Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных сетей и вычислительных средств

Курсовая работа

по дисциплине «Моделирование распределенных систем»

на тему «Тепловая карта для каждого абонента в зависимости от удаленности от БС»

Выполнили:

Студенты гр. ИА-831 и ИА-832

Зарубин М.Е.

Юшкевич И.А.

Соснин Д.Н.

Проверила:

доцент кафедры ТС и ВС

Дроздова В.Г.

Новосибирск 2020

Содержание

1. Текст задания к курсовой работе

1. (https://habr.com/ru/post/324596/) - разобраться с этим проектом и превратить в конфетку для курсового проекта.
2. В качестве "теплоты" карты использовать дальность от базовой(ых) станции(й). Отметить на карте картинкой\схемой\маркером.
3. Добавить несколько базовых станций на разной удаленности (минимум 800 метров в радиусе) друг от друга.
4. "Теплота" карты определяется как среднее арифметическое от дальности до всех базовых станций.
5. Составить отчет (с документированным кодом). Код должен быть в Приложении (в конце отчета).

2. Проделанная работа

2.1. Создание базовых станций

Мы выбрали 6 базовых станций на расстоянии 800 и более метров, взяли их координаты и пометили их изображениями флажков. Сделали всё это благодаря маркерам. Также координаты записали в отдельный файл, для дальнейшего использования.

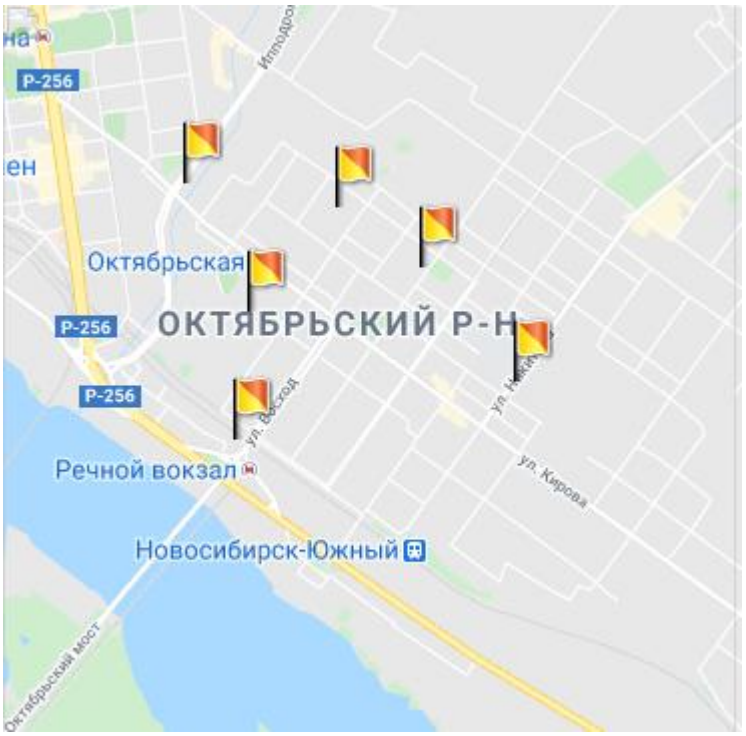

Рис.1. Размещённые станции на карте Google

2.2. Выборка точек для создания изображения

Для реализации области и выборки точек на ней мы сделали программу на языке python. Запуск происходит с двумя аргументами: файл с записанными координатами станций и файл, куда будем записывать координаты точек и их расстояние до ближайшей станции. Первый столбец расстояние, остальные два – координаты.

Для начала выбираем середину нашего изображения и в программе вписываем её координаты и расстояние до границ. С помощью этого определяем координаты границ. После проходим по всем точкам, определяя их дистанцию до ближайшей станции и записываем всё в файл.

Также реализован help для пользователей, тогда просто надо вместо аргументов передать слово help и всё.

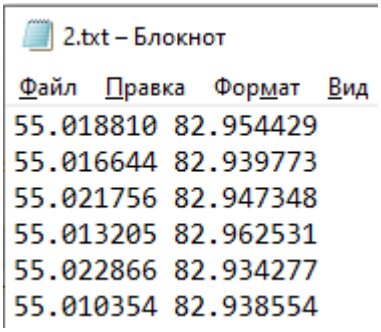

Рис.2. Содержание файла с координатами базовых станций

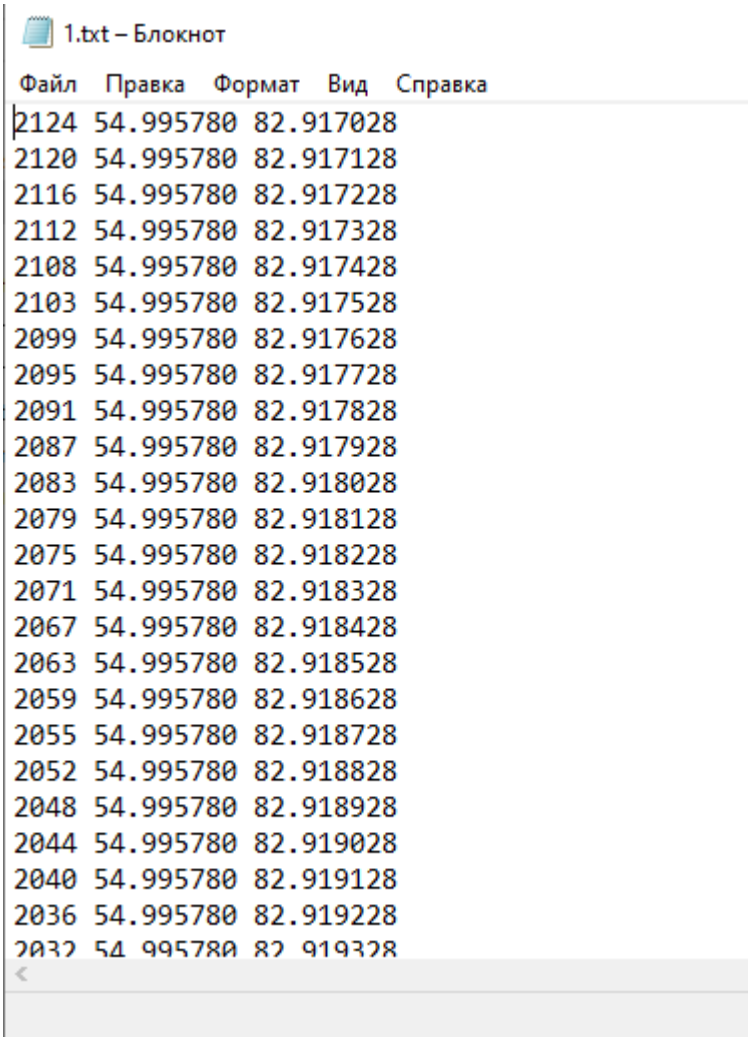

Рис.3. Содержание файла с расстояниями и координатами точек

2.3. Прорисовка изображения

Для прорисовки изображения мы также создали программу на python. В ней мы запускаем файл, в который записывали координаты точек и расстояние

до ближайшей станции, и благодаря этому расстоянию мы определяем какой цвет присвоить точке. Мы сами записали определённое количество цветов и с помощью циклов сделали выборку цвета.

Именно создание изображение сделали с помощью библиотеки PIL. Там использовали некоторые фильтры, например, размытие изображения, чтобы границы между цветами были меньше видны и его поворот.

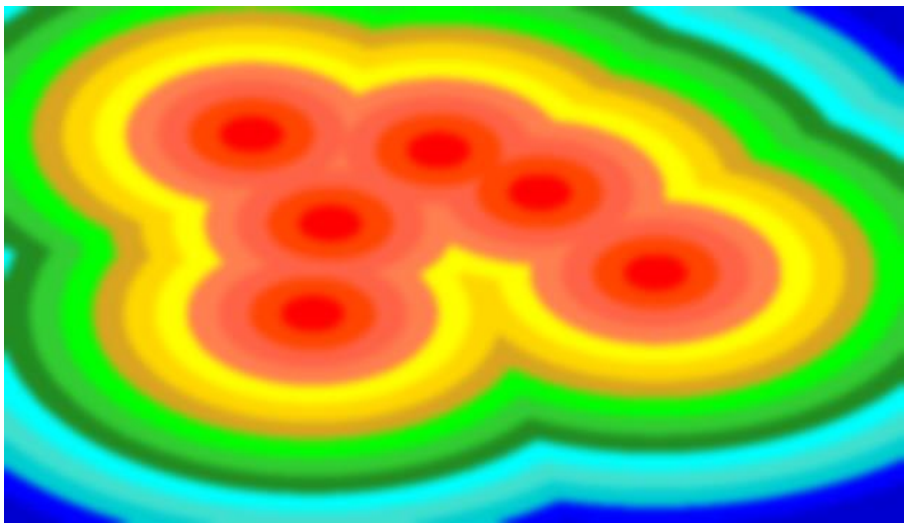

Рис.4. Полученное изображение

2.4. Подключение изображения к карте

В html файле реализовали подключение изображения к карте. Сделали его полупрозрачном, чтобы были видные наши станции и, что находится на карте.

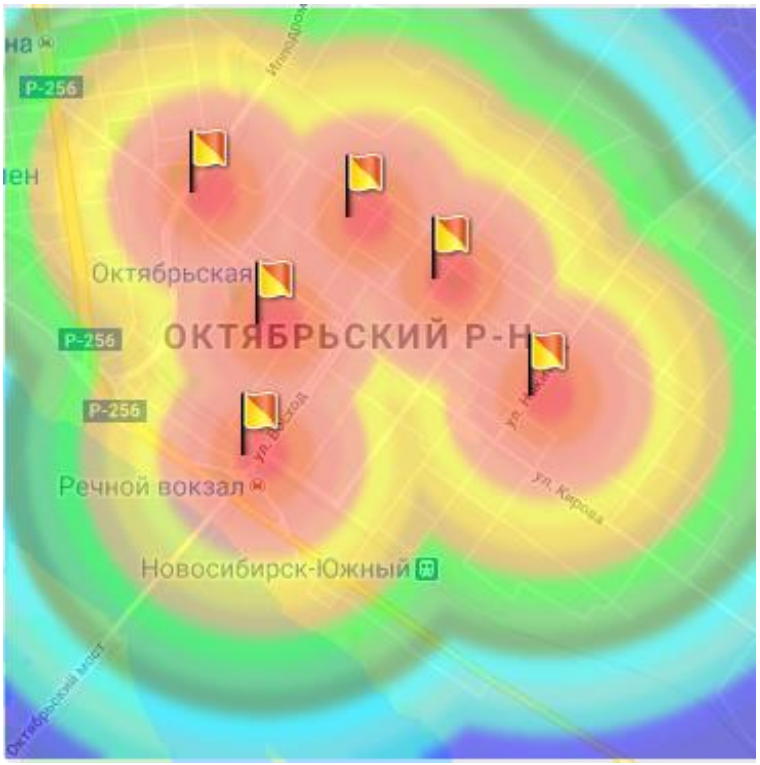

Рис.5. Добавленное изображение на карту Google

2.5. Дополнительное задание для защиты

Рис.6. Изображение до нажатия на карту

Рис.7. Изображение после нажатия на карту

3. Вывод

Для реализации полноценной тепловой карты нужно очень множество функций. Есть разные варианты реализации, но мы посчитали, что подключения изображения на карту будет одним из самых лучших. Также научились работать в javascript с картами и в python с разными библиотеками, например, как PIL.

4. Список используемой литературы

1. Статистика по стоимости недвижимости - визуализация на карте [Электронный ресурс]. Режим доступа: [https://habr.com/ru/post/324596], свободный – (05.01.2021).
2. Pillow обработка изображений в Python на примерах [Электронный ресурс]. Режим доступа: [https://python-scripts.com/pillow], свободный – (05.01.2021).
3. Maps JavaScript API [Электронный ресурс]. Режим доступа: [https://developers.google.com/maps/documentation/javascript/overview?hl=ru], свободный – (05.01.2021).

5. Приложение

Distant.py:

```
import math

from time import sleep

import sys

import os

def deg2rad(degrees):

    return math.pi*degrees/180.0

def rad2deg(radians):

    return 180.0*radians/math.pi

WGS84_a = 6378137.0  # Major semiaxis [m]

WGS84_b = 6356752.3  # Minor semiaxis [m]

def WGS84EarthRadius(lat):

    An = WGS84_a*WGS84_a * math.cos(lat)

    Bn = WGS84_b*WGS84_b * math.sin(lat)

    Ad = WGS84_a * math.cos(lat)

    Bd = WGS84_b * math.sin(lat)

    return math.sqrt( (An*An + Bn*Bn)/(Ad*Ad + Bd*Bd) )


def boundingBox(latitudeInDegrees, longitudeInDegrees, halfSideInKm):

    lat = deg2rad(latitudeInDegrees)

    lon = deg2rad(longitudeInDegrees)

    halfSide = 1000*halfSideInKm


    radius = WGS84EarthRadius(lat)

    pradius = radius*math.cos(lat)


    latMin = lat - halfSide/radius

    latMax = lat + halfSide/radius
```

```
    lonMin = lon - halfSide/pradius
    lonMax = lon + halfSide/pradius
    return (rad2deg(latMin), rad2deg(lonMin), rad2deg(latMax), rad2deg(lonMax))
MIN_LAT, MIN_LNG, MAX_LAT, MAX_LNG = boundingBox(55.013787, 82.948433, 2)
with open("min_max.txt", "w") as file:
    file.write(str(MIN_LAT) + ' ' + str(MIN_LNG) + ' ' + str(MAX_LAT) + ' ' + str(MAX_LNG))
def distance(LAT1, LNG1, LAT2, LNG2):
    radius = 6372795
    lat1 = LAT1 * math.pi / 180
    lat2 = LAT2 * math.pi / 180
    lng1 = LNG1 * math.pi / 180
    lng2 = LNG2 * math.pi /180

    cl1 = math.cos(lat1)
    cl2 = math.cos(lat2)
    sl1 = math.sin(lat1)
    sl2 = math.sin(lat2)

    delta = lng2 - lng1
    cdelta = math.cos(delta)
    sdelta = math.sin(delta)

    y = math.sqrt(math.pow(cl2 * sdelta, 2) + math.pow(cl1 * sl2 - sl1 * cl2 * cdelta, 2))
    x = sl1 * sl2 + cl1 * cl2 * cdelta
    ad = math.atan2(y, x)
    dist = ad * radius
    return dist
```

```
def create_file(filename1, filename2):
    processing = int(((MAX_LAT - MIN_LAT) / 0.0001) * ((MAX_LNG - MIN_LNG) / 0.0001) / 100)
    persent = 1
    x = MIN_LAT
    y = MIN_LNG
    x_coord = []
    y_coord = []
    minimum = 10000
    maximum = -1
    iter = 0
    file = open(filename1, 'w+')
    file_xy = open(filename2, 'r')
    base_station_x = []
    base_station_y = []
    with open(filename2) as inf:
        for line in inf:
            if not line[0].isdigit():
                continue

            x_coordinate, y_coordinate = line.strip().split()
            base_station_x.append(float(x_coordinate))
            base_station_y.append(float(y_coordinate))
    check_x = 0
    check_y = 0
    iter = 0
    print('[', end='')
    while x < MAX_LAT:
        y = MIN_LNG
```

```
        while y < MAX_LNG:
            minimum_dist_touch = 100000
            maximum = -1
            for i in range(len(base_station_x)):
                d = distance(x,  y, base_station_x[i], base_station_y[i])
                if d < minimum_dist_touch:
                    minimum_dist_touch = d
                if d > maximum:
                    maximum = d
            file.write('%s %.9s %.9s\n' % (int(minimum_dist_touch), x, y))
            x_coord.append(x)
            y_coord.append(y)
            y += 0.0001
            check_y += 1
            iter += 1
            if iter > processing:
                print('', end='\r')
                print(str(persent) + '%', end='')
                persent += 1
                iter = 0
        x += 0.0001
        check_x += 1
    if persent  >= 100:
        print('file completed successfully')
    with open("size_image.txt", 'w') as file:
        file.write(str(check_x) + ' ' + str(int(check_y/check_x)))


if __name__ == '__main__':
    if sys.argv[1] == 'help':
```

```
        print("В качестве параметров передать названия файлов")
        print('1. Куда записывать данные (файл будет перезаписан)\n2. Откуда брать координаты базовых станций')
    else:
        create_file(sys.argv[1], sys.argv[2])
```

Drawing.py:

```
from PIL import Image, ImageFilter
import sys
def run(filename):
    MIN_LAT = 0; MIN_LNG = 0
    MAX_LAT = 0; MAX_LNG = 0
    MAX_X = 0; MAX_Y = 0
    colors = []

    distant_lat_lng = []

    color_rgb = [
        (255, 0, 0), # RED
        (255, 69, 0), # Orange red
        (255, 99, 71), # Tomato
        (255, 127, 80), # Coral
        (255, 255, 0), # Yellow
        (255, 215, 0), # Gold
        (218, 165, 32), # Goldenrod
        (0, 255, 0), # Green
        (50, 205, 50), # Lime green
        (34, 139, 34), # Forest green
        (0, 255, 255), # Cyan
        (64, 224, 208),
```

```
        (0, 206, 209),
        (0, 0, 255), # Blue
        (0, 0, 205) # Medium Blue
    ]
    with open('min_max.txt', 'r') as file:
        for line in file:
            if not line[0].isdigit():
                continue
            MIN_LAT, MIN_LNG, MAX_LAT, MAX_LNG = line.strip().split()
    with open(filename, 'r') as inf:
        for line in inf:
            if not line[0].isdigit():
                continue
            dist, x_coordinate, y_coordinate = line.strip().split()
            distant_lat_lng.append([int(dist),            float(x_coordinate),
float(y_coordinate)])


    with open('size_image.txt', 'r') as file:
        for line in file:
            if not line[0].isdigit():
                continue
            MAX_X, MAX_Y = line.strip().split()
    MAX_X = int(MAX_X)
    MAX_Y = int(MAX_Y)
    def color(val):
        if val <= 140:
            return color_rgb[0]
        elif val <= 280:
            return color_rgb[1]
```

```
    elif val <= 420:
        return color_rgb[2]
    elif val <= 560:
        return color_rgb[3]
    elif val <= 700:
        return color_rgb[4]
    elif val <= 840:
        return color_rgb[5]
    elif val <= 980:
        return color_rgb[6]
    elif val <= 1120:
        return color_rgb[7]
    elif val <= 1260:
        return color_rgb[8]
    elif val <= 1400:
        return color_rgb[9]
    elif val <= 1540:
        return color_rgb[10]
    elif val <= 1680:
        return color_rgb[11]
    elif val <= 1820:
        return color_rgb[12]
    elif val <= 1960:
        return color_rgb[13]
    elif val > 1960:
        return color_rgb[14]
image = Image.new("RGBA", (MAX_X, MAX_Y))

IM = image.load()
```

```
    iter = 0
    p = 0
    processing = int((MAX_X * MAX_Y) / 100)
    present = 1
    for x in range(MAX_X):
        for y in range(MAX_Y):
            IM[x, y] = color(distant_lat_lng[iter][0])
            iter += 1
            p += 1
            if p >= processing:
                print('', end='\r')
                print(str(present) + '%', end='')
                present += 1
                p = 0
    rotate = image.rotate(90, expand=True)
    blured = rotate.filter(ImageFilter.GaussianBlur(3))
    blured.save("HeatMap.png", "PNG")
if __name__ == '__main__':
    if sys.argv[1] == 'help':
        print("***Программа для создания тепловой карты***")
        print("-В качестве опции передать имя файла в котором записанны данные,
необходимые для построения рисунка")
    else:
        run(sys.argv[1])
```

test.html:

```
<!DOCTYPE html>
<html>
      <head>
            <title>Google Maps</title>
```

```
            <meta content="author" value="4X_Pro" />
            <meta charset="utf-8" />
            <script                                    type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC_GTGGge8SF-
TlI8A15JsVescUBu1jexw&map_ids=1234&libraries=visualization"></script>
      </head>
      <body onload="initialize()">
            <div id="map_canvas" style="width:100%; height:600px"></div>
            <button id="map_left">← Влево</button>
            <button id="map_right">Вправо →</button>
            <script type="text/javascript">


                  var overlay = null;
                 var map;
                   var  swBound  =  new  google.maps.LatLng(54.99578028169929,
82.91702845149706);
  var neBound = new google.maps.LatLng(55.031793718300705, 82.97983754850291);
  var currentOverlayImg;
  TQOverlay.prototype = new google.maps.OverlayView();
function initialize() {
      var latlng = new google.maps.LatLng(55.014879, 82.948738);
    var myOptions = {
      zoom: 13,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    map    =    new    google.maps.Map(document.getElementById("map_canvas"),
myOptions);
     this.station = new Array;
```

```
                    this.pictures                                                =
'https://developers.google.com/maps/documentation/javascript/examples/full/im
ages/beachflag.png'
                    this.station[0] = new google.maps.Marker({
                      position: new google.maps.LatLng(55.018810, 82.954429),
                      map: this.map,
                      title: "Базовая станция 1",
                      icon: this.pictures
                    });
                    this.station[1] = new google.maps.Marker({
                      position: new google.maps.LatLng(55.016644, 82.939773),
                      map: this.map,
                      title: "Базовая станция 2",
                      icon: this.pictures
                    });
                    this.station[2] = new google.maps.Marker({
                      position: new google.maps.LatLng(55.021756, 82.947348),
                      map: this.map,
                      title: "Базовая станция 3",
                      icon: this.pictures
                    });
                    this.station[3] = new google.maps.Marker({
                      position: new google.maps.LatLng(55.013205, 82.962531),
                      map: this.map,
                      title: "Базовая станция 4",
                      icon: this.pictures
                    });
                    this.station[4] = new google.maps.Marker({
                      position: new google.maps.LatLng(55.022866, 82.934277),
                      map: this.map,
```

```
                        title: "Базовая станция 5",
                        icon: this.pictures
                      });
                      this.station[5] = new google.maps.Marker({
                        position: new google.maps.LatLng(55.010354, 82.938554),
                        map: this.map,
                        title: "Базовая станция 6",
                        icon: this.pictures
                      });
    updateMapImage();
  }
 function updateMapImage() {

    var bounds = new google.maps.LatLngBounds(swBound, neBound);
    var srcImage = "./ImagesToOverlay/apts1.txt.phantom.1000-2.png"; /*
url_base + ".png";*/
    overlay = new TQOverlay(bounds, srcImage, map);
  }
  function TQOverlay(bounds, image, map) {
    this.bounds_ = bounds;
    this.image_ = image;
    this.map_ = map;
    this.div_ = null;
    this.setMap(map);
  }
  TQOverlay.prototype.onAdd = function() {

    var div = document.createElement('DIV');
    div.style.border = "none";
```

```
    div.style.borderWidth = "0px";
    div.style.position = "absolute";
    currentOverlayImg = document.createElement("img");

    currentOverlayImg.src = this.image_;
    currentOverlayImg.style.width = "100%";
    currentOverlayImg.style.height = "100%";

    currentOverlayImg.style.opacity = .5;
    currentOverlayImg.style.filter = 'alpha(opacity=50)';

    div.appendChild(currentOverlayImg);
    this.div_ = div;
    var panes = this.getPanes();
    panes.overlayLayer.appendChild(div);
  }
  TQOverlay.prototype.draw = function() {
    var overlayProjection = this.getProjection();

    var sw =
overlayProjection.fromLatLngToDivPixel(this.bounds_.getSouthWest());
    var ne =
overlayProjection.fromLatLngToDivPixel(this.bounds_.getNorthEast());

    var div = this.div_;
    div.style.left = sw.x + 'px';
    div.style.top = ne.y + 'px';
    div.style.width = (ne.x - sw.x) + 'px';
    div.style.height = (sw.y - ne.y) + 'px';
  }
```

```
  TQOverlay.prototype.onRemove = function() {
    this.div_.parentNode.removeChild(this.div_);
    this.div_ = null;
  }
            </script>
      </body>
</html>
```