Dear ODTÜClass Users,
You can access/login ODTUClass 2023-2024 Spring semester at https://odtuclass2023s.metu.
If you want to access the page directly by typing **odtuclass.metu.edu.tr**, **please clear the cache** i
Best regards,
ODTÜClass Support Team

[CENG 315 ALL Sections] Algorithms

Description Subr

THE4

Available from: Saturday, November 25, 2023, 12:00 PM
Due date: Sunday, November 26, 2023, 11:59 PM
Requested files: the4.cpp, test.cpp, the4.h (Download)
Type of work: Individual work

Test I/O used in VPL is available here.

You decided to start a new life on a deserted island with your friend Isabelle. Your goal is to fill th predetermined plot dimensions. For this, you need to divide the land into rectangular plots with t least possible m^2 of unusable land.

Isabelle tells you the *dimensions* of the island. She also tells you the dimensions of the rectangular into. Any piece of land (the whole island or the already divided plots of the land) can be divided **ei** two rectangular plots with integer dimensions, **dividing completely through that land.** This is the divided **plots cannot be joined together**. Since the buildings to be placed in these plots cannot b **plots also cannot be rotated**. You can **create zero or more plots with each given dimension**. A of any of the desired dimensions after all dividing operations are completed.

Problem

In this exam, you are asked to calculate the **most efficient way of dividing the land** given the **dir** the **dimensions of the plots in a 2D array of booleans**, then **return the *m^2 of unusable land*** b

function defined below.

```
int divide_land(int X, int Y, bool** possible_plots);
```

- ***X, Y*:** dimensions X x Y of the total rectangular land of the island
- ***possible_plots*:** a 2D array of booleans, where each value stands for the existence of a possib cell as dimensions (for example, if possible_plot[2][3] == true, then 2 x 3 is a possible plot dim
- **to return:** m^2 of unusable land, meaning the land that does not belong to any plot after all th

Constraints and Hints:

- ***X* and *Y* are integers <= 600**
- ***possible_plots* is a 2D array with dimensions (X+1)x(Y+1). So, the maximum dimensions ai** will be at most 200 "true" cells in the 2D array - but the complexity of the optimal solution does
- You are expected to use a **dynamic programming approach** to reduce complexity.

Evaluation:

- After your exam, black-box evaluation will be carried out. You will get full points if you return the the cases that will be tested, and do not exceed the memory & time limits.

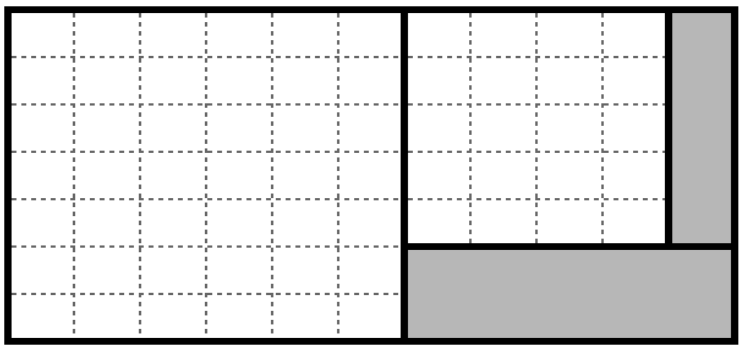
Example IO:

1)
Land Dimensions: X = 9, Y = 12
Possible Plots:
Plot 1: (1, 12) => possible_plot[1][12] = true;
Plot 2: (2, 6) => possible_plot[2][6] = true;
Plot 3: (3, 4) => possible_plot[3][4] = true;
Plot 4: (4, 3) => possible_plot[4][3] = true;
Plot 5: (6, 2) => possible_plot[6][2] = true;

m^2 of unused land: 0

2)
Land Dimensions: X = 11, Y = 7
Possible Plots:
Plot 1: (4, 5) => possible_plot[4][5] = true;
Plot 2: (6, 7) => possible_plot[6][7] = true;

m^2 of unused land: 15

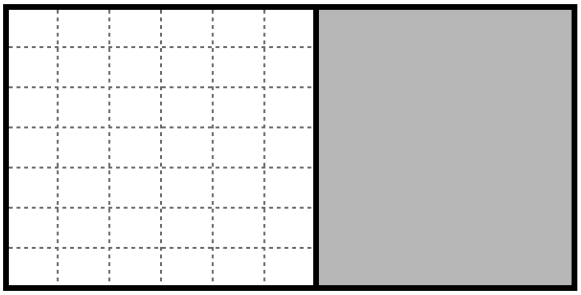


A visualized division for the second input, where the grey areas show unused land.

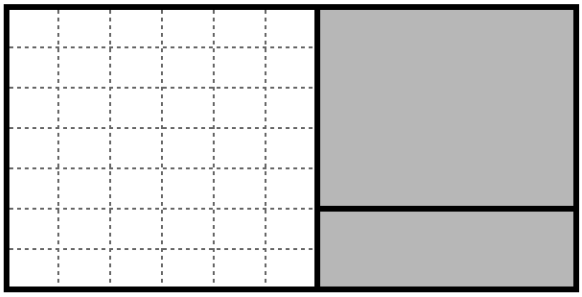
Step by step, the division goes as follows:

Step 0: the whole land is undivided and unused

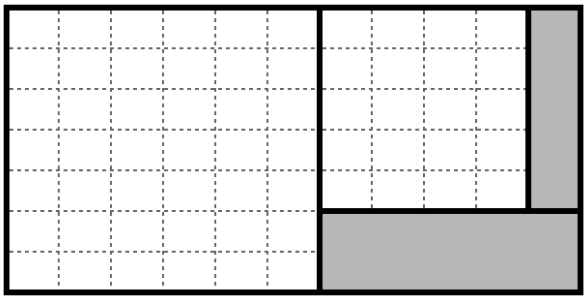
Step 1: The land is divided vertically, and the plot on the left with dimensions 6x7 is used, the rest of the land is unused.



Step 2: The unused land is divided horizontally, but since none of the pieces have the available plot dimensions, they are both unused.

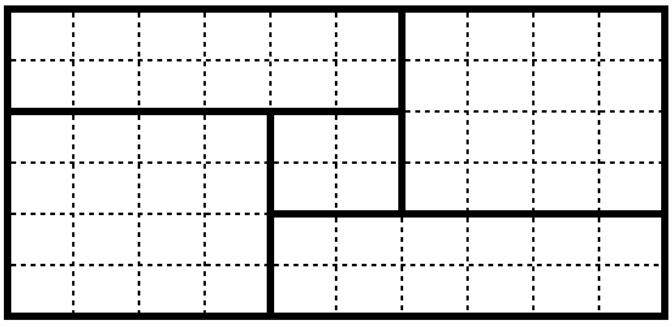


Step 3: The plot with 5x5 dimensions is divided vertically to create two plots with dimensions 4x5 and 1x5. Since 4x5 is an available plot, it is marked as used where as plot with dimensions 1x5 is left unused.



3)
Land Dimensions: X = 10, Y = 6
Possible Plots:
Plot 1: (2, 2) => possible_plot[2][2] = true;
Plot 2: (2, 6) => possible_plot[2][6] = true;
Plot 3: (4, 4) => possible_plot[4][4] = true;

Following is an **INVALID PLOT DIVISION** example:



Even though the plot sizes are valid, **the layout is impossible to reach when the land is divided completely each time.**

Specifications:

- There is 1 task to be solved in **36 hours** in this take-home exam.
- You will implement your solutions in ***the4.cpp*** file.
- You are free to add other functions to *the4.cpp*
- **Do not change** the first line of the4.cpp, which is #include "the4.h"
- Some libraries are included in "the4.h" for your convenience, you can use them freely.
- **Do not change** the arguments and the return value of the function ***divide_land()*** in the file the4
- **Do not include** any other library or write include anywhere in your the4.cpp file (not even in com
- You are given ***test.cpp*** file to test your work on **ODTUClass** or your **locale**. You can, and you are add different test cases.
- If you want to test your work and see your outputs you can compile your work on your locale as:

```
>g++ test.cpp the4.cpp -Wall -std=c++11 -o test

> ./test
```

- You can test your the4.cpp on the virtual lab environment. If you click **run**, your function will be **with test.cpp**. If you click **evaluate**, you will get **feedback** for your current work and your work a limited number of inputs.
- The grade you see in lab is not your final grade, **your code will be reevaluated with different i**

The system has the following limits:

- a maximum execution time of 16 seconds (your program needs to return in less than two second
- a 1 GB maximum memory limit,
- an execution file size of 4M.
- Solutions with longer running times will not be graded.
- If you are sure that your solution works in the expected complexity, but your evaluation fails due constant factors may be the problem.

Requested files

the4.cpp

```cpp
#include "the4.h"

// do not add extra libraries here

int divide_land(int X, int Y, bool** possible_plots){
    return X*Y;
}

```

test.cpp

```cpp
#include <iostream>
#include <fstream>
#include "the4.h"


void read_from_file(int& X, int& Y, bool**& possible_plots){
    int number_of_plots;
    char addr[]= "inp01.txt"; // 01-10 are available
    std::ifstream infile (addr);
    if (!infile.is_open()){
        std::cout << "File \'"<< addr
                  << "\' can not be opened. Make sure that this file exists." << std:
        return;
    }
    infile >> X;
    infile >> Y;
    infile >> number_of_plots;
    possible_plots = new bool*[X+1];
    for(int temp=0; temp < X+1; temp++) possible_plots[temp] = new bool[Y+1];
    for(int idx=0; idx < X+1; idx++) for(int idy=0; idy < Y+1; idy++) possible_plots[
    for(int temp=0; temp < number_of_plots; temp++){
        std::pair<int, int> plot;
        infile >> plot.first >> plot.second;
        possible_plots[plot.first][plot.second] = true;
    }
    infile.close();
}

int main(){
    int X, Y;
    bool** input_array;
    int minimum_unused_land, plot_number=1;

    read_from_file(X, Y, input_array);

    std::cout << "X: " << X << ", Y: " << Y << std::endl;
    for(int idx=0; idx < X+1; idx++)
        for(int idy=0; idy < Y+1; idy++)
            if(input_array[idx][idy])
                std::cout << "Plot " << plot_number++ << ": (" << idx << ", " << idy

    minimum_unused_land = divide_land(X, Y, input_array);

    std::cout << "Unused land: " << minimum_unused_land << " m^2" << std::endl;

    for(int idx=0; idx<X+1; idx++) delete[] input_array[idx];
    delete[] input_array;
    return 0;
}
```

the4.h

```h
#ifndef THE4_THE4_H
#define THE4_THE4_H
#include <vector>
#include <utility>
#include <algorithm>
#include <climits>

//updating this file will not change the execution in the VPL

int divide_land(int X, int Y, bool** possible_plots);

#endif //THE4_THE4_H
```