Speleology

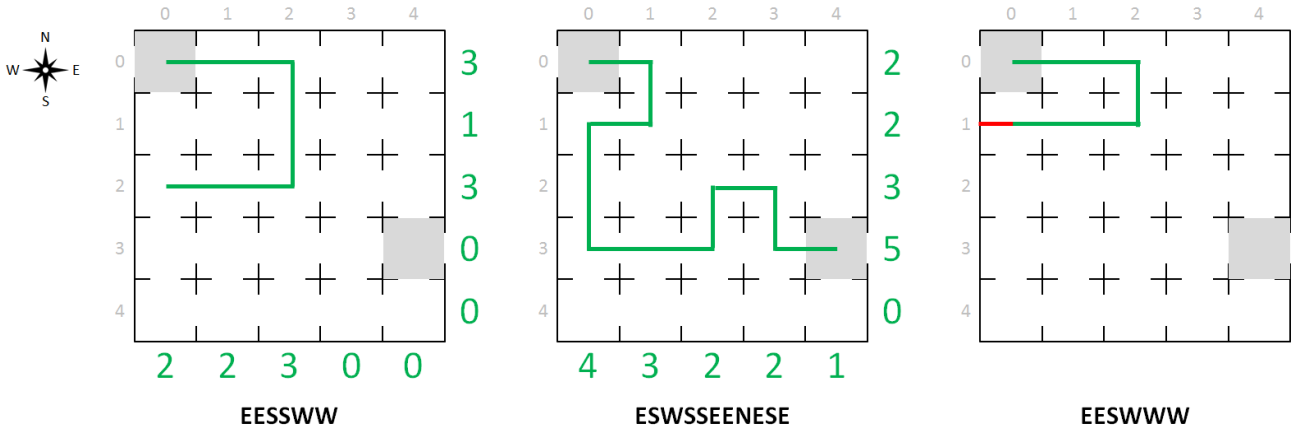
Speleology is a sub-domain of the geosciences that deals with the study of caves. Although considered serious science, the exploration of underground cavities also encompasses certain elements of sportiness, adventure and courage. Exploring caves, especially if their structure is only partially known, is not without any danger and requires skill and training.

Speleology is a cross-disciplinary field that combines knowledge of chemistry, biology, geology, physics, meteorology and cartography to develop portraits of caves as complex, evolving systems. It is the scientific study of the make-up of caves and other karst features, their structure, physical properties, history, life forms, and the processes by which they form (*speleogenesis*) and change over time (*speleomorphology*). The term speleology is also sometimes applied to the recreational activity of exploring caves, but this is more properly known as caving, spelunking or potholing. Speleology and caving are often connected, as the physical skills required for *in situ* study are the same.

Assignment

As part of your speleology training, you are given the map of a cave that consists of a number of halls. The halls of the cave are arranged in a rectangular $m \times n$ grid with m rows and n columns. The cave has only a single entrance and a single exit which are located in a hall at the edge of the grid. Each hall is connected to each of its neighbouring halls north, south, east and west, unless it collides with the edge of the cave. Your task is to find a valid path that goes from the hall where the entrance of the cave is located, and winds its way by visiting neighbouring halls until the hall is reached where the exit is located. The path may not go through any hall more than once. In addition, the cave map contains clues that tell how many halls the path passes in each row and column of the grid.

A sample map of a cave whose halls are arranged in a 5×5 grid is given below. In order to reference the halls in the grid, we increasingly index the rows from north to south and the columns from west to east, with indexing starting at zero (grey numbers on the sample map). The cave has an entrance at the western edge of hall (0, 0) and an exit at the eastern edge of hall (3, 4). A path is described as a succession of letters that indicate which of the neighbouring halls is visited next: `N` (north), `S` (south), `E` (east) or `W` (west).



Map of a cave that has its halls arranged in 5 rows and 5 columns, with an entrance in hall (0, 0) and an exit in hall (3, 4). The path ESWSSEENESE (center) leads to the exit, with the number of passages through the rows and the columns of the map indicated in green. The path EESSWW (left) does not lead to the exit. The path EESWWW (right) collides with the western edge of the cave.

The path ESWSSEENESE (center map in the figure) winds its way from the entrance to the exit, with the number of passages through the rows and columns of the map indicated in green. As such, there are for example 2 passages through the northernmost row and 4 passages through the westernmost column. In addition, the path visits each hall in the grid at most once. The path EESSWW (left map) is invalid because it does not lead the the exit. The path EESWWW (right) is equally invalid because it collides with the western edge of the cave.

Your task is to implement a class Cave whose objects represent a cave map. The cave consists of a number of halls, arranged in a rectangular $m \times n$ grid. Apart from the positions of the entrance and the exit of the cave, the map also contains clues that tell how many halls a valid path passes through each row and column of the grid. Make sure the objects of the class at least support the following methods, which can be used to check whether or not a given path is valid.

- An initialization method that can be used to define a cave map and the expected number of passages through each row and column of the grid. The method takes four arguments that are all sequences (lists or tuples) of positive integers. The first argument contains m integers that indicate how many halls a valid path visits in each row (from north to south) of the grid. The second argument contains n integers that indicate how many halls a valid path visits in each column (from west to east) of the grid. The last two arguments contain the row index r and the column index c of the halls where respectively the entrance and the exit of the cave are located. The initialization method must raise a n AssertionError with the message invalid cave map, in case at least one of the following conditions is not met:
  - all integers in the first argument are in the interval $[0, n]$
  - all integers in the second argument are in the interval $[0, m]$
  - the halls where the entrance and exit of the cave are located are at the edge of the grid
- A method path that takes a string argument. The given string must contain a sequence of the letters N (north), S (south), E (east) and W (west). These letters describe the successive movements to neighbouring halls that are followed by a path, starting from the hall where the entrance of the cave is located. The method must return a list containing the positions of the halls in the order in which they are visited by the given path. The position of a hall is described by a tuple (r, c), where r indicates the row index and c the column index

of the hall in the grid. In case the given path would make a movement that forces it to collide with the edge of the cave, the method must raise an AssertionError with the message collides with cave.

- A method passages that takes a string argument that has the same meaning as the argument passed to the method path. In case this string describes a path starting from the entrance that would make a movement that forces it to collide with the edge of the cave, the method must raise an AssertionError with the message collides with cave. The method must return a tuple that itself contains two tuples, respectively containing the number of halls visited by the given path in each row (from north to south) and column (from west to east) of the grid.
- A method validPath that takes a string argument that has the same meaning as the argument passed to the method path. In case this string describes a path starting from the entrance that would make a movement that forces it to collide with the edge of the cave, the method must raise an AssertionError with the message collides with cave. The method must return a Boolean value that indicates whether or not the given path is valid. A path is valid if it *i*) ends in the hall where the exit of the cave is located, *ii*) does not visit the same hall more than once, *iii*) passes through each row the expected number of times and *iv*) passes through each column the expected number of times.

Example

```
>>> cave = Cave((2, 2, 3, 5, 0), (4, 3, 2, 2, 1), (0, 0), (3, 4))

>>> cave.path('EESSWW')
[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1), (2, 0)]
>>> cave.passages('EESSWW')
((3, 1, 3, 0, 0), (2, 2, 3, 0, 0))
>>> cave.validPath('EESSWW')
False

>>> cave.path('ESWSSEENESE')
[(0, 0), (0, 1), (1, 1), (1, 0), (2, 0), (3, 0), (3, 1), (3, 2), (2, 2), (2, 3), (3, 3), (3, 4)]
>>> cave.passages('ESWSSEENESE')
((2, 2, 3, 5, 0), (4, 3, 2, 2, 1))
>>> cave.validPath('ESWSSEENESE')
True

>>> cave.path('EESWWW')
Traceback (most recent call last):
AssertionError: collides with edge
>>> cave.passages('EESWWW')
Traceback (most recent call last):
AssertionError: collides with edge
>>> cave.validPath('EESWWW')
Traceback (most recent call last):
AssertionError: collides with edge

>>> cave = Cave((2, 2, 3, 6, 0), (4, 3, 2, 2, 1), (0, 0), (3, 4))
Traceback (most recent call last):
AssertionError: invalid cave map

>>> cave = Cave((2, 2, 3, 5, 0), (4, 3, 2, 2, 1), (1, 1), (3, 4))
Traceback (most recent call last):
AssertionError: invalid cave map
```

Speleologie is een tak van de aardwetenschappen die zich bezighoudt met de studie van

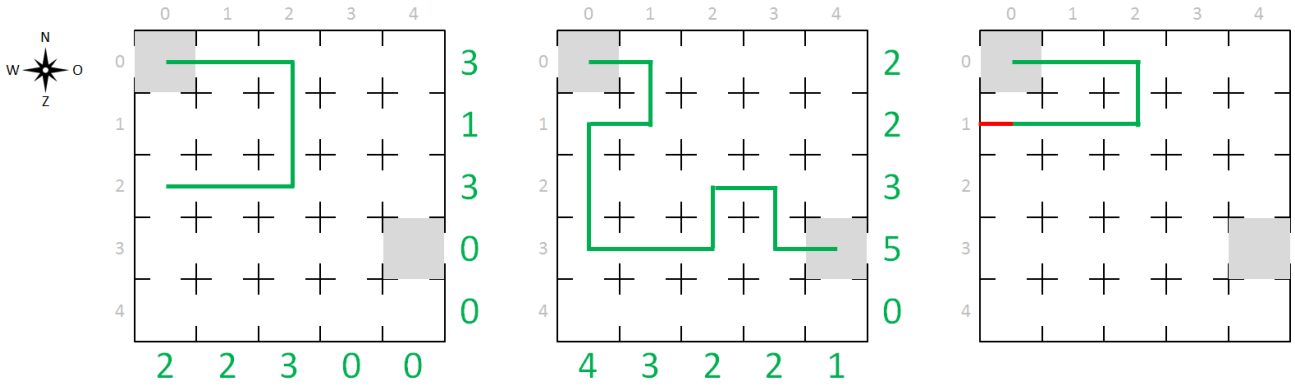
grotten. Hoewel speleologie een serieuze wetenschap is, komt er bij het verkennen van grotten ook een element van sportiviteit, avontuur en moed kijken. Het verkennen van grotten, zeker als zij nog onvolledig bekend zijn, is zeker niet zonder gevaar en vereist vaardigheid en training.

Als wetenschap bestudeert de speleologie onder andere de geologie (het ontstaan van gesteentelagen, grotten en druipsteen), de hydrologie (onderaardse waterlopen), de biologie (de fauna en flora in een grot), de paleontologie (overblijfselen van uitgestorven dieren in grotten) en de archeologie (grotschilderingen en werktuigen uit de prehistorie). Ook de geologie van streken waar grotten voorkomen is onderwerp van studie. De speleologie als sport kunnen we vergelijken met de bergsport, omdat voor een groot deel dezelfde technieken en materialen gebruikt worden.

Opgave

Als onderdeel van je speleologietraining krijg je de plattegrond van een grot die bestaat uit een aantal zalen. De zalen van de grot zijn gerangschikt in een rechthoekig $m \times n$ rooster met m rijen en n kolommen. De grot heeft slechts één ingang en één uitgang, en vanuit elke zaal kan je naar elk van de naburige zalen ten noorden, zuiden, oosten en westen, tenzij je daarbij tegen de rand van de grot botst. Je opdracht bestaat erin een geldige doorgang te vinden die vertrekt vanaf de zaal waar de ingang van de grot gelegen is, en steeds loopt via een naburige zaal totdat de zaal bereikt wordt waar de uitgang van de grot gelegen is. Hierbij mag elke zaal hoogstens één keer bezocht worden. Bovendien bevat de plattegrond ook aanwijzingen hoeveel zalen de doorgang moet passeren op elke rij en kolom van het rooster.

Als voorbeeld krijg je hieronder de plattegrond van een grot, waarvan de zalen gerangschikt zijn in een 5×5 rooster. Om makkelijk naar de zalen te kunnen verwijzen, nummeren we de rijen oplopend van noord naar zuid en de kolommen van west naar oost, waarbij we steeds beginnen nummeren vanaf nul (grijze getallen op de voorbeeld plattegrond). De grot heeft een ingang in zaal (0,0) en een uitgang in zaal (3, 4). We beschrijven een doorgang als een opeenvolging van letters die aangeven welk van de naburige zalen in een volgende stap bezocht wordt: N (noord), Z (zuid), O (oost) of W (west).



Plattegrond van een grot met 5 rijen en 5 kolommen, een ingang in cel (0, 0) en een uitgang in cel (3, 4). De doorgang OZWZZOONOZO (midden) is een doorgang die leidt naar de uitgang, waarbij het aantal passages doorheen de rijen en kolommen van de plattegrond worden aangegeven in het groen. De doorgang OOZZWW (links) leidt niet naar de uitgang. De doorgang OOZWWW (rechts) botst tegen de rand van de grot.

De doorgang OZWZZOONOZO (middelste plattegrond van de figuur) is een doorgang die vanaf de ingang leidt naar de uitgang, waarbij het aantal passages doorheen de rijen en kolommen van de plattegrond wordt aangegeven in het groen. Zo zijn er bijvoorbeeld 2 passages door de meest noordelijke rij en 4 passages door de meest westelijke kolom. Bovendien bezoekt deze doorgang elke zaal hoogstens één keer. De doorgang OOZZWW (linkse plattegrond) is niet geldig omdat die niet naar de uitgang leidt. De doorgang OOZWWW (rechtse plattegrond) is eveneens ongeldig, omdat er tegen de rand van de grot wordt gebotst.

Je opdracht bestaat erin een klasse Grot te schrijven waarvan elk object de plattegrond van een grot voorstelt. De grot bestaat uit een aantal zalen die gerangschikt zijn in een rechthoekig $m \times n$ rooster. Naast de posities van de ingang en uitgang, bevat de map ook aanwijzingen hoeveel zalen een geldige doorgang moet passeren op elke rij en kolom van het rooster. Zorg ervoor dat de objecten minstens over de volgende methoden beschikken, waarmee kan nagegaan worden of een gegeven doorgang al dan niet geldig is.

- Een initialisatiemethode waarmee de plattegrond van de grot kan vastgelegd worden, alsook het gewenste aantal passages op elke rij en kolom van het rooster. Aan deze methode moeten vier argumenten doorgegeven worden, waarbij elk argument een reeks (een lijst of een tuple) natuurlijke getallen is. Het eerste argument bevat m getallen die aangeven hoeveel passages de gezochte doorgang moet hebben door elke rij (van noord naar zuid) van het rooster. Het tweede argument bevat n getallen die aangeven hoeveel passages de gezochte doorgang moet hebben door elke kolom (van west naar oost) van het rooster. De laatste twee argumenten bevatten het rijnummer r en kolomnummer k van de zalen waar respectievelijk de ingang en de uitgang van de grot gelegen zijn. De initialisatiemethode moet een AssertionError opwerpen met de boodschap ongeldige plattegrond indien minstens één van de volgende voorwaarden niet voldaan is:
  - alle waarden van het eerste argument moeten in het interval $[0, n]$ liggen
  - alle waarden van het tweede argument moeten in het interval $[0, m]$ liggen
  - de zalen waarin de ingang en uitgang van de grot gelegen zijn moeten aan de rand van het rooster liggen
- Een methode doorgang waaraan een string moet doorgegeven worden. Deze string bevat een opeenvolging van de letters N (noord), Z (zuid), O (oost) en W (west). Deze letters

beschrijven de opeenvolgende verplaatsingen naar naburige zalen die een doorgang maakt, vertrekkende vanaf de zaal waar de ingang van de grot gelegen is. De methode moet de lijst van de posities van de zalen teruggeven in de volgorde waarin ze bezocht worden door de gegeven doorgang. Hierbij wordt de positie van een zaal beschreven door een tuple (r, k), waarbij r het rijnummer en k het kolomnummer van de zaal in het rooster aangeeft. Indien de beschreven doorgang een verplaatsing zou maken waardoor tegen de rand van de grot wordt gebotst, dan moet de methode een AssertionError opwerpen met de boodschap botst tegen rand.

- Een methode passages waaraan een string moet doorgegeven worden met dezelfde betekenis als bij de methode doorgang. Indien deze string een doorgang vanaf de ingang beschrijft, waarbij op een bepaald moment een verplaatsing zou gemaakt worden die tegen de rand van de grot botst, dan moet de methode een AssertionError opwerpen met de boodschap botst tegen rand. De methode moet een tuple teruggeven dat zelf twee tuples bevat, die het aantal passages aangeven die de gegeven doorgang maakt met respectievelijk de rijen (van noord naar zuid) en de kolommen (van west naar oost) van het rooster.
- Een methode geldigeDoorgang waaraan een string moet doorgegeven worden met dezelfde betekenis als bij de methode doorgang. Indien deze string een doorgang vanaf de ingang beschrijft, waarbij op een bepaald moment een verplaatsing zou gemaakt worden die tegen de rand van de grot botst, dan moet de methode een AssertionError opwerpen met de boodschap botst tegen rand. De methode moet een Booleaanse waarde teruggeven, die aangeeft of de gegeven doorgang al dan niet geldig is. Een doorgang is geldig als hij *i*) eindigt in de zaal waar de uitgang van de grot gelegen is, *ii*) dezelfde zaal niet meermaals bezoekt, *iii*) elke rij het opgegeven aantal keer passeert en *iv*) elke kolom het opgegeven aantal keer passeert.

Voorbeeld

```
>>> grot = Grot((2, 2, 3, 5, 0), (4, 3, 2, 2, 1), (0, 0), (3, 4))

>>> grot.doorgang('OOZZWW')
[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (2, 1), (2, 0)]
>>> grot.passages('OOZZWW')
((3, 1, 3, 0, 0), (2, 2, 3, 0, 0))
>>> grot.geldigeDoorgang('OOZZWW')
False

>>> grot.doorgang('OZWZZOONOZO')
[(0, 0), (0, 1), (1, 1), (1, 0), (2, 0), (3, 0), (3, 1), (3, 2), (2, 2), (2, 3), (3, 3), (3, 4)]
>>> grot.passages('OZWZZOONOZO')
((2, 2, 3, 5, 0), (4, 3, 2, 2, 1))
>>> grot.geldigeDoorgang('OZWZZOONOZO')
True

>>> grot.doorgang('OOZWWW')
Traceback (most recent call last):
AssertionError: botst tegen rand
>>> grot.passages('OOZWWW')
Traceback (most recent call last):
AssertionError: botst tegen rand
>>> grot.geldigeDoorgang('OOZWWW')
Traceback (most recent call last):
AssertionError: botst tegen rand
```

```
>>> grot = Grot((2, 2, 3, 6, 0), (4, 3, 2, 2, 1), (0, 0), (3, 4))
Traceback (most recent call last):
AssertionError: ongeldige plattegrond

>>> grot = Grot((2, 2, 3, 5, 0), (4, 3, 2, 2, 1), (1, 1), (3, 4))
Traceback (most recent call last):
AssertionError: ongeldige plattegrond
```