Case study 2022

Skills targeted:

- Structure, model and solve a concrete optimisation situation.
- Solve an optimisation problem using an algebraic language and a MIP solver.
- Formulate an implicit linear programming model with JuMP.
- Handle vectors and matrices with Julia.
- Present the optimisation results according a specified format.

Activities:

- Choose one of the two situations proposed to study and to solve.
- Write the linear programming model corresponding to this problem.
- Write the obtained model with JuMP.
- Write the all-in-one program which

```
1 brings together the data into an adequate datastructur
e,
2 states the optimisation model,
3 computes the optimal solution, and
4 reports the result with the specified format.
```

- Write a report presenting your productions on this case study
- Upload your report + julia code on Moodle

Situation 1: Guiding perseverance to discover Mars

Situation

While the real Perseverance rover is having fun on Mars, we imagine an alternative version that scouts out an $N \times N$ grid of Mars according to the following rules:

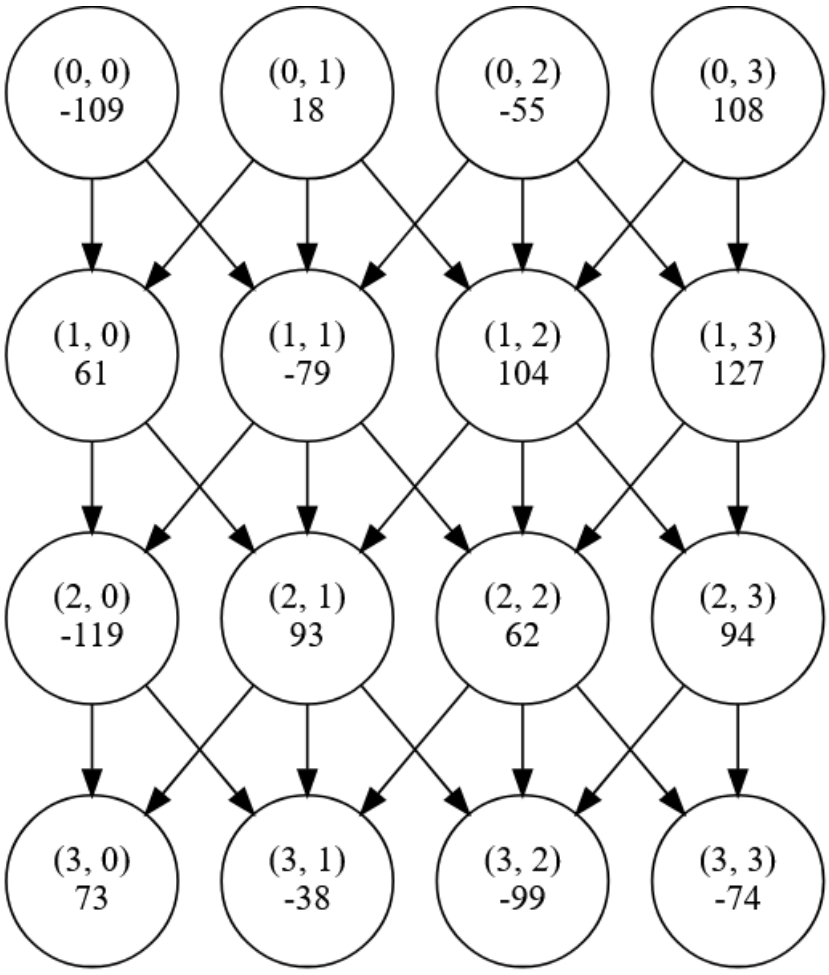
- Surveying a cell is possible only if all its upper neighbors were already explored. The upper neighbors of *(a,b)* are defined as *(a-1,b-1), (a-1,b), (a-1,b+1)*. Cells that are not on the $N \times N$ grid do not need to be surveyed first.
- Each cell has a "score" between *0-255* points, indicating how valuable it is to explore it.
- Exploring a cell also requires rover maintenance, equivalent to a "cost" of *128* points.

The goal of the rover is to earn the maximum score possible from the grid. This means choosing which cells to explore that satisfy condition 1, such that the total score gained, considering 2 and 3, is the maximum score possible.

We represent the grid as an $N \times N$ array of numbers given in hexadecimal format. As an example, consider the following 4×4 grid representation:

```
13 92 49 EC
BD 31 E8 FF
09 DD BE DE
C9 5A 1D 36
```

Which represents the following grid (the arrow $A \rightarrow B$ means "Exploring A is a prerequisite to exploring B"):



For example, the value *-109* in cell *(0,0)* is obtained by converting *13* in hexadecimal notion to *16+3=19* and subtracting *128*, obtaining *-109*. Similarly, the value *18* in *(0,1)* is obtained by converting *92* to *9*16+2=146* and subtracting *128*.

For the grid above, the optimal score is 424, and can be achieved via the following set:

$$[(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3)]$$

Question

Find the maximum score and a set of cells achieving it for the following *20×20* grid:

```
                BC E6 56 29 99 95 AE 27 9F 89 88 8F BC B4
2A 71 44 7F AF 96
                72 57 13 DD 08 44 9E A0 13 09 3F D5 AA 06
5E DB E1 EF 14 0B
                42 B8 F3 8E 58 F0 FA 7F 7C BD FF AF DB D9
13 3E 5D D4 30 FB
                60 CA B4 A1 73 E4 31 B5 B3 0C 85 DD 27 42
4F D0 11 09 28 39
                1B 40 7C B1 01 79 52 53 65 65 BE 0F 4A 43
CD D7 A6 FE 7F 51
                25 AB CC 20 F9 CC 7F 3B 4F 22 9C 72 F5 FE
F9 BF A5 58 1F C7
                EA B2 E4 F8 72 7B 80 A2 D7 C1 4F 46 D1 5E
FA AB 12 40 82 7E
                52 BF 4D 37 C6 5F 3D EF 56 11 D2 69 A4 02
0D 58 11 A7 9E 06
                F6 B2 60 AF 83 08 4E 11 71 27 60 6F 9E 0A
D3 19 20 F6 A3 40
                B7 26 1B 3A 18 FE E3 3C FB DA 7E 78 CA 49
F3 FE 14 86 53 E9
                1A 19 54 BD 1A 55 20 3B 59 42 8C 07 BA C5
27 A6 31 87 2A E2
                36 82 E0 14 B6 09 C9 F5 57 5B 16 1A FA 1C
8A B2 DB F2 41 52
                87 AC 9F CC 65 0A 4C 6F 87 FD 30 7D B4 FA
CB 6D 03 64 CD 19
                DC 22 FB B1 32 98 75 62 EF 1A 14 DC 5E 0A
A2 ED 12 B5 CA C0
                05 BE F3 1F CB B7 8A 8F 62 BA 11 12 A0 F6
79 FC 4D 97 74 4A
                3C B9 0A 92 5E 8A DD A6 09 FF 68 82 F2 EE
9F 17 D2 D5 5C 72
                76 CD 8D 05 61 BB 41 94 F9 FD 5C 72 71 21
54 3F 3B 32 E6 8F
                45 3F 00 43 BB 07 1D 85 FC E2 24 CE 76 2C
96 40 10 FB 64 88
                FB 89 D1 E3 81 0C E1 4C 37 B2 1D 60 40 D1
A5 2D 3B E4 85 87
                E5 D7 05 D7 7D 9C C9 F5 70 0B 17 7B EF 18
83 46 79 0D 49 59
```

Supply your solution as a number and a list of cells, given in the exact format above of a list of tuples (between the opening and closing brackets, each cell is listed as *(a,b)*, with *a,b* as comma-separated integers).

This problem has been proposed by IBM (https://www.research.ibm.com/haifa/ponderthis/challenges/March2021.html (https://www.research.ibm.com/haifa/ponderthis/challenges/March2021.html)) as ``the Month's Challenge'' in March 2021.

Entrée [ ]:

Situation 2: Packing different rectangles in a minimum-area rectangle

Situation

Rectangle packing is a packing problem where the objective is to determine whether a given set of small rectangles can be placed inside a given large polygon, such that no two small rectangles overlap.

Several variants exist and we consider here the variant where the objective is to pack different rectangles in a minimum-area rectangle. In this variant, the small rectangles can have varying lengths and widths, and their orientation is fixed (they cannot be rotated). The goal is to pack them in an enclosing rectangle of minimum area, with no boundaries on the enclosing rectangle's width or height.

This problem has an important application in combining images into a single larger image. A web page that loads a single larger image often renders faster in the browser than the same page loading multiple small images, due to the overhead involved in requesting each image from the web server.

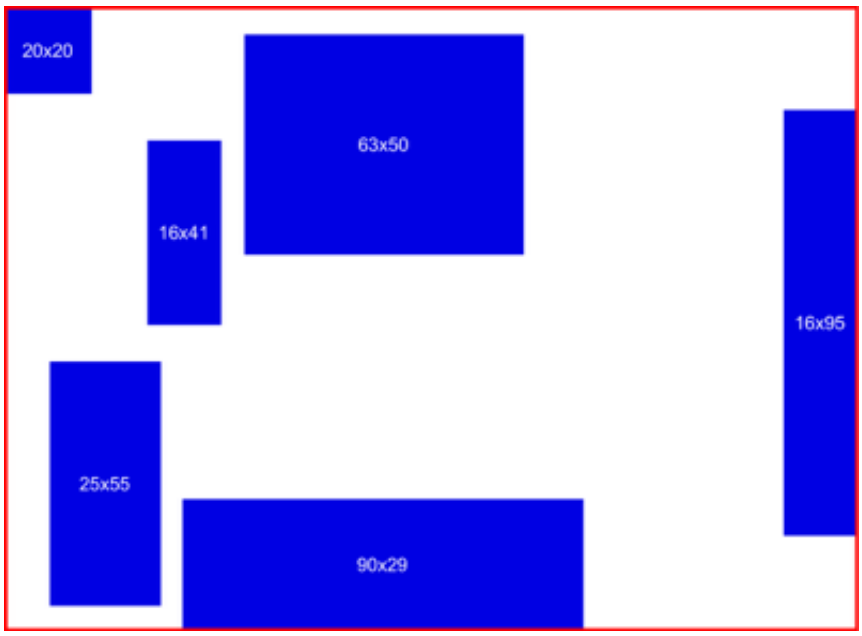
(Definition from *https://en.wikipedia.org/wiki/Rectangle_packing (https://en.wikipedia.org/wiki/Rectangle_packing)*)

Example of the application of this optimization problem for building CSS sprites:

(Image from [https://www.codeproject.com/Articles/210979/Fast-optimizing-rectangle-packing-algorithm-for-bu (https://www.codeproject.com/Articles/210979/Fast-optimizing-rectangle-packing-algorithm-for-bu)](https://www.codeproject.com/Articles/210979/Fast-optimizing-rectangle-packing-algorithm-for-bu))

Question

Find the minimum-area rectangle for the following rectangles:



with

```
w=[20,63,16,16,25,90]
h=[20,50,41,95,55,29]
```

Display automatically your optimal solution found using the plotting tools available in Julia.

Entrée [ ]: