Самые короткие и самые длинные слова

Напишите класс **MinMaxWordFinder**. Класс должен уметь анализировать текст и находить в нём слова наименьшей и наибольшей длины. Текст состоит из предложений, которые добавляются в обработку методом add_sentence. Метод shortest_words возвращает список самых коротких на данный момент слов, метод longest_words — самых длинных. Слова, возвращаемые методами shortest_words и longest_words, должны быть отсортированы по алфавиту.

Если одно из самых коротких слов встретилось в исходных предложениях несколько раз, оно должно столько же раз повториться в списке самых коротких слов. Самые длинные слова наоборот должны входить в список без повторов.

Формат ввода

Каждый тест представляет собой код, в котором будет использоваться ваш класс. Файл с решением не обязательно называть solution.py, он будет переименован автоматически. Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

Пример 1

Ввод	Вывод

`from solution import MinMaxWordFinder`
 	
`finder = MinMaxWordFinder()`
`finder.add_sentence('hello abc world')`
`finder.add_sentence('def asdf qwert')`
`print(' '.join(finder.shortest_words()))`
`print(' '.join(finder.longest_words()))`
	`abc def`
	`hello qwert world`

Пример 2

Ввод	Вывод

`from solution import MinMaxWordFinder`
 	
`finder = MinMaxWordFinder()`
`finder.add_sentence('hello')`
`finder.add_sentence('abc')`
`finder.add_sentence('world')`
`finder.add_sentence('def')`
`finder.add_sentence('asdf')`
`finder.add_sentence('qwert')`
`print(' '.join(finder.shortest_words()))`
`print(' '.join(finder.longest_words()))`
	`abc def`
	`hello qwert world`

Ввод	Вывод

Пример 3

Ввод	Вывод

`from solution import MinMaxWordFinder`
 	
`finder = MinMaxWordFinder()`
`finder.add_sentence('hello')`
`finder.add_sentence('  abc     def    ')`
`finder.add_sentence('world')`
`finder.add_sentence('            abc          ')`
`finder.add_sentence('asdf')`
`finder.add_sentence('qwert')`
`print(' '.join(finder.shortest_words()))`
`print(' '.join(finder.longest_words()))`
	`abc abc def`
	`hello qwert world`

Ограничивающий прямоугольник

Создайте класс **BoundingRectangle**, который обрабатывает точки на плоскости и строит по ним прямоугольник минимального размера, в который входят все эти точки. Если точка лежит на границе прямоугольника, считается, что она в него входит.

Нужно определить следующие методы (rect – экземпляр **BoundingRectangle**):

rect.add_point(x, y) — добавить новую точку.

rect.width() — ширина прямоугольника.

rect.height() — высота прямоугольника.

rect.bottom_y() — Y-координата нижней границы прямоугольника.

rect.top_y() — Y-координата верхней границы прямоугольника.

rect.left_x() — X-координата левой границы прямоугольника.

rect.right_x() — X-координата правой границы прямоугольника.

Гарантируется, что хотя бы одна точка будет добавлена в экземпляр до вызова методов, возвращающих описание прямоугольника.

Формат ввода

Каждый тест представляет собой код, в котором будет использоваться ваш класс. Файл с решением не обязательно называть solution.py, он будет переименован автоматически. Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

Пример 1

Ввод	Вывод

`from solution import BoundingRectangle`
 	
`rect = BoundingRectangle()`
`rect.add_point(-1, -2)`
`rect.add_point(3, 4)`
`print(rect.left_x(), rect.right_x())`
`print(rect.bottom_y(), rect.top_y())`
`print(rect.width(), rect.height())`
	`-1 3`
	`-2 4`
	`4 6`

Пример 2

Ввод	Вывод

`from solution import BoundingRectangle`
 	
`rect = BoundingRectangle()`
`rect.add_point(10, 20)`
`rect.add_point(5, 7)`
`rect.add_point(6, 3)`
`print(rect.left_x(), rect.right_x())`
`print(rect.bottom_y(), rect.top_y())`
`print(rect.width(), rect.height())`
	`5 10`
	`3 20`
	`5 17`

Пример 3

Ввод	Вывод

`from solution import BoundingRectangle`
 	
`rect = BoundingRectangle()`
`rect.add_point(-11, -12)`
`rect.add_point(13, -14)`
`rect.add_point(-15, 10)`
`print(rect.left_x(), rect.right_x())`
`print(rect.bottom_y(), rect.top_y())`
`print(rect.width(), rect.height())`
`print()`
`rect.add_point(-21, -12)`
`rect.add_point(13, -14)`
`rect.add_point(-15, 36)`
`print(rect.width(), rect.height())`
`print(rect.left_x(), rect.right_x())`
	`-15 13`
	`-14 10`
	`28 24`
	
	`34 50`
	`-21 13`
	`-14 36`
	
	`-14 78`
	`68 92`
	`-55 13`

Ввод	Вывод
print(rect.bottom_y(), rect.top_y()) print() rect.add_point(-21, 78) rect.add_point(13, -14) rect.add_point(-55, 36) print(rect.bottom_y(), rect.top_y()) print(rect.width(), rect.height()) print(rect.left_x(), rect.right_x()) print()	

Таблица

Реализуйте класс **Table**, который хранит целые числа в двумерной таблице. При инициализации Table(rows, cols) экземпляру передаются число строк и столбцов в таблице. Строки и столбцы нумеруются с нуля. Ячейки таблицы инициализируются нулями.

table.get_value(row, col) — прочитать значение из ячейки со строкой row, столбцом col. Если ячейка с индексами row и col не лежит внутри таблицы, нужно вернуть None.

table.set_value(row, col, value) — записать число в ячейку со строкой row, столбцом col. Гарантируется, что в тестах будет в запись только в ячейки внутри таблицы.

table.n_rows() — вернуть число строк в таблице.

table.n_cols() — вернуть число столбцов в таблице.

Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.
Файл с решением не обязательно называть solution.py, он будет переименован автоматически.
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

Пример 1

Ввод	Вывод
from solution import Table tab = Table(3, 5) tab.set_value(0, 1, 10) tab.set_value(1, 2, 20)	0 10 0 0 0 0 0 20 0 0 0 0 0 30 0

Ввод	Вывод

tab.set_value(2, 3, 30)
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()	

Пример 2

Ввод	Вывод

from solution import Table
 	
tab = Table(2, 2)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
tab.set_value(0, 0, 10)
tab.set_value(0, 1, 20)
tab.set_value(1, 0, 30)
tab.set_value(1, 1, 40)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
for i in range(-1, tab.n_rows() + 1):
    for j in range(-1, tab.n_cols() + 1):
        print(tab.get_value(i, j), end=' ')
    print()
print()
	0 0
	0 0
	
	10 20
	30 40
	
	None None None None
	None 10 20 None
	None 30 40 None
	None None None None

Пример 3

Ввод	Вывод

from solution import Table
 	
tab = Table(1, 1)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
tab.set_value(0, 0, 1000)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
	0
	
	1000
	
	None None None
	None 1000 None
	None None None

Ввод	Вывод

`    print()`
`print()`
 	
`for i in range(-1, tab.n_rows() + 1):`
`    for j in range(-1, tab.n_cols() + 1):`
`        print(tab.get_value(i, j), end=' ')`
`    print()`
`print()`	

Прямоугольники

Реализуйте класс **Rectangle** для описания прямоугольника, стороны которого параллельны осям координат.

При инициализации экземпляра передаются координаты левой нижней точки прямоугольника x и y, а также его ширина и высота w и h. Таким образом, координаты верхнего правого угла — (x + w) и (y + h).

При вызове метода intersection (например, rect1.intersection(rect2)) должен возвращаться прямоугольник, который возникает как пересечение rect1 и rect2. Если прямоугольники не пересекаются, должен возвращаться объект None.

Также необходимо реализовать метод get для каждого из атрибутов класса, возвращающий заданное значение данного атрибута. Пример: get_*() - возвращает значение атрибута *.

Гарантируется, что во входных данных ширина и высота любого прямоугольника положительны.

Если пересечением прямоугольников является точка или отрезок, то следует считать, что они не пересекаются.

Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.
Файл с решением не обязательно называть solution.py, он будет переименован автоматически.

Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

Пример 1

Ввод	Вывод

`from solution import Rectangle`
 	
`rect1 = Rectangle(0, 0, 10, 10)`
`rect2 = Rectangle(5, 5, 10, 10)`
`rect3 = rect1.intersection(rect2)`
 	
`if rect3 is None:`
`  print('No intersection')`
`else:`
`  print(rect3.get_x(), rect3.get_y(), rect3.get_w(), rect3.get_h())`
	`5 5 5 5`

Пример 2

Ввод	Вывод

`from solution import Rectangle`
 	
`rect1 = Rectangle(0, 0, 10, 10)`
`rect2 = Rectangle(10, 0, 10, 10)`
`rect3 = rect1.intersection(rect2)`
 	
`if rect3 is None:`
`  print('No intersection')`
`else:`
`  print(rect3.get_x(), rect3.get_y(), rect3.get_w(),`
`rect3.get_h())`
	`No intersection`

Пример 3

Ввод	Вывод

`from solution import Rectangle`
 	
`rect1 = Rectangle(3, 5, 2, 1)`
`rect2 = Rectangle(1, 2, 10, 10)`
`rect3 = rect1.intersection(rect2)`
 	
`if rect3 is None:`
`  print('No intersection')`
`else:`
`  print(rect3.get_x(), rect3.get_y(), rect3.get_w(), rect3.get_h())`
	`3 5 2`

Таблица с изменяемым размером

Реализуйте класс **Table**, который хранит целые числа в двумерной таблице. При инициализации Table(rows, cols) экземпляру передаются число строк и столбцов в таблице. Строки и столбцы нумеруются с нуля.

table.get_value(row, col) — прочитать значение из ячейки в строке row, столбце col. Если ячейка с индексами row и col не лежит внутри таблицы, нужно вернуть None.

table.set_value(row, col, value) — записать число в ячейку строки row, столбца col. Гарантируется, что в тестах будет в запись только в ячейки внутри таблицы.

table.n_rows() — вернуть число строк в таблице

table.n_cols() — вернуть число столбцов в таблице

table.delete_row(row) — удалить строку с номером row

table.delete_col(col) — удалить колонку с номером col

table.add_row(row) — добавить в таблицу новую строку с индексом row. Номера строк >= row, должны увеличится на единицу. Новая строка состоит из нулей.

table.add_col(col) — добавить в таблицу новую колонку с индексом col. Номера колонок >= col, должны увеличится на единицу. Новая колонка состоит из нулей.

Формат ввода

Каждый тест представляет собой код, в котором будут использоваться ваш класс.
Файл с решением не обязательно называть solution.py, он будет переименован автоматически.
Тест запускается с вашим классом, а его вывод сравнивается с правильным решением.

Пример 1

Ввод	Вывод
`from solution import Table`	`0 10 0 0 0` `0 0 20 0 0`

Ввод	Вывод

tab = Table(3, 5)
tab.set_value(0, 1, 10)
tab.set_value(1, 2, 20)
tab.set_value(2, 3, 30)
for i in range(tab.n_rows()):
  for j in range(tab.n_cols()):
  print(tab.get_value(i, j), end=' ')
  print()
print()
 	
tab.add_row(1)
 	
for i in range(tab.n_rows()):
  for j in range(tab.n_cols()):
  print(tab.get_value(i, j), end=' ')
  print()
print()
	0 0 0 30 0
	
	0 10 0 0 0
	0 0 0 0 0
	0 0 20 0 0
	0 0 0 30 0

Пример 2

Ввод	Вывод

from solution import Table
 	
tab = Table(2, 2)
 	
for i in range(tab.n_rows()):
  for j in range(tab.n_cols()):
  print(tab.get_value(i, j), end=' ')
  print()
print()
 	
tab.set_value(0, 0, 10)
tab.set_value(0, 1, 20)
tab.set_value(1, 0, 30)
tab.set_value(1, 1, 40)
 	
for i in range(tab.n_rows()):
  for j in range(tab.n_cols()):
  print(tab.get_value(i, j), end=' ')
  print()
print()
 	
for i in range(-1, tab.n_rows() + 1):
  for j in range(-1, tab.n_cols() + 1):
  print(tab.get_value(i, j), end=' ')
  print()
print()
 	
tab.add_row(0)
tab.add_col(1)
 	
for i in range(-1, tab.n_rows() + 1):
  for j in range(-1, tab.n_cols() + 1):
  print(tab.get_value(i, j), end=' ')
  print()
print()
	0 0
	0 0
	
	10 20
	30 40
	
	None None None None
	None 10 20 None
	None 30 40 None
	None None None None
	
	None None None None None
	None 0 0 0 None
	None 10 0 20 None
	None 30 0 40 None
	None None None None None

Ввод	Вывод

Пример 3

Ввод	Вывод

from solution import Table
 	
tab = Table(1, 1)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
tab.set_value(0, 0, 1000)
 	
for i in range(tab.n_rows()):
    for j in range(tab.n_cols()):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
for i in range(-1, tab.n_rows() + 1):
    for j in range(-1, tab.n_cols() + 1):
        print(tab.get_value(i, j), end=' ')
    print()
print()
 	
tab.add_row(0)
tab.add_row(2)
tab.add_col(0)
tab.add_col(2)
 	
tab.set_value(0, 0, 2000)
tab.set_value(0, 2, 3000)
tab.set_value(2, 0, 4000)
tab.set_value(2, 2, 5000)
 	
for i in range(-1, tab.n_rows() + 1):
    for j in range(-1, tab.n_cols() + 1):
        print(tab.get_value(i, j), end=' ')
    print()
print()
	0
	
	1000
	
	None None None
	None 1000 None
	None None None
	
	None None None None None
	None 2000 0 3000 None
	None 0 1000 0 None
	None 4000 0 5000 None
	None None None None None