МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)

ОТЧЁТ

по дисциплине «Программирование на Python (методы хранения и обработки данных)»
на тему «Вычисление мощности множества точек пересечения границы выпуклой оболочки с заданной прямой»

Группа Б21-215

Студент А.А. Шахназаров

Руководитель работы
к.ф.-м.н., доцент Е.А. Роганов

Москва 2022

Аннотация

Отчёт посвящён модификации проекта «Выпуклая оболочка». Решалась задача подсчёта количества точек пересечения выпуклой оболочки с заданной прямой.

Содержание

1. Введение

Модификация проекта «Выпуклая оболочка»[2] решает задачу индуктивного перевычисления выпуклой оболочки последовательно поступающих точек плоскости и таких её характеристик, как количество точек пересечения с заданной прямой. Решение этой задачи требует знания теории индуктивных функций [3], основ аналитической геометрии и векторной алгебры и языка Python [4].

Для подготовки пояснительной записки необходимо знакомство с программой компьютерной вёрстки LaTeX [5], умение набирать математические формулы [6] и включать в документ графические изображения и исходные коды программ.

2. Необходимые для решения задачи теоретические аспекты

Прямые пересекаются, если векторное произведение их направляющих векторов не равно нулю. Но ребро выпуклой оболочки не прямая, а отрезок, поэтому пересечение образуется лишь тогда, когда концы отрезка лежат по разные стороны от прямой. Чтобы это проверить, проведём два вектора, начала которых будут совпадать с началом направляющего вектора прямой, а концы — с концами отрезка (рис. 1), и найдём модули векторного произведения этих векторов с направляющим. Если концы лежат по разные стороны, то произведение модулей будет меньше нуля и равно нулю, если какой-либо из концов будет лежать на прямой.

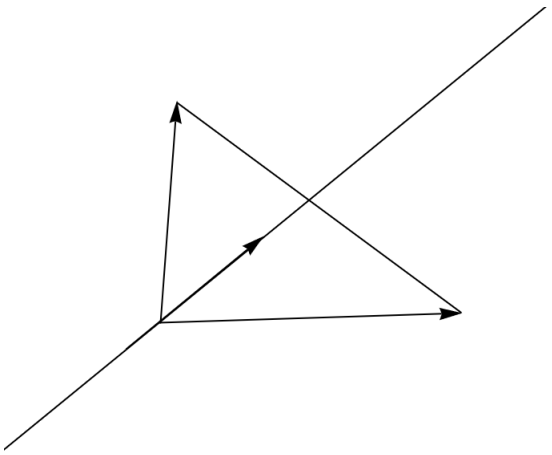

Рис. 1.

Пусть прямая задана двумя точками с координатами (x_1, y_1) и (x_2, y_2), а ребро выпуклой оболочки — точками с координатами (x_3, y_3) и (x_4, y_4), тогда условие их пересечения выглядит следующим образом:

$$(x_2 - x_1) * (y_4 - y_3) - (x_4 - x_3) * (y_2 - y_1) \neq 0 \quad \text{и}$$

$$((x_2 - x_1) * (y_3 - y_1) - (x_3 - x_1) * (y_2 - y_1)) * ((x_2 - x_1) * (y_4 - y_1) - (x_4 - x_1) * (y_2 - y_1)) \leq 0.$$

Если обе точки ребра лежат на прямой, то получится бесконечное множество точек пересечения. Тогда оба векторных произведения достроенных векторов будут равны

нулю:

$$(x_3 - x_1) * (y_2 - y_1) - (x_2 - x_1) * (y_3 - y_1) = 0 \quad \text{и}$$

$$(x_4 - x_1) * (y_2 - y_1) - (x_2 - x_1) * (y_4 - y_1) = 0.$$

Чтобы понять, лежит ли некоторая точка с координатами (x_5, y_5) на прямой, построим вектор с концом в этой точке и началом, совпадающим с началом направляющего вектора прямой. Если лежит, то векторное произведение этого и направляющего вектора будет равно нулю:

$$(x_5 - x_1) * (y_2 - y_1) - (x_2 - x_1) * (y_5 - y_1) = 0$$

3. Используемые структуры данных и алгоритмы

В проект был добавлен класс `Line`, задача которого хранить в себе два объекта класса `R2Point`, представляющих две точки прямой, находить точку пересечения с ребром выпуклой оболочки с помощью метода `intersection` и определять, лежит ли конец ребра на прямой, с помощью метода `is_border`.

```
class Line:

    def __init__(self, p, q):
        self.p, self.q = p, q

    # Точка пересечения
    def intersection(self, c, d):
        if (self.q.x - self.p.x) * (c.y - d.y) - (c.x - d.x) \
                * (self.q.y - self.p.y) != 0 and \
                ((self.q.x - self.p.x) * (c.y - self.p.y)
                    - (c.x - self.p.x) * (self.q.y - self.p.y)) * \
                ((self.q.x - self.p.x) * (d.y - self.p.y)
                    - (d.x - self.p.x) * (self.q.y - self.p.y)) <= 0:
            return 1
        elif (c.x - self.p.x) * (self.q.y - self.p.y) == (c.y - self.p.y)\
                * (self.q.x - self.p.x) \
                and (d.x - self.p.x) * (self.q.y - self.p.y)\
                == (d.y - self.p.y) * (self.q.x - self.p.x):
            return math.inf
        else:
            return 0

    def is_border(self, a):
        if (a.x - self.p.x) * (self.q.y - self.p.y) == (a.y - self.p.y)\
                * (self.q.x - self.p.x):
            return True
```

В класс `Segment` добавлен метод `g`, возвращающий мощность множества точек пересечения прямой с отрезком.

```
def g(self):
    return self.fixed_line.intersection(self.p, self.q)
```

где `fixed_line` — атрибут класса `Figure`, представляющий две точки прямой:

```
Figure.fixed_line = Line(R2Point(), R2Point())
```

В класс `Polygon` также добавлен метод `g`, возвращающий мощность множества точек пересечения прямой с рёбрами, но само значение мощности вычисляется в изменённых методах `__init__` и `add`.

```
def g(self):
    return self.g
```

Изменения в `__init__` и `add`:

```
def __init__(self, a, b, c):

      #...

      if self.fixed_line.is_border(a) or self.fixed_line.is_border(b)\
              or self.fixed_line.is_border(c):
          self._g = self.fixed_line.intersection(a, b)\
              + self.fixed_line.intersection(b, c) + \
              self.fixed_line.intersection(c, a) - 1
      else:
          self._g = self.fixed_line.intersection(a, b)\
              + self.fixed_line.intersection(b, c) + \
              self.fixed_line.intersection(c, a)


# добавление новой точки
def add(self, t):

        # ...

        # удаление освещённых рёбер из начала дека
        p = self.points.pop_first()
        while t.is_light(p, self.points.first()):
            self._perimeter -= p.dist(self.points.first())
            self._area += abs(R2Point.area(t, p, self.points.first()))
            if self.fixed_line.is_border(p):
                self._g -= \
                    self.fixed_line.intersection(p, self.points.first())-1
            else:
                self._g -=\
                    self.fixed_line.intersection(p, self.points.first())
            p = self.points.pop_first()
            print(self.fixed_line.intersection(p, self.points.first()))
            print(self.fixed_line.intersection(p, self.points.first()))
        self.points.push_first(p)

        # удаление освещённых рёбер из конца дека
        p = self.points.pop_last()
```

```
        while t.is_light(self.points.last(), p):
            self._perimeter -= p.dist(self.points.last())
            self._area += abs(R2Point.area(t, p, self.points.last()))
            if self.fixed_line.is_border(p):
                self._g -=\
                    self.fixed_line.intersection(p, self.points.last()) - 1
            else:
                self._g -=\
                    self.fixed_line.intersection(p, self.points.last())
            p = self.points.pop_last()
        self.points.push_last(p)

        # добавление двух новых рёбер
        self._perimeter += t.dist(self.points.first()) + \
            t.dist(self.points.last())
        if self.fixed_line.is_border(t) and (
                self.fixed_line.is_border(self.points.first())
                or self.fixed_line.is_border(self.points.last())):
            self._g = math.inf
        elif self.fixed_line.is_border(t):
            self._g += 1
        else:
            self._g += self.fixed_line.intersection(t, self.points.last())\
             + self.fixed_line.intersection(t, self.points.first())
        if math.isnan(self._g) and self.fixed_line.is_border(t):
            self._g = math.inf
        if math.isnan(self._g):
            self._g = 2
        self.points.push_first(t)

    return self
```

Добавленные тесты:

```
class TestSegment:

    # Инициализация (выполняется для каждого из тестов класса)
    def setup_method(self):
        self.f = Segment(R2Point(0.0, 0.0), R2Point(1.0, 0.0))
        Figure.fixed_line = Line(R2Point(1.0, 1.0), R2Point(2.0, 2.0))

    # Функция `g` вычисляется корректно
    def test_g(self):
        assert self.f.g() == approx(1.0)

class TestPolygon:

    # Инициализация (выполняется для каждого из тестов класса)
    def setup_method(self):
        self.f = Polygon(
```

```
            R2Point(
                0.0, 0.0), R2Point(
                1.0, 0.0), R2Point(
                0.0, 1.0))
        Figure.fixed_line = Line(R2Point(1.0, 1.0), R2Point(2.0, 2.0))

    # Функция ‘g‘ вычисляется корректно для треугольника
    def test_g1(self):
        t = Segment(R2Point(1.0, 2.0), R2Point(2.0, 1.0))
        t = t.add(R2Point(3.0, 3.0))
        assert t.g() == approx(2.0)

    # Функция ‘g‘ вычисляется корректно для четырехугольника с вершиной на прямой
    def test_g3(self):
        t = Segment(R2Point(1.0, 2.0), R2Point(2.0, 1.0))
        t = t.add(R2Point(3.0, 3.0))
        t = t.add(R2Point(3.0, 4.0))
        assert t.g() == approx(2.0)

    # Функция ‘g‘ вычисляется корректно для квадрата с удалением ребра на прямой
    def test_g5(self):
        t = Segment(R2Point(1.0, 1.0), R2Point(2.0, 2.0))
        t = t.add(R2Point(1.0, 2.0))
        t = t.add(R2Point(2.0, 1.0))
        assert t.g() == approx(2.0)
```

Для изображения прямой в класс `TKDrawer` был добавлен метод `draw_infinite_line`:

```
def draw_infinite_line(self, p, q):
        self.canvas.create_line(x(p)+(x(p)-x(q)) * 600, y(p)+(y(p)-y(q))
                                * 600, x(q)+(x(p)-x(q))
                                * -600, y(q)+(y(p)-y(q)) * -600,
                                fill="red", width=2)
        self.root.update()
```

Список литературы и интернет-ресурсов

[1] `https://edu-support.mephi.ru/materials/230/edu/misc/russian.md?to=html` — Рекомендации по русскоязычному набору текста.

[2] `https://edu-support.mephi.ru/materials/214/edu/lectures/13/lecture__2.md?to=html` — Описание проекта «Выпуклая оболочка».

[3] `https://edu-support.mephi.ru/materials/214/edu/lectures/09/lect-29.md?to=html` — Теория индуктивных функций.

[4] `https://www.python.org/` — Официальный сайт языка Python.

[5] С.М. Львовский. *Набор и вёрстка в системе LaTeX, 3-е изд., испр. и доп.* — М., МЦНМО, 2003. Доступны исходные тексты этой книги.

[6] D. E. Knuth. *The TeXbook.* — Addison-Wesley, 1984. Русский перевод: Дональд Е. Кнут. *Все про TeX.* — Протвино, РДТеХ, 1993.