Does your implementation produce correct results for all three algorithms?
- Do you follow the requirements faithfully? (such as Makefile and output format)

(2) Comments and coding style: 10%

- Is the code organized and easy to read? (indentation, spaces, styles)
- Do variable names reflect what they really are?
- Are there enough comments that explains what the code blocks are doing?

(3) Your report: 30%

- Can the reader understand the object and result of your experiments?
- Can the reader see the result visually?

6. Notes

- You should write your own code. You can discuss ideas with other students, but definitely should not copy their work. You can also find ideas and source codes on the Internet. My suggestion is that you first try to figure out the algorithm yourself, and then look up on the Internet if you find it difficult. Do not copy the source code from the Internet.

- As announced in the first class, duplicates will receive zero grade.

- 10% of the evaluation goes to the practice of writing the code. Your code should not just work but should also look good to other programmers. There are many books and materials on code writing practice. One example is a book called “Code Complete” by Steve McConnell.

- You may write your program in your own environment (Windows, Linux, MAC). But you should test your code on the Linux machine before submitting the file. Each of you will be given an account to the department Linux server.

- Remember that the TA will place your files in a directory, build your code using ‘make’, and run the code with the test inputs. Make sure everything works before submitting your work.

- Do NOT submit binary files. Submit only the files listed in the Submission section.

7. Late Policy

10% of the score is deducted for each day, up to five days. Submissions are accepted up to five days after the deadline.