TÉRINFORMATIKA

3. forduló

ULYSSYS

A kategória támogatója: Ulyssys Kft.

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

A feladatlap több csatolmányt is tartalmaz, ezért a megoldását asztali gépen javasoljuk!

Megoldásokhoz ajánlott a következő **PostgreSQL** Docker image használata (OSM adatokkal fel van töltve): https://hub.docker.com/r/szokimoki/oitm-postgis-osm

```
docker pull szokimoki/oitm-postgis-osm
docker run -d -p 5432:5432 --name oitm szokimoki/oitm-postgis-osm
```

Python docker (rengeteg package-el): https://hub.docker.com/r/szokimoki/oitm-python

```
docker pull szokimoki/oitm-python
docker run -it szokimoki/oitm-python bash
```

Ajánlott asztali térinformatikai szoftver **QGIS**: https://qgis.org/hu/site/forusers/download.html

Alapértelmezett beállításokkal a fenti docker DB elérése:

```
Host: localhost
Port: 5432
Database: postgres
```

```
Username: postgres
Password: postgres
```

A letölthető dokumentum a Dockerhez való csatlakozáshoz ad segítséget!

Felhasznált idő: 00:00/40:00 **Elért pontszám:** 0/20

Indítás előtti csatolmányok

1. feladat 0/7 pont

Futóverseny

Az alábbi 2 kódrészlet hasonló eredményt ad vissza, azonban a 2 közül az egyik lassabban fut.

Melyik kód fut lassabban, és melyik kódrészlet teszi jelentősen lassúvá?

Kód #1

```
def process_raster(input_folder, clip_vector, raster_output):
    for sat_input in os.listdir(input_folder):
        print (sat_input)
        fname = sat_input.split('.')[0]
        print (fname + ' processing...')
        input_raster = gdal.Open(os.path.join(input_folder, sat_input))
        band3 = input_raster.GetRasterBand(3).ReadAsArray()
        band7 = input_raster.GetRasterBand(7).ReadAsArray()
        ndvi = ((band7 - band3)/(band7 + band3))*100
        no_veg = (ndvi <= 20.0)*1 + (ndvi > 20.0)*0

        kwargs = {
                'format': 'GTiff',
                'outputType': gdal.GDT_Float32,
                'bandList': [1,2],
                'noData': 0
                }

        raster_out_path = os.path.join(raster_output, fname + '_no_veg.tiff')
        raster_clipped_path = (os.path.join(raster_output, fname + '_no_veg_clipped.tiff'))

        raster_out = gdal.Translate(raster_out_path, input_raster, **kwargs)

        raster_out.GetRasterBand(2).WriteArray(ndvi)
        raster_out.GetRasterBand(1).WriteArray(no_veg)

        gdal.Warp(raster_clipped_path, raster_out, cutlineDSName=clip_vector, cropToCutline=True, dstNodata=0
        os.remove(raster_out_path)
        print (raster_clipped_path + ' processed!')
```

Kód #2

```
def process_raster(input_folder, clip_vector, raster_output):
    for sat_input in os.listdir(input_folder):
        print (sat_input)
        fname = sat_input.split('.')[0]
        print (fname + ' processing...')
        input_raster = gdal.Open(os.path.join(input_folder, sat_input))
        band3 = np.array(input_raster.GetRasterBand(3).ReadAsArray())
        band7 = np.array(input_raster.GetRasterBand(7).ReadAsArray())
        ndvi = (band7 - band3)/(band7 + band3)
        no_veg = (ndvi <= 0.2)*1 + (ndvi > 0.2)*0

        kwargs = {
                'format': 'GTiff',
                'outputType': gdal.GDT_Byte,
                'noData': 0
                }

        raster_out_path = os.path.join(raster_output, fname + '_no_veg.tiff')
        raster_clipped_path = (os.path.join(raster_output, fname + '_no_veg_clipped.tiff'))

        raster_out = gdal.Translate(raster_out_path, input_raster, **kwargs)

        raster_out.GetRasterBand(2).WriteArray(ndvi)
        raster_out.GetRasterBand(1).WriteArray(no_veg)

        gdal.Warp(raster_clipped_path, raster_out, cutlineDSName=clip_vector, cropToCutline=True, dstNodata=0
        os.remove(raster_out_path)
        print (raster_clipped_path + ' processed!')
```

Válasz

- ○ Az 1., mert az NDVI fölöslegesen van felszorozva 100-zal
- ○ A 2., mert a sávok specifikusan numpy array-ként vannak beolvasva
- ○ Az 1., mert a kimeneti adatformátum Float32
- ● A 2., mert nincs a kimeneti képsávok száma specifikálva
  Ez a válasz helyes, de nem jelölted meg.
- ○ Az első és a harmadik válaszlehetőség együtt.

Magyarázat

Egy 30km x 30km-es Sentinel-2 teszterületre lefuttatva az 1. kód kb 2 másodperc, míg a 2. kód 4 másodperc alatt fut le.

Indoklás:

Az **első** és a **harmadik** lehetőség a "csak" a memóriában a változók tárolási kapacitását növeli, így összességében kevésbé lassítják a kód futását.

A **második** válasz irreleváns a futási idővel kapcsolatban, mivel a gdal eleve numpy arrayként olvassa be a képsávokat.

Azért a **negyedik** lehetőség *(tehát "A 2., mert nincs a kimeneti képsávok száma specifikálva")* a megoldás, mert ha nem specifikáljuk a kiírandó sávokat (így azok számát), akkor az eredeti képben lévő összes sáv kiírásra kerül, amely azt eredményezi, hogy a kimeneti fájlunk viszonylag nagy lesz. Míg a képeken futó számítások viszonylag egyszerűek, így nem igényelnek nagy számítási kapacitást, addig a memóriában lévő adat lemezre írása időigényes feladat.

2. feladat 0/7 pont

Tű a szénakazalban

Adott az alábbi Sentinel-2 kivágat. Keresd meg a "TÛ" -t a szénakazalban! (Titkos tipp: Segítenek az ASCII kódok egymás után írva)

Hol található az egysávos raszterben? (válaszként vesszővel elválasztva a raszterben található pixel helyét kérjük pl.: 333,455)

Válaszok

A helyes válasz:

1222,1526

677775,5647355

Magyarázat

Kedves Versenyzők!

A kérdést 0 pontosra állítottuk, mivel az első feladatban a mellékelt SHP egy korábbi verziót tartalmaz a poi-kból, mint – adott esetben a napokban pull-olt docker image-ből készült konténerben található – adatbázis. Így a második feladatra egységes megoldás nem volt adható.

Elnézést kérünk a kellemetlenségért!

(2022.11.09.)

Python kód, mely beolvassa és kiadja a raszterben elfoglalt helyét:

Az ASCII kódok: T = 84, Û = 219, tehát a raszterben keresendő érték: 84219

```
import numpy as np
from osgeo import gdal, osr


filename = "/opt/satellite/szenakazal.tif"
ds = gdal.Open(filename)
band = ds.GetRasterBand(1)
myarray = band.ReadAsArray()
result = np.where(myarray == 84219)
print(result)
```

3. feladat 0/6 pont

Semmit szédítő magasság

Készíts szintvonalakat a pontok alapján, majd állapítsd meg, hogy a legmagasabb pont körüli szintvonal formája melyik alakhoz hasonlít leginkább!

A szintvonalakat lineáris interpolációval készítsd el 15 m-s szintvonal-távolságokkal!

Válaszok

- [ ] Négyszög
- [ ] Háromszög

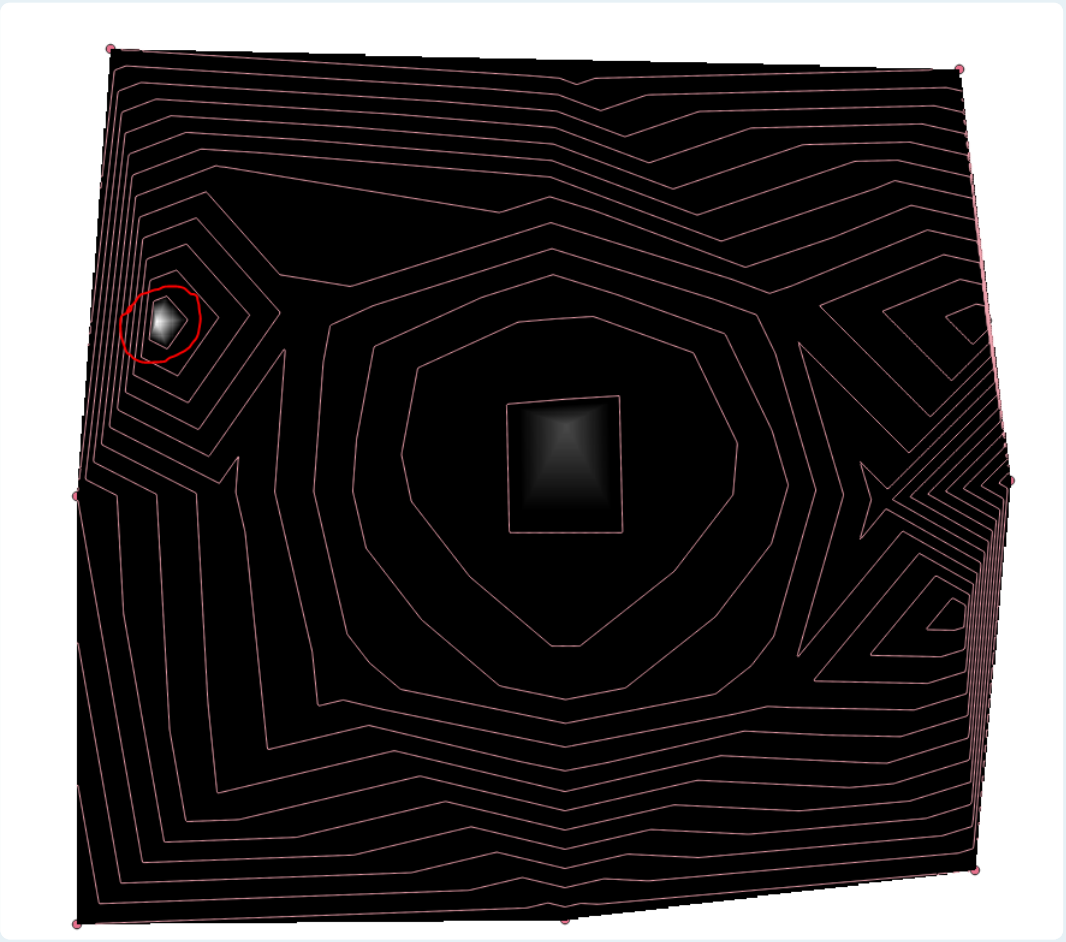
Kör
Ez a válasz helyes, de nem jelölted meg.

Egyik sem ezek közül
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Elsőként készítsünk egy TIN rasztert a TIN Interpolation Tool-al QGIS-ben a pontok alapján a megadott paraméterekkel, majd ez alapján hozzuk létre a szintvonalakat a Contour eszközzel.

A 2 legmagasabb szintvonal 255 m-en helyezkedik el. A legmagasabb ponthoz tartozót a TIN vagy a pontokat tartalmazó bemeneti állomány alapján tudjuk megállapítani.



Így látható, hogy a legmagasabb ponthoz tartozó szintvonal egy sokszög, azaz a **negyedik** a jó válasz.

Frissítés: a feladat IDW interpolációval is megoldható. Ez esetben az adathalmazból generált szintvonalak a legmagasabb pont körül körként rajzolódnak ki. Bár az eljárás gyakorlati alkalmazása kevésbé elterjedt, a megközelítés helyes.