1. 实现 GeoHash 编码函数

算法原理其实就是二分法 左侧记 0 右侧记 1 然后使用 base32 进行编码即可

代码：

```
'''
Author: M1tsuha
Date: 2024-05-06 14:02:49
LastEditors: M1tsuha
LastEditTime: 2024-05-06 15:00:38
FilePath: \prac3_506\geohash.py
Description:

Copyright (c) 2024 by 1300935620@qq.com, All Rights Reserved.
'''

def encode_geohash(longitude, latitude, precision):
    """
    将经纬度编码为指定精度的 GeoHash 字符串。

    参数:
    longitude - 经度，浮点数。
    latitude - 纬度，浮点数。
    precision - 编码精度，即生成的 GeoHash 字符串的长度。

    返回值:
    编码后的 GeoHash 字符串。
    """
    base32_map = "0123456789bcdefghjkmnpqrstuvwxyz"  # base32 编码字符集
    min_lat, max_lat = -90.0, 90.0  # 纬度的范围
    min_lon, max_lon = -180.0, 180.0  # 经度的范围
    geohash = []  # 用于存储生成的 GeoHash 字符串
    bit = 0  # 当前处理的 bit 位
    ch = 0  # 用于存储 5 个 bit 位转换后的 base32 字符的索引
    bit_length = 0  # 已处理的 bit 位长度

    while len(geohash) < precision:  # 循环直到达到指定的精度
        if bit_length % 2 == 0:  # 处理偶数位，编码经度
            mid = (min_lon + max_lon) / 2
            if longitude > mid:  # 如果当前经度在二分范围右侧
                ch |= 1 << (4 - bit)
                # ch 初始化为 00000 位于二分右侧则当前位记 1
                # 即左移 4-bit 位 标记对应的 bit 位
```

```
                min_lon = mid  # 更新经度范围的左边界

            else:  # 经度在二分范围左侧
                max_lon = mid  # 更新经度范围的右边界

        else:  # 处理奇数位，编码纬度
            mid = (min_lat + max_lat) / 2
            if latitude > mid:  # 如果当前纬度在二分范围右侧
                ch |= 1 << (4 - bit)  # 标记对应的bit 位
                min_lat = mid
            else:
                max_lat = mid

        bit += 1  # 移动到下一个bit 位
        bit_length += 1  # 更新已处理的bit 位长度

        if bit == 5:  # 每5 个bit 位转换为一个base32 字符
            geohash.append(base32_map[ch])  # 添加base32 字符到结果中
            bit = 0  # 重置bit 位计数器
            ch = 0  # 重置base32 索引

    return ''.join(geohash)  # 将结果中的字符连接成字符串

# 计算不同长度的GeoHash 编码
coords = (115.83122,37.49867)  # 经度，纬度
lengths = [8, 9, 10]

for length in lengths:
    print(f'{length}-bit GeoHash: {encode_geohash(coords[0], coords[1],
length)}')
```

输出：

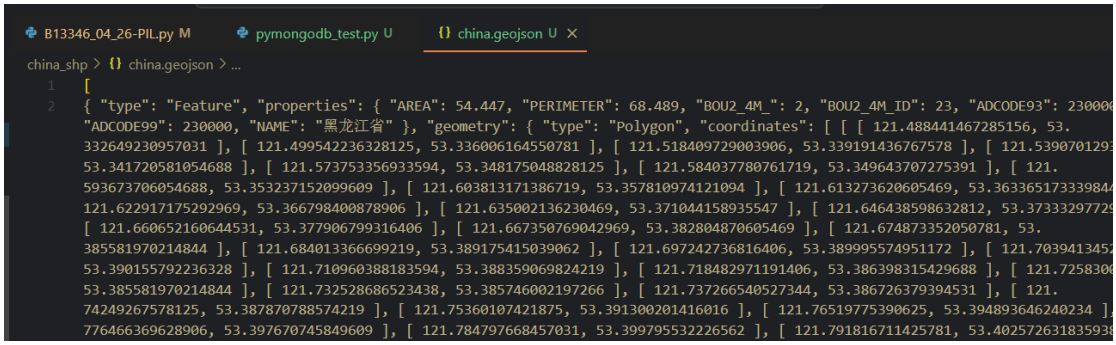
```
C:/Anaconda/python.exe c:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Pr
actice/prac3_506/geohash.py
8-bit GeoHash: wwdm7f3t
9-bit GeoHash: wwdm7f3tx
10-bit GeoHash: wwdm7f3tx4
PS C:\Users\86187\Desktop\Term\sophomore_2\GeoSpatialDataAnalysis\Practice\prac3_506>
```

2. 首先将 china.shp 文件导入到 MongoDB 中

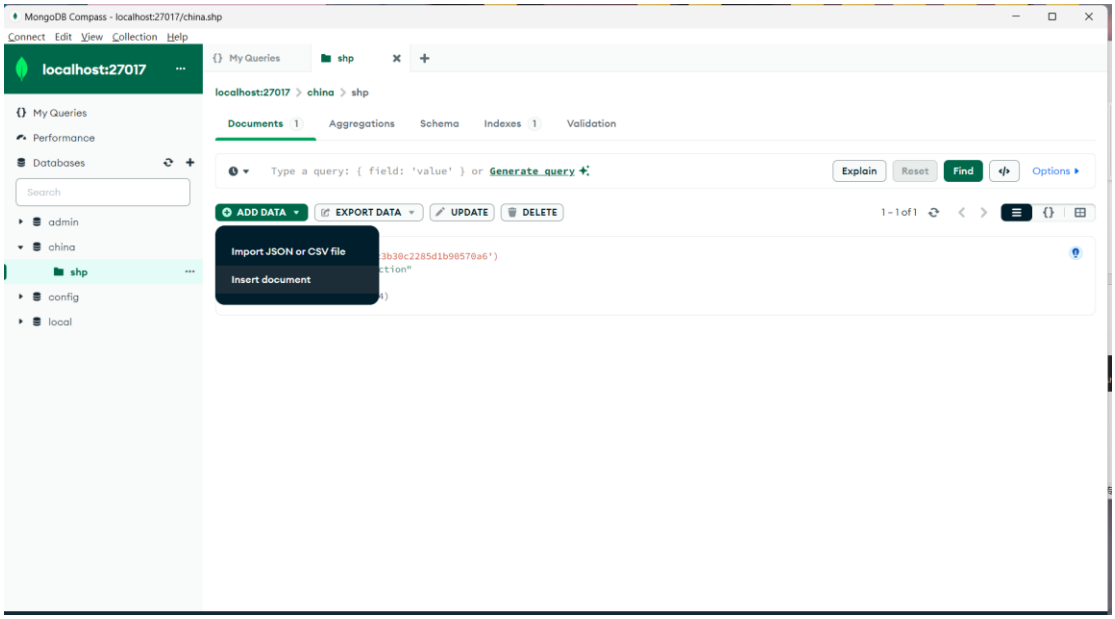
因为 shapefile 文件不能直接导入到 MongoDB 中，所以要先将.shp 文件转为 GeoJSON 文件，使用 ogr2ogr 命令行转换

```
ogr2ogr -f PostgreSQL "PG:dbname=m1tsuha user=postgres password=123456" -lco
PG_USE_COPY=YES -lco SHAPE_ENCODING=GBK -progress -update -append  -gt -1 -nln china
C:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Data/china_shp/china.shp
```

转换成 geojson 文件后 要把文件头删掉 使整体文件是 json 的格式 否则只会导入一个文件数据



接下来可以通过 GUI 可视化界面导入或者命令行导入

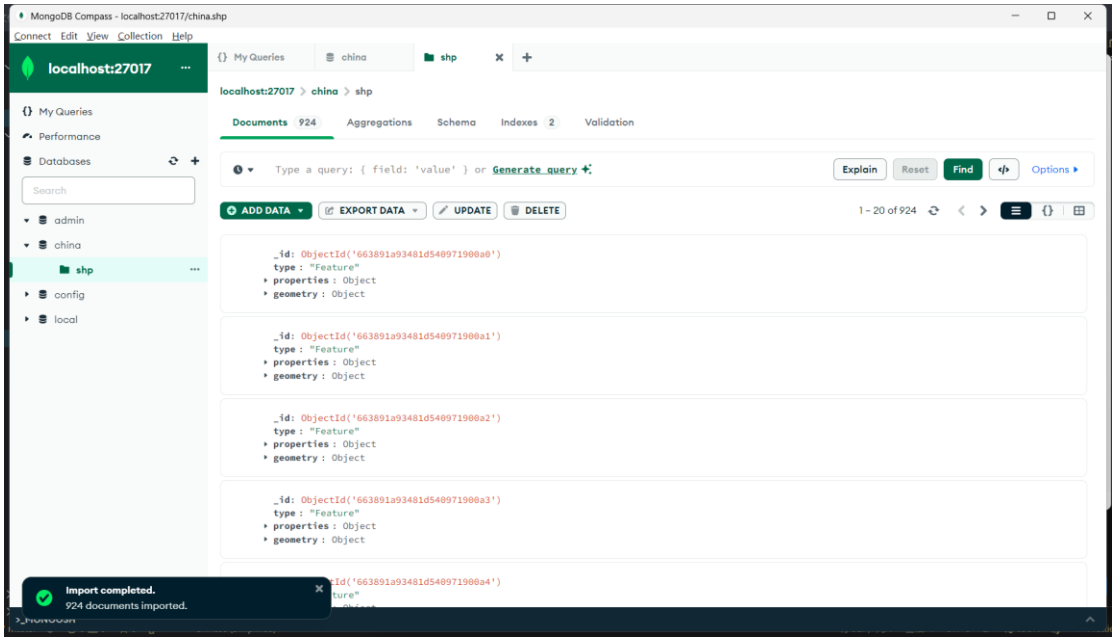


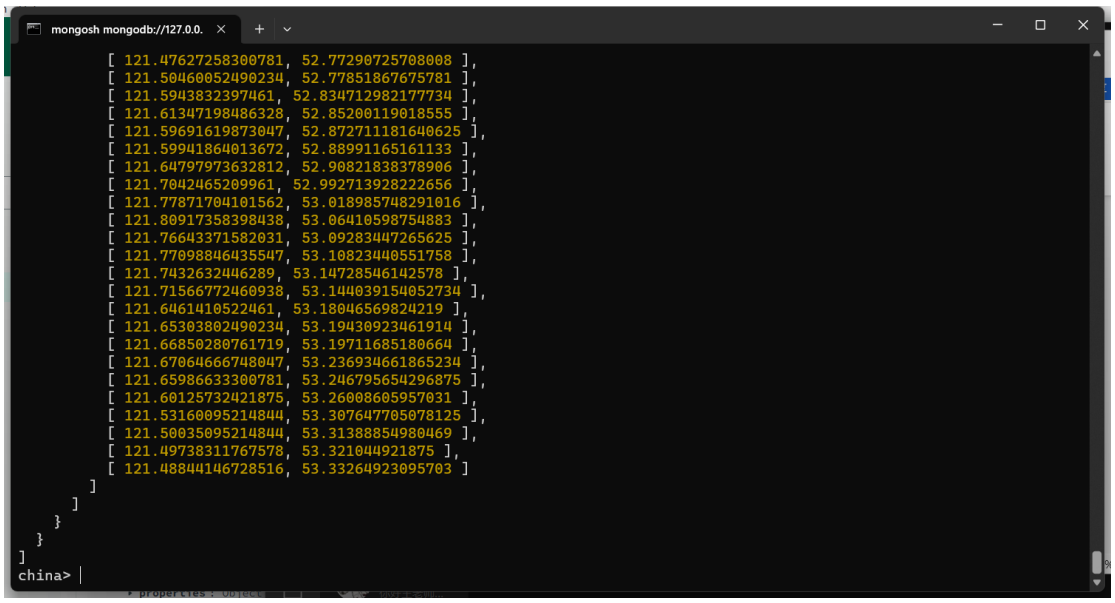
或者

导入 GeoJSON 到 MongoDB

mongoimport --db yourdatabase --collection yourcollection --file output.geojson -jsonArray

导入后的结果





然后用 python 开发，访问 MongoDB，进行 intersect 查询

代码如下：

```
'''
Author: M1tsuha
Date: 2024-05-06 15:26:26
LastEditors: M1tsuha
LastEditTime: 2024-05-06 16:21:06
FilePath: \Learning-Geospatial-Analysis-with-Python-Third-
Edition\pymongodb_test.py
Description:
```

```
Copyright (c) 2024 by ${git_name_email}, All Rights Reserved.
'''
from pymongo import MongoClient

# 连接到 MongoDB
client = MongoClient('localhost', 27017)
db = client['china']  # 使用的数据库名称
collection = db['shp']  # 使用的集合名称

# 定义直线的两点
line = {
    "type": "LineString",
    "coordinates": [
        [84.06, 26.18],  # 起点坐标
        [109.56, 46.02]  # 终点坐标
    ]
}

# 执行查询，查找与直线相交的地理对象
query_result = collection.find({
    "geometry": {
        "$geoIntersects": {
            "$geometry": line
        }
    }
})

# 打印结果，排除地理坐标信息
for doc in query_result:
    print({key: value for key, value in doc.items() if key !=
'geometry'})
```

处理结果如下

```
onda/python.exe c:/Users/86187/Desktop/Term/sophomore_2/GeoSpatialDataAnalysis/Code/Learning-Geospatial-Analysis-with-Python-Third-Editi
on/pymongodb_test.py
{'_id': ObjectId('663891a93481d540971900a1'), 'type': 'Feature', 'properties': {'AREA': 129.113, 'PERIMETER': 129.933, 'BOU2_4M_': 3, 'B
OU2_4M_ID': 15, 'ADCODE93': 150000, 'ADCODE99': 150000, 'NAME': '内蒙古自治区'}}
{'_id': ObjectId('663891a93481d54097190151'), 'type': 'Feature', 'properties': {'AREA': 114.331, 'PERIMETER': 76.629, 'BOU2_4M_': 179, '
BOU2_4M_ID': 54, 'ADCODE93': 540000, 'ADCODE99': 540000, 'NAME': '西藏自治区'}}
{'_id': ObjectId('663891a93481d540971900e2'), 'type': 'Feature', 'properties': {'AREA': 71.363, 'PERIMETER': 59.562, 'BOU2_4M_': 68, 'BO
U2_4M_ID': 63, 'ADCODE93': 630000, 'ADCODE99': 630000, 'NAME': '青海省'}}
{'_id': ObjectId('663891a93481d540971900a5'), 'type': 'Feature', 'properties': {'AREA': 41.508, 'PERIMETER': 76.781, 'BOU2_4M_': 7, 'BOU
2_4M_ID': 62, 'ADCODE93': 620000, 'ADCODE99': 620000, 'NAME': '甘肃省'}}
PS C:\Users\86187\Desktop\Term\sophomore_2\GeoSpatialDataAnalysis\Code\Learning-Geospatial-Analysis-with-Python-Third-Edition>
```

3. 改造遥感数据分类计算程序

先看整段代码

```
from osgeo import gdal_array
import numpy as np
import random

# Load the image into numpy using gdal
src = "GF1.jpg"
srcArr = gdal_array.LoadFile(src)

# Function to classify and save image
def classify_and_save(num_classes, tgt):
    # Split the histogram into num_classes bins as our classes
    classes = np.histogram(srcArr, bins=num_classes)[1]

    # Generate random color look-up table (LUT)
    lut = [[random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255)] for _ in range(num_classes + 1)]

    # Starting value for classification
    start = 1

    # Set up the RGB color JPEG output image
    rgb = np.zeros((3, srcArr.shape[0], srcArr.shape[1]), np.float32)

    # Process all classes and assign colors
    for i in range(len(classes)):
        mask = np.logical_and(start <= srcArr, srcArr <= classes[i])
        for j in range(len(lut[i])):
            rgb[j] = np.choose(mask, (rgb[j], lut[i][j]))
        start = classes[i] + 1

    # Save the image
    output = gdal_array.SaveArray(rgb.astype(np.uint8), tgt,
format="JPEG")
    output = None

# Classify and save images with 5, 10, and 15 classes
classify_and_save(5, "classified_5_classes.jpg")
classify_and_save(10, "classified_10_classes.jpg")
classify_and_save(15, "classified_15_classes.jpg")
```

在源程序上更改的地方为：
随机产生 LUT 颜色查找表 注意长度必须为 len(classes)+1

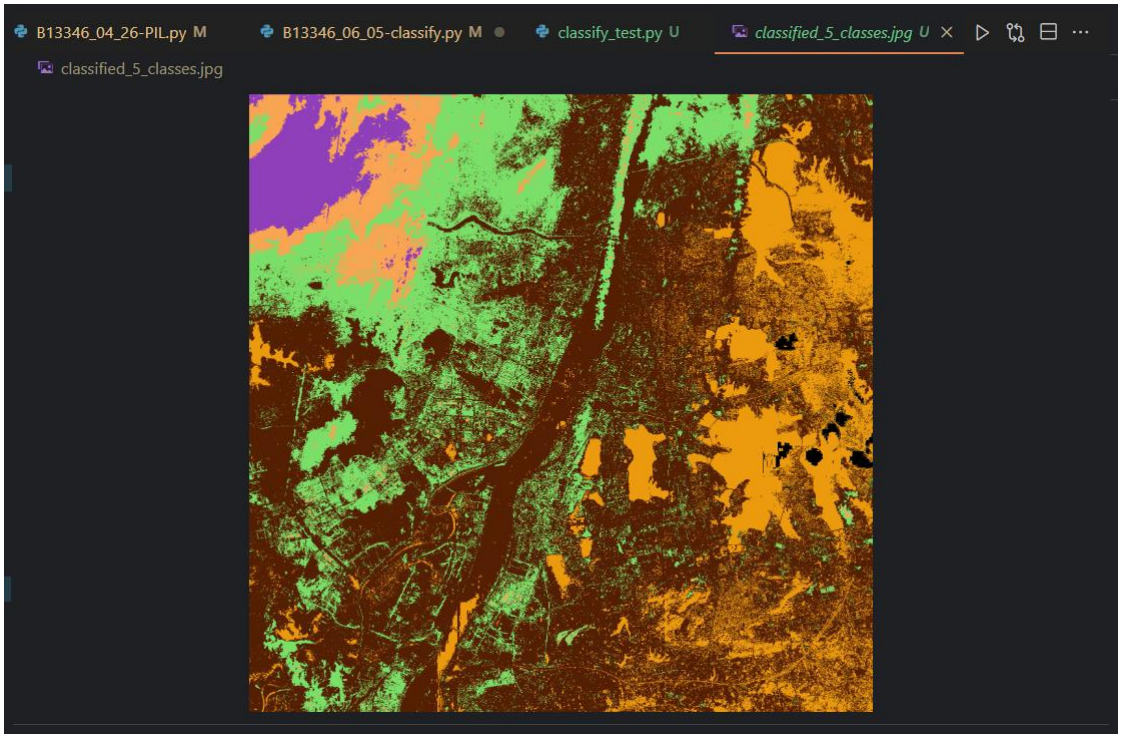
```
# Generate random color look-up table (LUT)
lut = [[random.randint(0, 255), random.randint(0, 255)
        , random.randint(0, 255)] for _ in range(num_classes + 1)]
```

将源程序处理关键步骤写成函数的形式，以便根据不同的类别个数进行分类

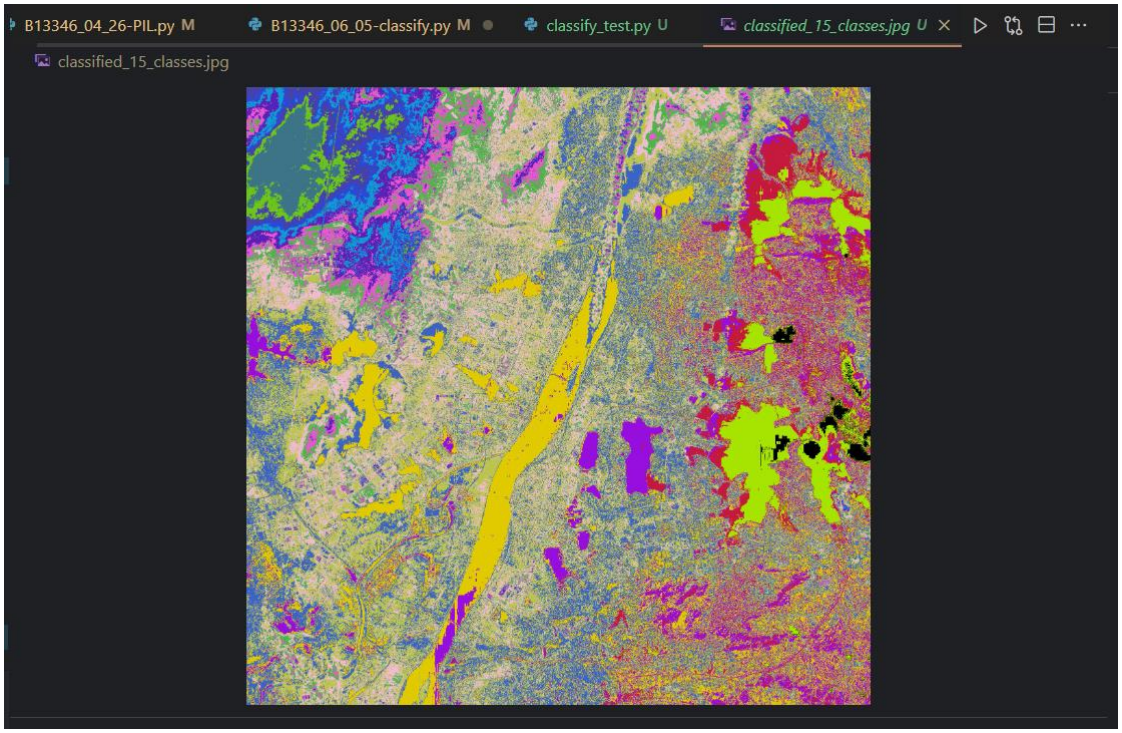
```
# Function to classify and save image
def classify_and_save(num_classes, tgt):
    # Split the histogram into num_classes bins as our classes
    classes = np.histogram(srcArr, bins=num_classes)[1]
```

其余函数通过遍历所有类别和颜色，根据像素值的范围将像素分配给相应的类别，并应用颜色查找表进行着色的关键步骤与源程序相同。

处理结果如下：




B13346_04_26-PIL.py M
B13346_06_05-classify.py M
classify_test.py U
classified_10_classes.jpg U
classified_10_classes.jpg


B13346_04_26-PIL.py M
B13346_06_05-classify.py M
classify_test.py U
classified_15_classes.jpg U
classified_15_classes.jpg