COMP3010 - Algorithm Theory and Design

Assignment 1 - Allocating Tasks on a Supercomputing

```
Worth: 20%  overall unit
Due: 11.55pm Sunday 3 September on iLearn.
```

*Objectives: the purpose of this assignment is to test your ability to design **efficient** advanced algorithms, analyse them, and implement them in Java*. Your programs MUST work correctly in JAVA (Java 16 or above). This assignment also tests your ability to understand standard specifications and follow instructions carefully. Failing to submit the assignment as required will cause you to fail the assignment.

Overview

The University is a member of the National Computational Infrastructure (NCI) using the nation's most highly integrated high-performance research computing environment, including NCI's supercomputer which caters to the biggest, most highly parallel workloads of Australia's largest research organisations. The University has hired you to manage the jobs allocated to the Supercomputer and make sure it is effectively used. The project is to allocate all possible computationally intensive tasks that can be done in batch off-line to the supercomputer. However distributing the tasks fairly among the Computing processors in the supercomputer appears challenging: the tasks often have different size and they should be distributed as evenly as possible to guarantee that they will be completed as quickly as possible. Also, and more importantly, as some tasks are related, these should be run on the same or close processors to avoid large amount of data being transfered over the networks. You have been hired to write a program that will automatically allocate the tasks to each processor of the supercomputer.

For this assignment, you are given a simpler version of this problem: there are p processors (although p can be arbitrarily large) and there are NxM tasks to allocate. The load of these tasks are given as a NxM matrix (also arbitrarily large) of non-negative integers. However, you cannot make any other assumptions on the properties of these numbers (i.e., you CANNOT assume a maximum number or a maximum size of the matrix dimension). You must allocate each task with its complete load to one of the processors while **minimising the largest total load that any processor will receive** (as you can assume that processors are identical and a unit of task takes the same time on any processor). The constraints are:
- Each task cannot be shared accross processors (i.e., the whole load value of one matrix cell must be allocated to one processor).
- To respect the relationship between tasks, all tasks allocated to one PC must be **a sub-rectangle of the matrix**.

Your Work.

For the following problem, your program must ask the user (in this particular order, respecting the format) for:

1. the number p of processors that will receive Task Loads (i.e., an integer followed by end-of-line), then
2. the number r of rows and the number c of columns of the matrix of loads (i.e., two integers followed by end-of-line), then
3. on each of the following r lines, the c values of loads.

Your task is to find a solution that minimise the largest total load that any processor will receive by allocating a sub-rectangle of the matrix to each of the p processors. You must output:

1. On the first line: the largest sum of load allocated to a processor, then
2. on the next p lines: the coordinates in the matrix of the sub-rectangles allocated by your program to each processor in the order (top row, left column, bottom row, right column) (i.e., a list of four integers, in this particular order, followed by an end-of-line. Please note that the first row and the first column must be labelled 0, respectively, hence the last row (resp. column) is r-1 (resp. c-1).

Be careful not to output anything else (e.g., debugging comments,...). In any case, it is paramount (for automarking purpose, hence assessment validity) that the last p+1 lines are in the order and format expected. To make sure that you follow the specifications, you are provided with a skeleton java file, *ass1_comp3010.java*, with expected input and output.

You can assume that the user will not try to crash your program by entering invalid data (negative values,...). You can also assume that the number of processors is at least 2, and that the number of rows and the number of columns are both strictly larger than the number of processors (otherwise it is trivial). You must at least submit a main program in a file called *ass1_comp3010.java*. However, you can create as many other functions and create other files if needed (make sure you submit them).

I will NOT be running your program in Eclipse or any other IDE, but use a terminal interface: I will be compiling your Java code, and running it using a script. That is, I will be compiling it (*javac ass1_comp3010.java*) and running it using *java ass1_comp3010*. So test your program accordingly:

```
1. open a terminal window, go to your assignment folder.
2. run the command: javac ass1_comp3010.java
3. run the command: java ass1_comp3010
```

Please note that, as your program will be also tested automatically you must comply to the specifications. In particular, pay attention to the required order of the output (use the skeleton program given). (Eg, you cannot just print an Arraylist.)

An example executable solution, *ass1_sample* will be shown in the Lectures to help you understand the input/output requirements. However, please note that, as you will see clearly with the first example, this solution is not optimal (i.e., does not does not select the sub-rectangles with minimum overall load as it could be done by splitting vertically allocating the first 2 columns to one processor and the last column to the other). In fact, it will often give you a solution far from the minimum load. So do not believe in the quality of your program is optimal if it gives similar or close answers. You can create as many other functions/methods/helpers/... and create other files if needed (make sure you submit them). Here are few sample executions (which valid but is possibly non-optimal):

```
> java ass1_sample
Enter the number of processors that need to share the load:
2
Enter the number of rows of the matrix of the loads:
3
Enter the number of columns of the matrix of the loads:
3
Enter the load matrix (row by row, left to right):
1 2 3
1 2 3
1 2 3

The maximum load (first line) followed by
the 2 regions selected are (top row, left column, bottom row, right column):
12
0 0 1 2
2 0 2 2

------------------
> java ass1_sample
Enter the number of processors that need to share the load:
2
Enter the number of rows of the matrix of the loads:
4
Enter the number of columns of the matrix of the loads:
4
Enter the load matrix (row by row, left to right):
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

The maximum load (first line) followed by
the 2 regions selected are (top row, left column, bottom row, right column):
8
0 0 1 3
2 0 3 3

------------------
>java ass1_sample
Enter the number of processors that need to share the load:
3
Enter the number of rows of the matrix of the loads:
4
Enter the number of columns of the matrix of the loads:
4
Enter the load matrix (row by row, left to right):
1 1 1 1
1 1 1 1
1 1 1 1
1 120 1 1

The maximum load (first line) followed by
the 3 regions selected are (top row, left column, bottom row, right column):
122
0 0 2 3
3 0 3 0
3 1 3 3

------------------
> java ass1_sample
Enter the number of processors that need to share the load:
4
Enter the number of rows of the matrix of the loads:
5
Enter the number of columns of the matrix of the loads:
5
Enter the load matrix (row by row, left to right):
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 120 1 1

The maximum load (first line) followed by
the 4 regions selected are (top row, left column, bottom row, right column):
122
0 0 3 4
4 0 4 0
4 1 4 1
4 2 4 4

------------------
>java ass1_sample
Enter the number of processors that need to share the load:
2
Enter the number of rows of the matrix of the loads:
3
Enter the number of columns of the matrix of the loads:
3
Enter the load matrix (row by row, left to right):
1 120 1
1 1 1
1 1 1

The maximum load (first line) followed by
the 2 regions selected are (top row, left column, bottom row, right column):
122
0 0 0 2
1 0 2 2
```

Your task:

1. Submit a file (e.g., *ass1_comp3010.java* and other separate files if necessary - no zip files) that use an EFFICIENT algorithm to provide the best output as possible. By efficient, we mean that your program must use a low complexity (both in time and in memory-size) AND that the largest total load that any processor will receive is as small as possible. ALL programs should be clearly presented (e.g., including invariants, pre- and post-conditions when needed) with appropriate style, presentation, clarity, lack of typos.
2. You must submit a report/portfolio/blog (word or pdf are both acceptable) summarising the different steps of your attempts (what works and what did not), and reflecting on the things you learnt along the way. It must include one page describing the efficiency analysis of your program based on the input (it is up to you to determine what the important input here) using asymptotic notations (at least big-Oh notation, and, if appropriate, big-Omega and big-Theta). All portfolios/reports do not need to be long, 2 pages maximum, but they must be clear on where you are at.

Your Work.

You must work on your own and must write your program **ass1_comp3010.java** (14%) and analysis portfolio (6%).

Warning and Hints

- You **cannot discuss** specific approaches and **cannot show** algorithms and code to others. (See study guide regarding plagiarism.)
- If you re-use any part of code which has been presented in the web or is in a text book, make sure you cite the reference when you do so.
- For each input, there may be several optimal solutions.
- For finding the best "minimal" that any processor will receive, the quality of your solution will be compared to the best solution find by one of you.
- If your solution is too far from the minimum and your algorithm is inefficient, you will end-up with less than 50%.

Marking Scheme -

1. CORRECTNESS - 10/14 (Automatic testing.) The correctness and quality of your solution. For each test, if correct, you will get marked based on how far you are from the best solution find by others. However, there will be a time-limit for each test and your program will be killed (and hence failed the test) if it takes too long (i.e., the complexity of the algorithm is too high).
2. EFFICIENCY - 4/14 (Hand marking.)- The complexity effectiveness of the solutions and methods used. The effectiveness here is defined as the **lowest asymptotic complexity in the worst case analysis** of your solution. For example, it is easy to design a program that will try all possible combinations but the analysis should show you that the complexity is too high and you'll get 0 for effectiveness and risk to get your program killed at the execution.
3. analysis portfolio - 6/20 (Hand Marking).
4. **Failing** to write the efficiency analysis using asymptotic notation will cause you to fail the assignment (maximum mark will be 8/20).

Your program (with other files if needed) and portfolio MUST be submitted electronically using iLearn as requested (NO paper or email will be accepted). That is, for the programming part, you must submit to iLearn at least one Java file, that is: *ass1_comp3010.java* compiling correctly including the corresponding main program.

Trials

I will run weekly trials for all submitted attempts Mondays 21 and 28 August, 9am. Submit early version by Sunday night.

Frequently Asked Questions (FAQ)

I'll attempt to answer any questions about the assignment sent to me by email within 1 or 2 open days. Although if your question is genuinely of interest, an answer will be posted in the relevant iLearn discussion page by the next working day, so that everyone can benefit from it. So please, read the FAQ before asking. **You are expected to read the FAQ regularly (and at least before the submission deadline).** This assignment also tests your ability to understand standard specifications and follow instructions carefully. Failing to submit the assignment as required might cause you to fail the assignment.