Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Курс «Алгоритмика»

Вторая лабораторная работа

Построение медианы множества точек

Работу выполнил
студент **Сюй Минчуань**

2021

Содержание

1 Задачу о построении медианы множества точек

1.1 Постановка задачи

Глубиной $D(p)$ **точки** p в конечном множестве S из n точек на евклидовой плоскости называется число выпуклых оболочек (выпуклых слоев), которые должны быть удалены из S прежде, чем будет удалена точка p.

Глубиной $M(S)$ **множества** S называется максимум глубины точек, входящих в S. Очевидно $M(S) \leqslant [\frac{n}{3} - 1]$. Одна из точек, имеющих глубину $M(S)$, называется **медианой** множества точек S.

Пусть $S_m = \{p : D(p) = m, p \in S\}$ – множество точек глубины m. **Функцией глубин** S называется $F(m) = |S_m|, m = 0, 1, \ldots, M(S)$ – количество точек в S, имеющих глубину m.

Даны координаты городов России, и в этой задаче:

1. Нужно разработать алгоритм и реализовать программу для определения функции глубины для множества городов России.
2. Нужно вычислить функцию глубины и найти медианный город России.

1.2 Описание данных

На примере данных (1) видим, что для каждого города России представлены их координаты по северной широте и восточной долготе. Бывают два вида координат:

1. Например, у города Абакана иметь координаты 53°43 с.ш. и 91°26 в.д. – это точка на евклидовой плоскости с координатами $x = 91 + 26/60 = 91.43, y = 53 + 43/60 = 53.72$.
2. А у города Арзамаса иметь координаты 55.38 с.ш. и 43.87 в.д. – это точка на евклидовой плоскости с координатами $x = 43.87, y = 55.38$.

Понятно, что нужно преобразовать координаты первого типа во вторую, с которой удобно работать. В связи с этим написал функцию для преобразования координат. Кроме координатных данных ещё признак (num_people) по численности населения города, но для поставленной задачи эти данные нам не нужны.

```
[2]: city_data = pd.read_csv("citys.csv", delimiter = ",", encoding = 'UTF-8')
     city_data.head()
```

[2]:

	CITY	y	x	num_people
0	Абакан	53°43′	91°26′	187
1	Альметьевск	54°54′	52°18′	158
2	Ангарск	52°34′	103°55′	225
3	Арзамас	55.38	43.87	104
4	Армавир	45°00′	41°07′	189

Рис. 1: Пример исходных данных

```
[4]: city_data = process_coordiates(city_data, ['x','y'])
     city_data.head()
```

[4]:

	CITY	y	x	num_people
0	Абакан	53.716667	91.433333	187
1	Альметьевск	54.9	52.3	158
2	Ангарск	52.566667	103.916667	225
3	Арзамас	55.38	43.87	104
4	Армавир	45.0	41.116667	189

Рис. 2: Пример данных после преобразования

```
[5]: city_data[city_data['CITY'] == 'Москва']
```

[5]:

	CITY	y	x	num_people
67	Москва	55.75	37.616667	12678

Рис. 3: Координаты города Москвы

2 Метод решения, анализ и программная реализация алгоритма

2.1 Описание метода решения

Для нахождения выпуклой оболочки использовался **алгоритм Джарвиса** (или алгоритм заворачивания подарков). Он имеет сложность $\mathcal{O}(nh)$, где n - число всех точек, h - число точек в выпуклой оболочке.

В общем, алгоритм описывается следующим образом: Пусть $p[n]$ - массив точек множества.

Jarvismarch(p):

1. $p[1]$ = самая левая нижняя точка множества S;
2. $p[2]$ = соседняя точка от $p[1]$ справа (Выбирается ребро с минимальным углом относительно оси абсцисс)
3. $i = 2$;
4. **do:**
   (a) **for** для каждой точки j от 1 до |p|, кроме уже попавших в выпуклую оболочку, но включая p[1]:
   p[i+1] = точка, образующая минимальный косинус с прямой p[i-1]p[i].
   (b) i = i + 1.
   while p[i] != p[1]
5. **return** p;

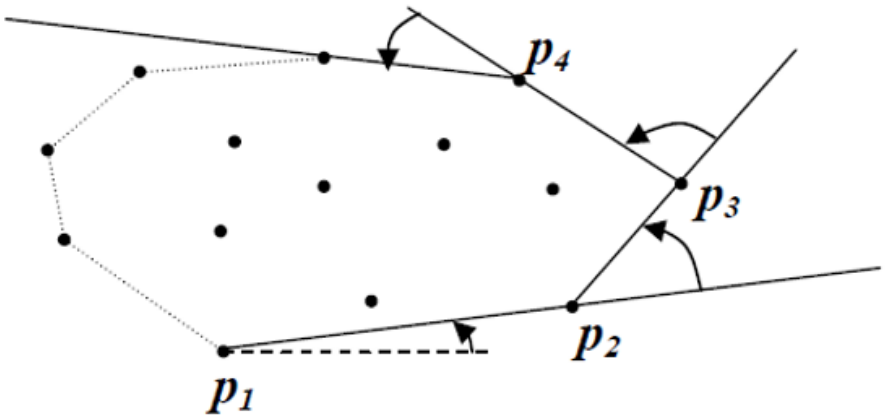

Рис. 4: Иллюстрация алгоритма Джарвиса

2.2 Описание программной реализации

Алгоритм Джарвиса реализовал сам без использования библиотечных функций. Программная реализация на Python представлена в .ipynb файле. Для запуска программы необходимо установлен Jupyter Notebook. Ниже приведены описания функции:

1. **def process_coordiates(data, coor_name_list):**
   функция преобразования координат городов. На вход исходные данные в виде таблицы и координаты, которые нужны преобразовать. На выход преобразованные данные.
2. **def find_next(H, points, p_map):**
   функция для нахождения следующей точки в выпуклой оболочке. На вход текущий массив точек выпуклой оболочки, координаты, и отображение индексов для правильного считывание координатных данных (ведь при удалении точек из множества, индексы координат в матрице могут быть изменены). На выход флаг для нахождения следующей точки и новый массив точек выпуклой оболочки.
3. **def jarvismarch(points, p_map):**
   Алгоритм Джарвиса. На вход координаты и отображение индексов, на выход массив точек построенной выпуклой оболочки (с учетом порядок)
4. **def depth_of_point(point_ix, points):**
   функция для вычисления глубины одной города. На вход город и данные координат, на выход глубина города и соответствующий список построенных выпуклых оболочек.
5. **def max_depth_of_set(points):**
   функция для высичления глубины множества. На вход данные координат, на выход глубина множества и множества городов (медианы), которые имеют глубину множества.
6. **def F(m, points):**
   функция глубин множества. На вход значение глубины и данные координат, на выход количество городов, имеющих заданную глубину, и все массива точек с разными глубинами.

2.3 Результаты поставленной задачи

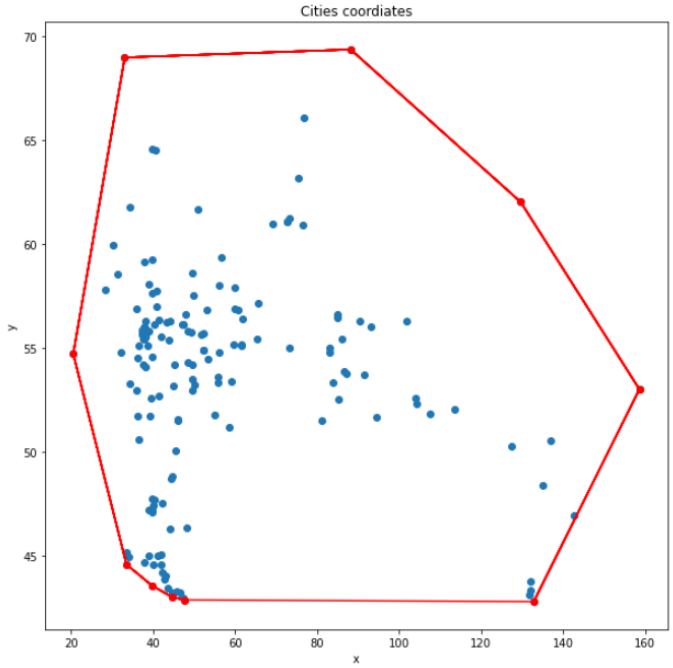

Рис. 5: Первая выпуклая оболочка

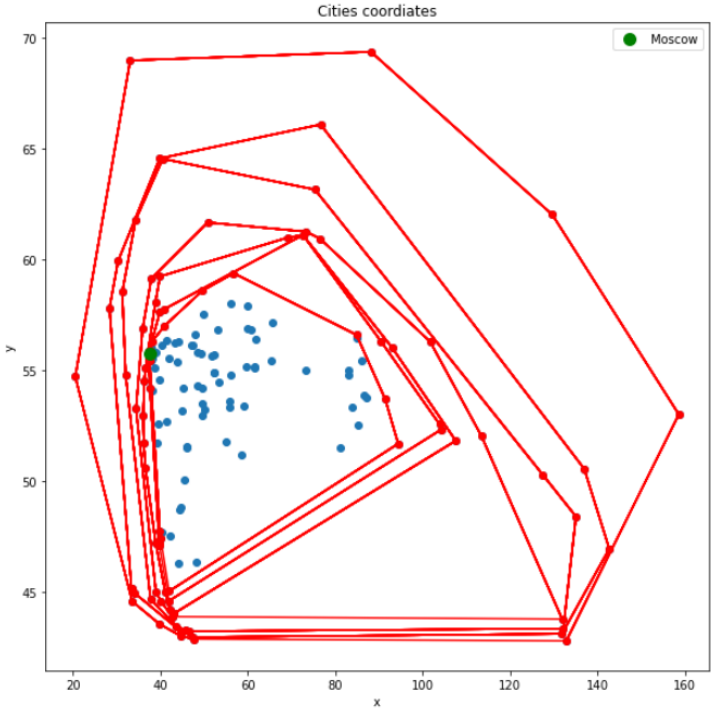

Рис. 6: Выпуклые оболочки для Москвы (с глубиной $M = 7$)

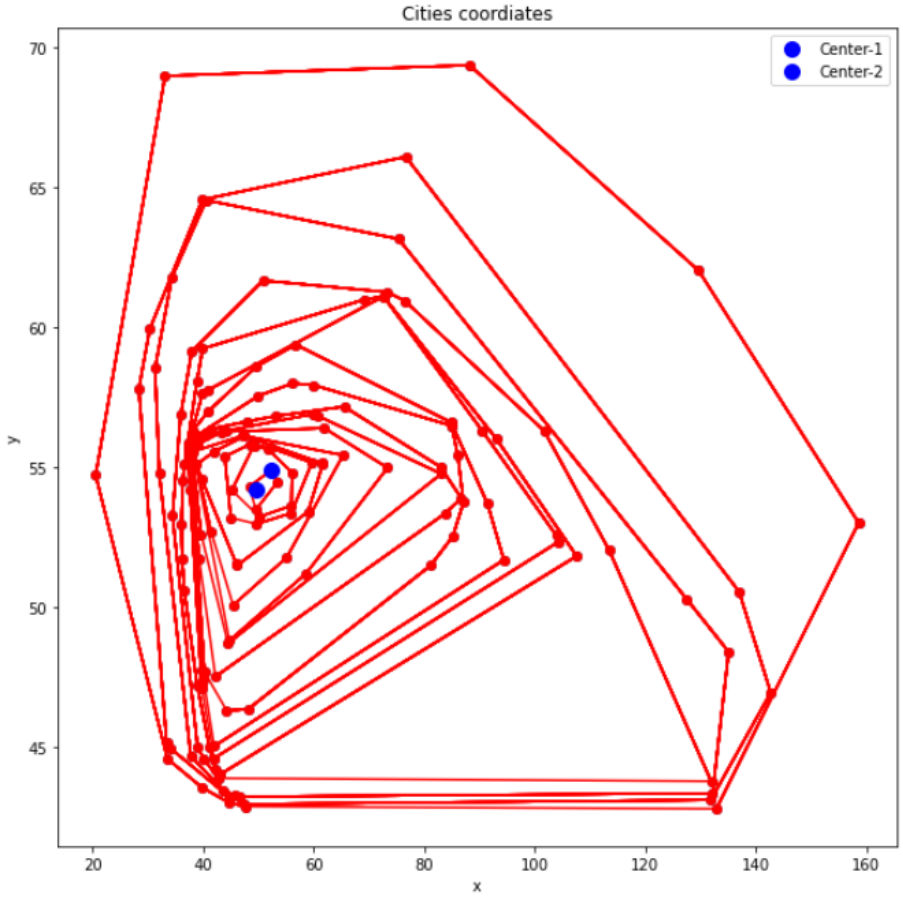

Рис. 7: Медианные города России

```
[16]: # Раньше уже подсчитал, что max_depth = 16
      S_m, M = F(16, city_xy)
      print(S_m[16], M)
```

```
[29, 155] 2
```

```
[17]: city_data.iloc[['29','155']]
```

[17]:

	CITY	y	x	num_people
29	Димитровград	54.22	49.63	113
155	Электросталь	54.9	52.3	156

Рис. 8: Медианные города России - это Димитровград и Электросталь