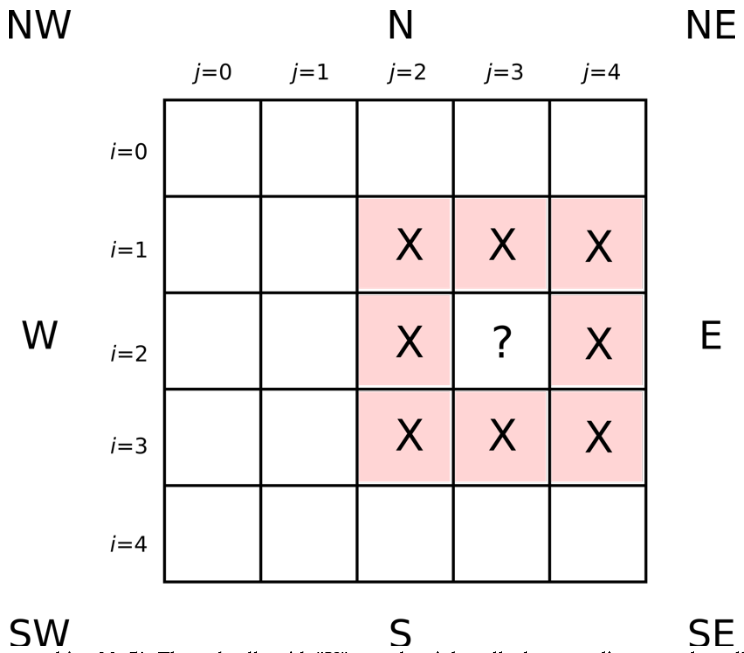
The Bushfire Model

We model a bushfire as occurring in a **square grid** made up of M by M cells. The **cell** $c_{i,j}$ refers to the cell in the ith row and the jth column of the grid. Each cell typically has eight **adjacent** cells (horizontally, vertically and diagonally). Cells on the edge of the grid are adjacent to only five other cells; cells in the corners of the grid are adjacent to only three other cells. For convenience, we take the top of the grid to be North, the right side to East, the bottom to be South and the left side to be West.



Bushfire grid with `lang:mathjax;M=5`. The red cells with "X"s are the eight cells that are adjacent to the cell `lang:mathjax;c2,3`.

Each cell in the M by M grid has several **attributes**:

- **fuel load**: the amount of combustible material (trees, leaf litter, etc) in that cell. The fuel load of a cell is represented as a non-negative integer, where 0 (zero) equates to no combustible material.
- **height**: the elevation of the cell. Height is represented as a positive integer (a cell with height 3 is higher than a cell with height 2, which is in turn higher than a cell with height 1, and so on). Fires will tend to spread more rapidly uphill, and more slowly downhill, compared to flat ground.
- **burning**: a Boolean value indicating whether the cell is currently burning or not. A cell can only be burning if it's fuel load is greater than 0 (zero).

In addition, the entire grid is characterised by two further attributes:

- **ignition threshold**: how combustible the landscape is (for instance, a dry landscape would be more combustible than a wet landscape, and have a lower ignition threshold). The ignition threshold is represented by an integer greater than 0 (zero). A non-burning cell will start burning if its **ignition factor** (explained below) is greater than or equal to the ignition threshold.
- **ignition factor**: the intensity of fire around a particular non-burning cell at a given point in time that, in combination with the **ignition threshold** described above, will be used to

determine whether a cell will start burning. The ignition factor is floating point number; details of how to calculate the ignition factor are provided on the following slides.

- **wind direction**: if a wind is blowing, it can carry embers that allow a fire to spread more rapidly in a particular direction. Wind may blow *from* one of eight directions: North, Northeast, East, Southeast, South, Southwest, West or Northwest (abbreviated N, NE, E, SE, S, SW, W, NW) or there may be no wind. How the wind affects ignition will be explained further on the following slides.

Question 1: Parsing a Bushfire Scenario File

Your task is to write a function **`parse_scenario(filename)`** that parses a file with the structure described below, validates the contents and returns either a dictionary containing all values required to specify a model scenario if the contents are valid, or `None` if any of the contents are invalid.
The structure of the file is as follows:

- an integer specifying the width and height (M) of the square landscape grid
- M lines, each containing M integer values (separated by commas), defining the initial fuel load of the landscape (a visual representation of the M by M grid)
- M lines, each containing M integer values (separated by commas), defining the height of the landscape (again, a visual representation of the M by M grid)
- an integer specifying the ignition threshold for this scenario
- a one- or two-character string specifying the wind direction for this scenario (or `'None'` if there is no wind)
- one or more lines, each containing the (i,j) coordinates (row number, column number) of a cell which is burning at the start of the simulation

For example, the file `'bf0.dat'`:

```
bf0.dat
2
2,2
0,2
1,2
1,2
1
N
0,0
```

specifies a 2 by 2 landscape:

- there is an initial fuel load of 2 in three of the four cells, with a fuel load of 0 in the cell c1,0
- the height of all cells in the first column (j=0) is 1, while the height of all cells in the second column (j=1) is 2 (i.e. the landscape slopes up toward the East)
- the ignition threshold is 1
- a wind is blowing from the North
- the top left cell c0,0 is burning at the start of the simulation

You may assume that the input files are well-formed, but your function should ensure that:

- the dimensions of the grid are a positive integer;
- the ignition threshold is a positive integer not greater than eight;
- the wind direction is valid; and
- the coordinates of the burning cells are (a) located on the landscape, and (b) have non-zero initial fuel load.

If all values are valid, your function should return a dictionary with key/value pairs as follows:

- **`f_grid`**: a list of lists (of dimensions M by M)
- **`h_grid`**: a list of lists (of dimensions M by M)
- **`i_threshold`**: an integer

- **`w_direction`**: a string or `None`
- **`burn_seeds`**: a list of tuples

If there is no wind, **`w_direction`** can be either an empty string `''` or `None`.
If any values are invalid, your function should return `None`.
For example:

```
>>> parse_scenario('bf0.dat')
{'f_grid': [[2, 2], [0, 2]],
 'h_grid': [[1, 2], [1, 2]],
 'i_threshold': 1,
 'w_direction': 'N',
 'burn_seeds': [(0, 0)]}
>>> parse_scenario('bf.dat')
{'f_grid': [[1, 2, 1], [0, 1, 2], [0, 0, 1]],
 'h_grid': [[1, 1, 1], [2, 1, 2], [3, 2, 2]],
 'i_threshold': 1,
 'w_direction': 'N',
 'burn_seeds': [(1, 1)]}
```

Updating the Model

The state of the bushfire model is defined by the attributes (fuel load, height, burning, ignition threshold and wind direction) described on the previous slide. Of these, fuel load and burning status may change over time as the bushfire spreads. Height, ignition threshold and wind direction are fixed and never change.

The state of the model is updated in discrete timesteps, t=0,1,2,3, …
At each timestep, the following things happen:

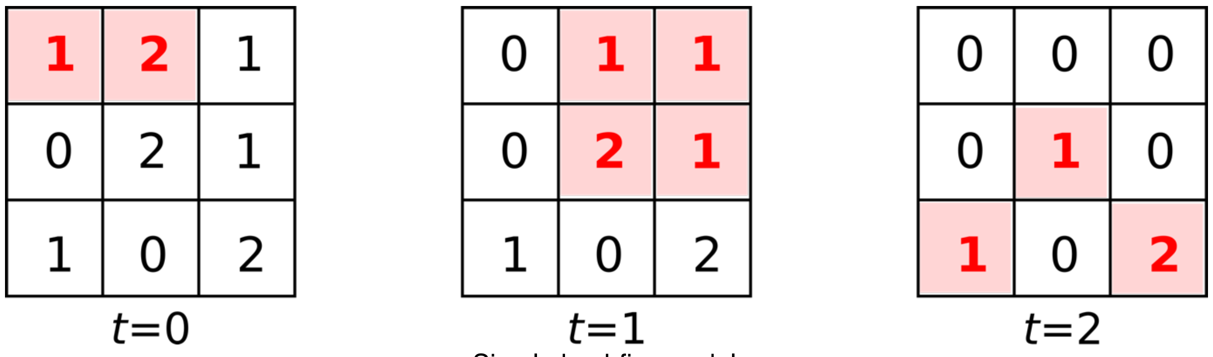
- If a cell is currently burning, it's fuel load will be reduced by one *in the following timestep*. If this will result in its fuel load reaching zero, it will stop burning *in the following timestep*.
- If a cell is not currently burning, it may start burning, depending on its proximity to other cells which are burning *in this timestep*. The rules describing how to determine if a cell catches fire is described on the next slide.

Note, the future state of each cell in time t+1 is determined on the basis of the current state of the grid at time t. We stop updating the model when no cells are burning, as the state will not change after this point.

Determining When a Cell Catches Fire

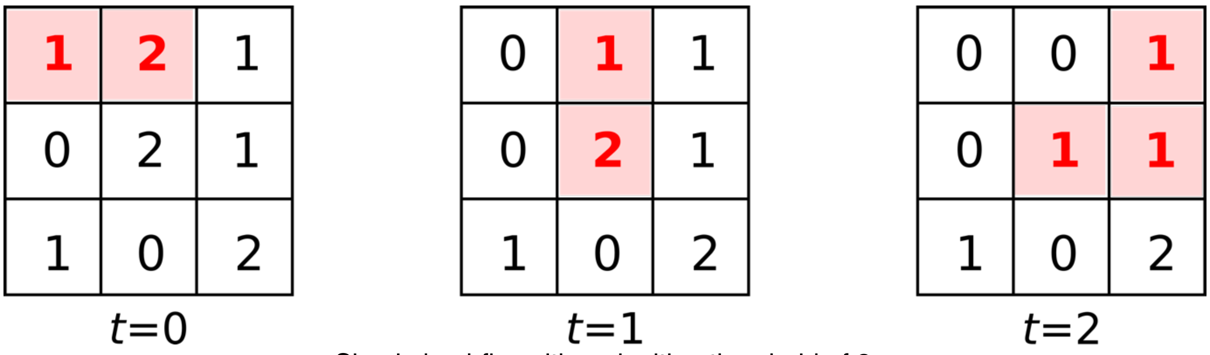
To determine whether a cell ci,j catches fire, we calculate its **ignition factor** and compare this to the landscape's **ignition threshold**. As stated above, a cell will catch fire if its ignition factor is greater than or equal to the ignition threshold. A cell must be currently not burning and have a fuel load greater than 0 (zero) in order to catch fire.

We will start by considering the simplest case: a flat landscape with no wind in which cell ci,j is not currently burning. In this case, each cell adjacent to ci,j that *is* burning contributes 1 point to ci,j's ignition factor. Thus, if the ignition threshold is 1, ci,j will start burning if it is adjacent to at least one burning cell. If the ignition threshold is 2, ci,j will start burning only if it is adjacent to at least two burning cells.



Simple bushfire model

Consider the above sequence of landscape states, in which all cells are of the same height, there is no wind, and the ignition threshold is 1. At the first timestep (t=0), the two cells c0,0 and c0,1 are burning. During this timestep, the fuel load of each of these two cells will decrease by one (causing the fire in c0,0 to stop burning in the next time step, t=1). If we then consider at the surrounding cells, c1,0 has a fuel load of 0 (zero), so it cannot catch fire. Cells c0,2, c1,1 and c1,2 each have a fuel load greater than 0 (zero) and are adjacent to one of the currently burning cells, therefore they will start burning in the next time step (t==1).



Simple bushfire with an ignition threshold of 2

This example is identical to that above, but the ignition threshold is now 2, meaning that each non-burning cell needs to be adjacent to two or more burning cells in order to start burning. Therefore, cells c0,2 and cells c1,2 will **not** start burning at t=1, as they were only adjacent to a single burning cell at t=0.
Only cells that are located within the bounds of the landscape grid can contribute to a cell's ignition factor. Thus, a cell located in the corner of a grid will, in this simple case, only have three adjacent cells that may cause it to start burning.

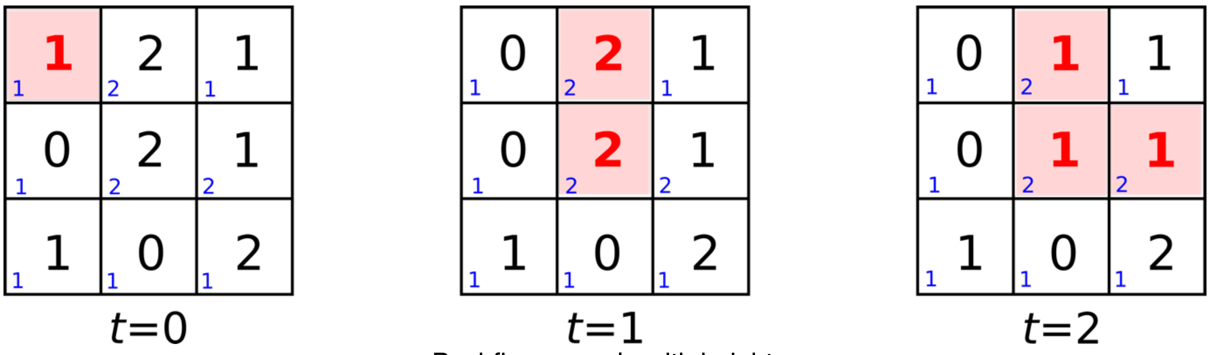
Determining When a Cell Catches Fire – Extended Model with Height

Height and wind modify can modify the calculation of the basic ignition factor described on the previous slide. The effect of landscape height is described on this slide, and the effect of wind on the following slide.

Height

On a flat landscape, where neighbouring cells are of the same height, each burning cell contributes 1 point to an adjacent non-burning cell's ignition factor.

However, bushfires tend to spread more rapidly uphill, therefore if a cell ci,j has height that is greater than that of an adjacent burning cell, that cell will contribute *twice* as much (ie, 2 points) to ci,j's ignition factor. Conversely, bushfires tend to spread more slowly downhill, therefore an adjacent burning cell with height greater than ci,j's height will contribute only *half* as much (ie, 0.5 points) to ci,j's ignition factor.



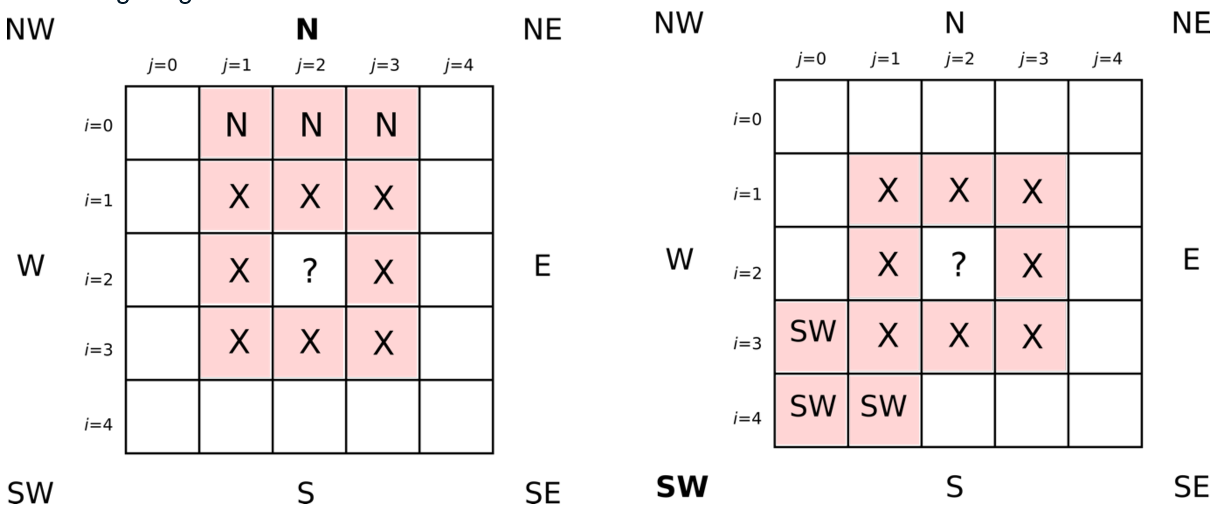
Bushfire example with height

In the above sequence, the height of each cell is indicated by a small blue number. No wind is blowing and the ignition threshold of the landscape is 2. At the first timestep (t=0) a single cell c0,0 is burning. The cell to the South of it (c1,0) has a fuel load of 0 (zero) and hence cannot catch fire. On a flat landscape, the other two adjacent cells (c0,1 and c1,1 would not catch fire either, as their ignition factor would only be 1 (from the single burning cell c0,0), below the ignition threshold of 2. However, in this landscape, cells c0,1 and c1,1 are higher than cell c0,0, therefore its contribution to their ignition factor is doubled to 2 points, high enough to equal the landscape's ignition threshold and cause them to start burning in the following timestep (t=1).

In constrast, the top-right cell c0,2 will escape being burnt in timestep t=2 (and beyond) as it is lower than the surrounding cells, hence they each contribute only 0.5 points to its ignition factor. Thus, even when all three surrounding cells are burning, c0,2's ignition factor is only 1.5, below the landscape ignition threshold of 2.

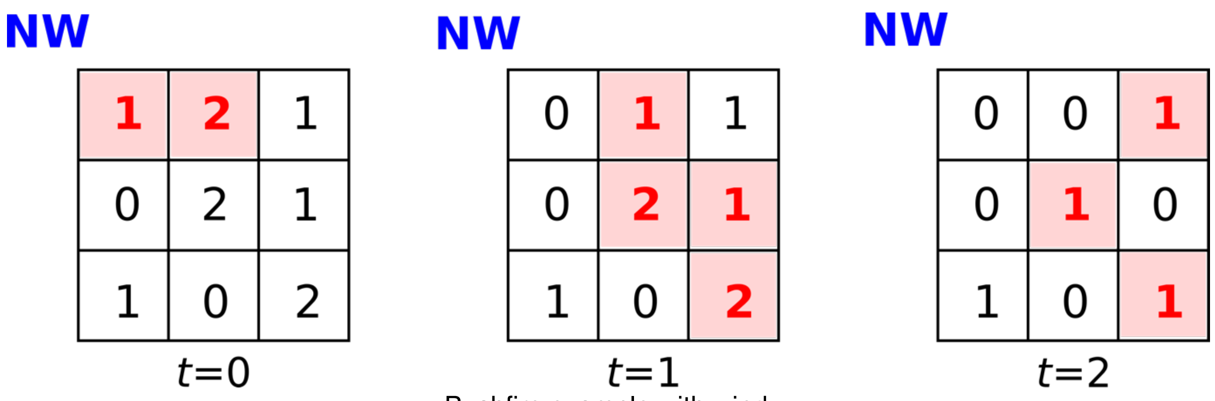
Determining When a Cell Catches Fire – Extended Model with Wind

Wind can carry burning embers that allow a fire to spread more rapidly in a particular direction. If a wind is blowing, up to three *additional* cells are considered to be adjacent to ci,j for the purpose of calculating its ignition factor.



Bushfire grid showing extra adjacent cells when wind is blowing from the North (left) or Southwest (right)

For example, if a wind is blowing from the North, the cell two cells *above* ci,j and the cells immediately to the left and right of this cell are considered adjacent to ci,j (ie, cells ci–2,j–1, ci–2,j and ci–2,j+1). If any of these additional cells are burning, they will also contribute when calculating ci,j's ignition factor. If a wind is blowing from the Southwest, the cell two cells *below and to the left* of ci,j and the cells immediately above and to the right of this cell are considered adjacent to ci,j. That is, cells ci+1,j–2, ci+2,j–2 and ci+2,j–1. Of course, these additional cells must be within the bounds of the landscape in order to have any effect, as in the simple case on the previous slide.



Bushfire example with wind

The sequence shown above is identical to the simple example shown on the previous slide (on a flat landscape, with ignition threshold of 2) except that the wind is now blowing from the Northwest. As a consequence, cell c0,0 is considered adjacent to c1,2, which therefore has an ignition factor of 2 (as c0,1 is also adjacent and burning) and hence will start burning at t=1. In addition, c0,0 and c0,1 are also both considered adjacent to c2,2, which will also start burning at t=1. Bushfires spread much more rapidly when the wind is blowing!

When considering the joint effects of height and wind, you should compare the heights of ci,j and each of its adjacent cells on a *pairwise* basis, disregarding the heights of any other surrounding cells. For example, if c0,0 was higher than c2,2 and c0,1 was lower than c2,2 then they would contribute 0.5 points and 2 points respectively to c2,2's ignition factor, irrespective of the heights of the intervening cells (c1,1, etc).

Question 2: Determine If a Cell Starts Burning

Based on the rules described earlier, your task is to write a function **`check_ignition(b_grid, f_grid, h_grid, i_threshold, w_direction, i, j)`** that takes as arguments the burning state **`b_grid`** (at time t), current fuel load **`f_grid`** (at time t), height **`h_grid`**, ignition threshold **`i_threshold`**, wind direction **`w_direction`** and coordinates **`i`** and **`j`** of a cell, and returns `True` if that cell will catch fire at time t+1 and `False` otherwise.

The arguments are of the following types:

- **`b_grid`**: a list of lists of Boolean values (of dimensions M by M)
- **`f_grid`**: a list of lists of integers (of dimensions M by M)
- **`h_grid`**: a list of lists of integers (of dimensions M by M)
- **`i_threshold`**: an integer
- **`w_direction`**: a string (if wind is blowing), otherwise `None` (if no wind is blowing)
- **`i`** and **`j`**: integers ($i<M$; $j<M$)

You may assume that all arguments are valid, as defined in Question 1.

For example:

```
>>> check_ignition([[True, False], [False, False]], [[2, 2], [2, 2]], [[1,
1], [1, 1]], 1, 'N', 0, 1)
True
>>> check_ignition([[True, False], [False, False]], [[2, 0], [2, 2]], [[1,
1], [1, 1]], 1, 'N', 1, 0)
True
>>> check_ignition([[True, True, False], [False, False, False], [False,
False, False]], [[1, 1, 1], [1, 1, 1], [1, 0, 0]], [[2, 2, 1], [2, 3, 1],
[1, 1, 1]], 1, None, 0, 2)
False
>>> check_ignition([[True, True, False], [False, False, False], [False,
False, False]], [[1, 1, 1], [1, 1, 1], [1, 0, 0]], [[2, 2, 1], [2, 3, 1],
[1, 1, 1]], 2, None, 1, 1)
True
```

Question 3: Run the Model

Your task is to write a function
`Run_model(f_grid, h_grid, i_threshold, w_direction, burn_seeds)`
that takes as arguments the initial fuel load **`f_grid`** (i.e. at time $t=0$), height **`h_grid`**, ignition threshold **`i_threshold`**, wind direction **`w_direction`** and a list of cells **`burn_seeds`** that are burning at time $t=0$, and returns a tuple containing (a) the final state of the landscape once the fire has stopped burning, and (b) the total number of cells that have been burnt by the fire (including any initially burning cells in **`burn_seeds`**).

The arguments are of the following types:

- **`f_grid`**: a list of lists (of dimensions M by M)
- **`h_grid`**: a list of lists (of dimensions M by M)
- **`i_threshold`**: an integer
- **`w_direction`**: a string
- **`burn_seeds`**: a list of integer tuples (i, j) where $i<M$ and $j<M$

You may assume that all arguments are valid, as defined in previous questions. You have been provided with a reference version of the function **`check_ignition`** as described in Question 2. You may find it helpful to define one or more additional functions that carry out a single step of the model run, determining the new burning state and fuel load at time $t+1$ on the basis of the model state at time t.

For example:

```
>>> run_model([[2, 2], [2, 2]], [[1, 1], [1, 1]], 1, 'N', [(0, 0)])
([[0, 0], [0, 0]], 4)
>>> run_model([[2, 0], [0, 2]], [[1, 1], [1, 1]], 2, 'S', [(0, 0)])
([[0, 0], [0, 2]], 1)
```

Question 4: Test Cases

In addition to implementing your solutions, you are also required to submit a set of test cases that can be used to test your Question 3 **`run_model`** function.
You should aim to make your test cases as **complete** as possible. That is, they should be sufficient to pick up incorrect implementations of the model.

Your set of test cases may assume that the input passed to your function will be of the correct types and will be well-formed.

Your test cases suite will be evaluated by running it on several **known** incorrect implementations, in which it should detect incorrect behaviour; i.e. returning an incorrect final state of the landscape and/or number of cells burnt.

You should specify your test cases as a series of calls to the function **`test_run_model(input_args, expected_return_value)`**, where the first argument **`input_args`** contains a list of **`f_grid`**, **`h_grid`**, **`i_threshold`**, **`w_direction`** **`burn_seeds`** representing the call to the function **`run_model`** (described in Question 3) and **`expected_return_value`** is the expected return from the function **`run_model`**, namely a list containing the final state of the landscape once the fire has stopped burning, and the total number of cells that have been burnt by the fire (as in Question 3).

That is, you specify both the *arguments* **and** *return value* for **`run_model`** as arguments to **`test_run_model`**.

For example, using the first two examples from Question 2:

```python
from testcase_tournament import test_fn as test_run_model

test_run_model([[[2, 2], [2, 2]], [[1, 1], [1, 1]], 1, 'N', [(0,
0)]],[[[0, 0], [0, 0]], 4])
test_run_model([[[2, 0], [0, 2]], [[1, 1], [1, 1]], 2, 'S', [(0, 0)]],
[[[0, 0], [0, 2]], 1])
```

Question 5: Bonus

The final question is for bonus marks, and is deliberately quite a bit harder than the four basic questions (and the number of marks on offer is deliberately not commensurate with the amount of effort required — bonus marks aren't meant to be easy to get!). Only attempt this is you have completed the earlier questions, and are up for a challenge!

In this question, you will use the bushfire model to determine the optimal cell or cells in a landscape in which to conduct a prescribed burn in order to best protect a town from a future bushfire of unknown timing and origin.

For this question, we modify our original definition of a landscape to include a **town cell**, containing a town. The town cell has a non-zero fuel load; that is, the town *can* catch fire.

Prescribed burns

A prescribed burn is a controlled fire used as part of forest management in order to reduce the risk of future uncontrolled fires.

In our simulation model, the rules of a prescribed burn are that it will only occur on a day with *no* wind, and will commence on a single **prescribed burn cell** with a non-zero fuel load. A prescribed burn will *not* be conducted on the cell containing the town.

A prescribed burn spreads just like a normal bushfire; however, due to the controlled nature of the fire, any burning cell contributes only *half* as many points to the ignition factor of adjacent cells as it ordinarily would (taking slope into account). That is, if it would normally contribute 0.5, 1 or 2 points, it will now only contribute 0.25, 0.5, or 1 point. This reduction applies both to the original prescribed burn cell and to any cell that subsequently catches fire during the prescribed burn. As with a normal bushfire, a prescribed burn will continue until no cells remain on fire.

Scoring prescribed burn cells

We filter out invalid prescribed burn cells and score the remaining valid prescribed burn cells as follows:

- Any prescribed burn cell that results in the the town cell catching fire is deemed invalid.
- Following the completion of a prescribed burn, we will consider scenarios in which potential bushfires start in any (single) seed cell with a non-zero fuel load (after the prescribed burn), except for the town cell, on a day with any possible wind conditions. Thus, for a landscape of dimensions M, we will consider *up to* $(M2-2)\times 9$ bushfire scenarios. 2 is subtracted because we don't seed a bushfire on the town cell or cells with zero fuel load, of which there is *at least* one, being the cell in which the prescribed burn was conducted. For each seed cell there are 9 possible wind directions to consider, including no wind.
- Valid prescribed burn cells are scored according to the proportion of scenarios in which the town cell caught fire.

The optimal cell or cells for prescribed burning are those with the *lowest* score; that is, that have been more effective at protecting the town.

In the first example below, there are 4 cells with a non-zero fuel load, one of which ($c1,1$) is the town cell. Therefore there are three cells in which a prescribed burn can be conducted. None of these will result in the town being burnt, therefore they are all valid. When we test the 18 possible bushfire scenearios, we find that for one valid prescribed burn cell ($c0,1$), *all* subsequent bushfires will result in the town catching fire. For the other two prescribed burn cells ($c0,0$ and $c1,0$), only half of the subsequent bushfires will result in the town catching fire; thus, either of these would be the optimal location in which to carry out a prescribed burn in this landscape.

Your task

Write a function **`plan_burn(f_grid, h_grid, i_threshold, town_cell)`** that determines the optimal prescribed burn cell or cells. **`f_grid`**, **`h_grid`** and **`i_threshold`** are all as defined in Questions 2 and 3. **`town_cell`** is a tuple containing the coordinates of the town cell.
Your function should return a sorted list containing the coordinates of the optimal prescribed burn cell or cells, as defined above. If there are no valid prescribed burn cells, this list will be empty.

For example:

```
>>> plan_burn([[2, 2], [1, 2]], [[1, 2], [1, 2]], 2, (1, 1))
[(0, 0), (1, 0)]
>>> plan_burn([[0, 0, 0, 0, 0], [0, 2, 2, 0, 0], [0, 0, 0, 1, 0], [0, 0,
0, 1, 0], [1, 0, 0, 0, 0]], [[2, 2, 2, 2, 2], [2, 1, 2, 2, 2], [2, 2, 2,
2, 2], [2, 2, 2, 1, 2], [2, 2, 2, 2, 2]], 2, (3, 3))
[(1, 1), (1, 2), (2, 3)]
```