

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНДУСТРИАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВПО «МГИУ»)
КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА

по дисциплине «Методы хранения и обработки информации»
на тему «Вычисление количества пар вершин выпуклой оболочки,
расстояние между которыми не превосходит единицу. Нахождение
суммы длин рёбер, расстояние от середины и ровно одного из концов до
плоскости $x = 2$ строго меньше 1.»

Группа

2361

Студент

Г.А. Нахов

Руководитель работы
к.ф.-м.н., доцент

Е.А. Роганов

Москва 2014

Аннотация

Работа посвящена модификации проектов «Выпуклая оболочка» и «Изображение проекции полиэдра». В первом из этих проектов решалась задача на вычисление количества пар вершин выпуклой оболочки, расстояние между которыми не превосходит единицу. Во втором проекте вычислялась сумма длин рёбер, середина и ровно один из концов которых находятся на расстоянии строго меньше 1 от плоскости $x = 2$.

Содержание

1.	Введение	3
2.	Модификация проекта «Выпуклая оболочка»	3
3.	Модификация проекта «Изображение проекции полиэдра»	8
4.	Приложение А	12
5.	Приложение Б	13

1. Введение

Проект «Выпуклая оболочка» [1] решает задачу индуктивного перевычисления выпуклой оболочки последовательно поступающих точек плоскости и таких её характеристик, как периметр и площади выпуклой оболочки. Решение этой задачи требует знания теории индуктивных функций [2], основ аналитической геометрии, векторной алгебры и языка Ruby [3].

Проект «Изображение проекции полиэдра» [4] — пример классической задачи, для успешного решения которой необходимо знакомство с основами вычислительной геометрии. Для этого необходимы хорошее понимание ряда разделов аналитической геометрии и векторной алгебры, основ объектно-ориентированного программирования и языка Ruby.

В данной курсовой работе рассматриваются модификации двух эталонных проектов «Выпуклая оболочка» и «Изображение проекции полиэдра». Целями работы являются:

- 1) в проекте «Выпуклая оболочка» вычислить количество пар вершин выпуклой оболочки, расстояние между которыми не превосходит единицу;
- 2) в проекте «Изображение проекции полиэдра» вычислить сумму длин рёбер, середина и ровно один из концов которых находятся на расстоянии строго меньше 1 от плоскости $x = 2$.

Общее количество строк в рассмотренных проектах составляет около 0000000000000, из которых более 00000000000 были изменены или добавлены автором в процессе работы над задачами модификации.

2. Модификация проекта «Выпуклая оболочка»

Постановка задачи

Модифицируйте эталонный проект таким образом, чтобы вычислялось количество пар вершин выпуклой оболочки, расстояние между которыми не превосходит единицу. На начальном этапе работы программа запрашивает координаты вершин выпуклой оболочки, которая впоследствии индуктивно добавляет ее в выпуклую оболочку. Для решения нашей задачи необходимо выводить, помимо значений периметра и площади, количество пар вершин выпуклой оболочки, расстояние между которыми не превосходит единицу.

Для выполнения задания необходимо обладать базовыми знаниями в линейной алгебре. 2 точки, при их соединении, образуют отрезок, который можно представить в виде вектора. Чтобы получить длину (модуль) вектора \overrightarrow{AB} необходимо воспользоваться следующей формулой:

$$|\overrightarrow{AB}| = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}$$

Рассмотрим пример. Для последовательности точек $A(0, 0)$, $B(1, 0)$ и $C(0, 5)$ программа выводит 1 в качестве количества пар, расстояние между которыми не превосходит единицу. После того, как мы ввели следующую точку $D(1, 5)$, количество пар, расстояние между которыми не превосходит единицу, равняется 2 (рис. 1).

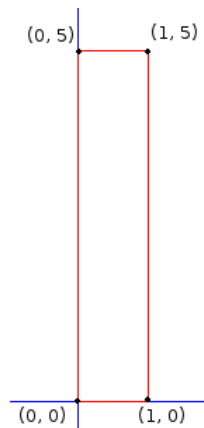


Рис. 1.

Все требуемые функции будут вписываться в файлы проекта `convex.rb` и `r2point.rb`.

Решение

Для выполнения задания необходимо вычислять расстояние между вершинами выпуклой оболочки. В эталонном проекте изначально присутствует функция `dist`, которая считает расстояние между вершинами выпуклой оболочки.

Модификация кода

При реализации кода были произведены изменения в четырех файлах эталонного проекта: `r2point.rb`, `convex.rb`, `run_convex.rb`, `run_tkconvex.rb`. Рассмотрим изменения кода в файле `r2point.rb`. В него был добавлен метод `distance`:

```
class R2Point
...
  def distance(a)
    return (self.dist(a)<=1)? 1 : 0
  end
...
```

Данный метод рассматривает две точки, и проверяет, не превышает ли расстояние между вершинами 1. Для этого используется метод `dist` класса `R2Point`. В случае, если расстояние меньше либо равно единице, то метод `distance` возвращает значение 1, в противном случае, возвращается 0. На этом модификация файла `r2point.rb` завершена.

Рассмотрим модификацию файла `convex.rb`. В классе `Segment` добавляется метод `dist`:

```
class Segment < Figure
  ...
  def dist
    return (@p.dist(@q) <= 1)? 1 : 0
  end
  ...
end
```

Этот метод проверяет длину двуугольника (отрезка). Если она больше единицы возвращается значение 0, иначе возвращается значение 1.

Затем редактируется класс `Polygon`. Первоначально при создании объекта класса `Polygon` создается треугольник. Необходимо при инициализации объекта посчитать расстояния между вершинами треугольника и узнать, сколько пар вершин, между которыми расстояние не более единицы.

```
class Polygon < Figure
  attr_reader :points, :perimeter, :area

  def initialize(a, b, c)
    ...
    @dist = a.distance(b) + b.distance(c) + c.distance(a)
  end
  ...
end
```

В последствии, при добавлении новых точек индуктивно вычисляется расстояние от добавляемой точки, до первой и последней вершин выпуклой обложки :

```
...
@perimeter += t.dist(@points.first) + t.dist(@points.last)
@dist += t.distance(@points.first) + t.distance(@points.last)
@points.push_first(t)
...
```

Как и при добавлении, так и при удалении вычисляется расстояние между вершинами, и если расстояние между парами вершин превышает 1, то удаляются из переменной.

```
p = @points.pop_first
while t.light?(p, @points.first)
  @perimeter -= p.dist(@points.first)
  @area      += R2Point.area(t, p, @points.first).abs
  @dist      -= p.distance(@points.first)
  p = @points.pop_first
end
@points.push_first(p)
```

```

...
p = @points.pop_last
while t.light?(@points.last, p)
  @perimeter -= p.dist(@points.last)
  @area      += R2Point.area(t, p, @points.last).abs
  @dist      -= p.distance(@points.last)
  p = @points.pop_last
end
@points.push_last(p)

```

Далее рассмотрим примеры работы программы. В случае, когда создается объект класса **Void** количество вершин выпуклой оболочки равно нулю. Когда добавляется вершина создается объект класса **Point**. В обоих случаях количество вершин оболочки меньше двух, считать расстояние между вершинами невозможно и возвращается число 0. При создании объектов классов **Segment** и **Polygon** количество вершин больше либо равно двум и необходимо считать расстояния между вершинами. Для точек с координатами $(0, 0)$ и $(1, 0)$ расстояние между которыми равно 1, следовательно, программа выведет 1 (рис. 2).

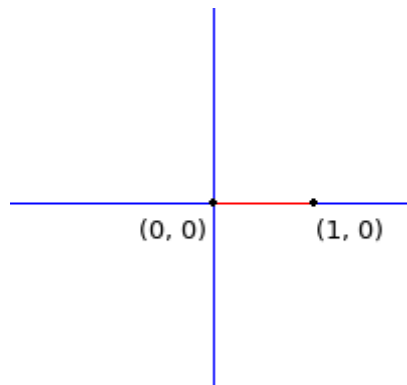


Рис. 2.

При добавлении новой точки $(0, 1)$, расстояние между всеми точками пересчитывается, и программа выводит результат 2. Так как расстояние между вершинами $(0, 0)$ и $(0, 1)$, $(0, 0)$ и $(1, 0)$ равно 1, а расстояние между вершинами $(0, 1)$ и $(1, 0)$ превосходит 1 (рис. 3).

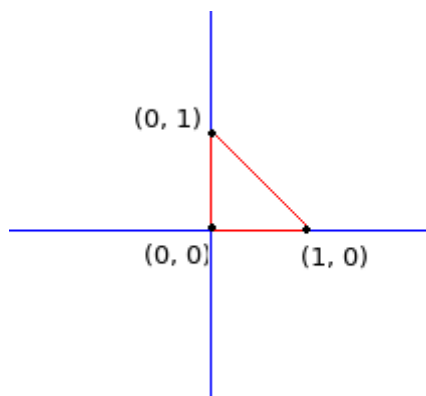


Рис. 3.

При добавлении точки $(1, 1)$, программа пересчитывает расстояния между точками и выводит ответ 4 (рис. 4).

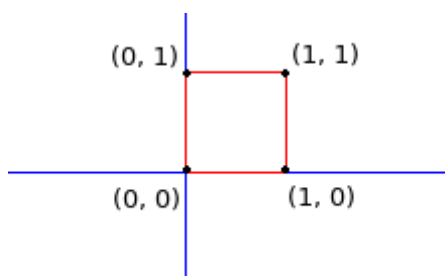


Рис. 4.

Далее добавляется новая точка $(1, 3)$. Точка $(1, 1)$ удаляется и выводится ответ 2 (рис. 5).

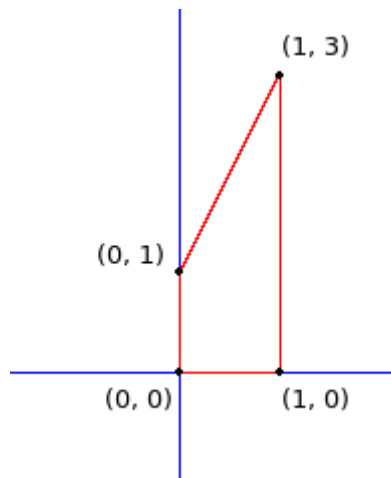


Рис. 5.

3. Модификация проекта «Изображение проекции полиэдра»

Постановка задачи

Модифицируйте эталонный проект таким образом, чтобы определялась и печаталась следующая характеристика полиэдра: сумма длин рёбер, середина и ровно один из концов которых находится на расстоянии строго меньше 1 от плоскости $x = 2$. Для выполнения задания необходимо анализировать расположение середины и вершин всех рёбер полиэдра относительно плоскости $x = 2$. Для этого модифицируем эталонный проект, добавив необходимые методы нахождения середины ребра, расстояния от точки до плоскости $x = 2$ и вычисления длины ребра.

Решение задачи и модификация кода

Чтобы определить, является ли точка хорошей, необходимо узнать находится ли координата x любой точки на расстоянии строго меньше единицы от прямой $x = 2$. Так как длина — величина положительная будем использовать метод `abs`, позволяющий получить модуль числа, к которому применяется этот метод. Для определения, является ли точка удовлетворяющей условию, в классе `R3` файла `common/polyedr.rb`, добавим метод `good?`:

```
class R3
  ...
  def good?
    if (2.0 - @x).abs < 1.0
      return true
    end
  end
end
```



```

    else
      return false
    end
  end
end
...

```

Ребро в нашем проекте является отрезком в пространстве, поэтому для нахождения середины ребра необходимо найти середину отрезка. Координаты середины отрезка в пространстве $[a, b]$ вычисляются по формуле:

$$x_c = \frac{a_x + b_x}{2}; \quad y_c = \frac{a_y + b_y}{2}; \quad z_c = \frac{a_z + b_z}{2}.$$

В классе `Edge`, добавим метод, `center` который создает новую точку координаты которой находятся по формулам, указанным выше:

```

class Edge
  ...
  def center
    xc=(@fin.x+@beg.x)/2.0
    yc=(@fin.y+@beg.y)/2.0
    zc=(@fin.z+@beg.z)/2.0
    return R3.new(xc,yc,zc)
  end
  ...

```

Так же добавим метод `length`, который возвращает длину ребра, где переменная `coef` является коэффициентом гомотетии.

```

class Edge
  ...
  def length
    Math.sqrt(((@fin.x-@beg.x)**2+(@fin.y-@beg.y)**2+(@fin.z-@beg.z)**2))/@coef
  end
  ...

```

Далее добавим метод `func` в классе `Polyedr` возвращающий сумму длин рёбер, середина и ровно один из концов которых — «хорошие» точки.

```

class Polyedr
  ...
  def func
    sum=0
    edges.each do |e|
      if e.center.good?(e.coef) && (e.beg.good?(e.coef) ^ e.fin.good?(e.coef))
        sum += e.length
      end
    end
  end
end

```

```

    return sum
end
...

```

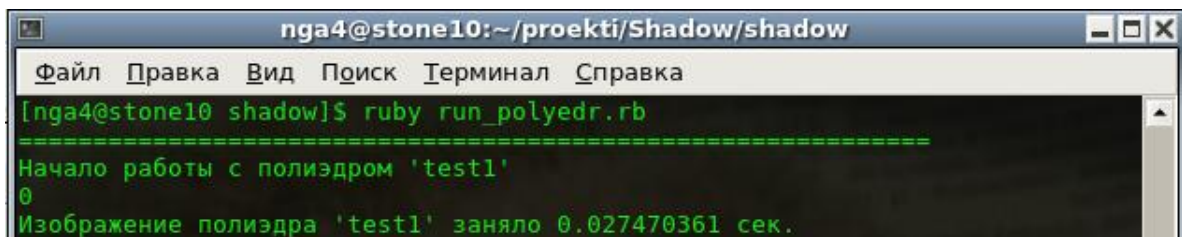
После этого остается модифицировать файл, запускающий программу.

```

#!/usr/bin/env ruby
# encoding: UTF-8
require './polyedr'
require '../common/tk_drawer'
TkDrawer.create
%w(test1 ).each do |name|
  puts '=====',
  puts "Начало работы с полиэдром '#{name}'"
  start_time = Time.now
  a=Polyedr.new("../data/#{name}.geom")
  a.draw
  puts a.func
  puts "Изображение полиэдра '#{name}' заняло #{Time.now - start_time} сек."
  print 'Hit "Return" to continue -> '
  gets
end

```

Модификация завершена. Работа программы с эталонным файлом test1.geom (рис. 6):



```

nga4@stone10:~/proekti/Shadow/shadow
Файл  Правка  Вид  Поиск  Терминал  Справка
[nga4@stone10 shadow]$ ruby run_polyedr.rb
=====
Начало работы с полиэдром 'test1'
0
Изображение полиэдра 'test1' заняло 0.027470361 сек.

```

Рис. 6.

Сам файл:

```

10.0 0.0 0.0 0.0
4 1 4
1.5 0.0 0.0
3.5 0.0 0.0
3.5 5.0 0.0
1.5 5.0 0.0
4 1 2 3 4

```

Список литературы и интернет-ресурсов

- [1] <http://edu.msiu.ru/files/19873-lecture.html> — Описание проекта «Выпуклая оболочка».
- [2] Е.А. Роганов *Основы информатики и программирования*. — М., МГИУ, 2002.
- [3] <http://ru.wikipedia.org/wiki/Ruby> — Википедия (свободная энциклопедия) о языке Ruby.
- [4] <http://edu.msiu.ru/files/26490-lecture.html> — Описание проекта «Изображение проекции полиэдра».
- [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 \TeX book*. — Addison-Wesley, 1984. Русский перевод: Дональд Е. Кнут. *Все про \TeX* . — Протвино, РД \TeX , 1993.
- [7] Е.А. Роганов, Н.Б. Тихомиров, А.М. Шелехов. *Математика и информатика для юристов*. — М., МГИУ, 2005. Доступны исходные тексты этой книги.

4. Приложение А

Модификация эталонного проекта `r2point.rb`:

```
> # расстояние между вершинами меньше 1?
> def distance(a)
>   if self.dist(a)<=1
>     1
>   else
>     0
>   end
> end
```

Модификация эталонного проекта `convex.rb`:

```
> def dist;      0    end

> def dist
>   if @p.dist(@q) <= 1
>     1
>   else
>     0
>   end
> end

< attr_reader :points, :perimeter, :area
---
> attr_reader :points, :perimeter, :area, :dist

> @dist = a.distance(b) + b.distance(c) + c.distance(a)

> @perimeter += t.dist(@points.first) + t.dist(@points.last)
> @dist += t.distance(@points.first) + t.distance(@points.last)
> @points.push_first(t)

> p = @points.pop_first
> while t.light?(p, @points.first)
>   @perimeter -= p.dist(@points.first)
>   @area      += R2Point.area(t, p, @points.first).abs
>   @dist      -= p.distance(@points.first)
>   p = @points.pop_first
> end
> @points.push_first(p)

> p = @points.pop_last
```

```

> while t.light?(@points.last, p)
>   @perimeter -= p.dist(@points.last)
>   @area      += R2Point.area(t, p, @points.last).abs
>   @dist      -= p.distance(@points.last)
>   p = @points.pop_last
> end
> @points.push_last(p)
>

```

5. Приложение Б

Модификация эталонного проекта `common/polyedr.rb`:

```

> def good?(coef=1.0) #проверка точки на "хорошесть" (добавлено)
>   if (2.0 - @x).abs < 1.0 #если расстояние до прямой x=2 строго меньше 1, то вернется true
>     return true
>   else
>     return false
>   end
> end

> def center
>   xc=(@fin.x+@beg.x)/2.0
>   yc=(@fin.y+@beg.y)/2.0
>   zc=(@fin.z+@beg.z)/2.0
>   return R3.new(xc,yc,zc)
> end

> def func
>   sum=0
>   edges.each do |e|
>     if e.center.good?(e.coef) && (e.beg.good?(e.coef) ^ e.fin.good?(e.coef))
>       sum += e.length
>     end
>   end
>   return sum
> end

```

Модификация эталонного проекта shadow/run_polyedr.rb:

```
> #!/usr/bin/env ruby
> # encoding: UTF-8
> require './polyedr'
> require '../common/tk_drawer'
> TkDrawer.create
> %w(test1 ).each do |name|
>   puts '=====',
>   puts "Начало работы с полиэдром '#{name}'"
>   start_time = Time.now
>   a=Polyedr.new("../data/#{name}.geom")
>   a.draw
>   puts a.func
>   puts "Изображение полиэдра '#{name}' заняло #{Time.now - start_time} сек."
>   print 'Hit "Return" to continue -> '
>   gets
> end
```