Week 2: Identify Nearest Health Facilities

UPDATE

Thank you for your analysis. Despite our warning efforts so far, the virus continues to spread rapidly. We want to get infected individuals treatment as quickly as possible, so we need your help to calculate which hospital or clinic is closest to each known infected individual in the population.

Your goal for this notebook will be to identify the nearest hospital or clinic for each infected person.

Imports

In [1]:
```
import cudf
import cuml
import cupy as cp
```

Load Population Data

Begin by loading the `lat`, `long` and `infected` columns from `'./data/week2.csv'` into a cuDF data frame called `gdf`.

In [2]:
```
gdf = cudf.read_csv('./data/week2.csv', usecols = ['lat','long', 'infected'])
```

Load Hospital and Clinics Data

For this step, your goal is to create an `all_med` cuDF data frame that contains the latitudes and longitudes of all the hospitals (data found at `'./data/hospitals.csv'`) and clinics (data found at `'./data/clinics.csv'`).

In [3]:
```
all_med = cudf.read_csv('./data/hospitals.csv')
all_med.append(cudf.read_csv('./data/hospitals.csv'))
```

Out[3]:

	OrganisationID	OrganisationCode	OrganisationType	SubType	Sector	OrganisationStat
0	17970	NDA07	Hospital	Hospital	Independent Sector	Visib
1	17981	NDA18	Hospital	Hospital	Independent Sector	Visib
2	18102	NLT02	Hospital	Hospital	NHS Sector	Visib
3	18138	NMP01	Hospital	Hospital	Independent Sector	Visib
4	18142	NMV01	Hospital	Hospital	Independent Sector	Visib
...	...	...	...	...	...	
1224	10617466	RTH18	Hospital	UNKNOWN	Independent Sector	Visib
1225	10617482	RX3KI	Hospital	UNKNOWN	Independent Sector	Visib
1226	10617567	RGT1W	Hospital	UNKNOWN	Independent Sector	Visib
1227	10617714	RAX0A	Hospital	UNKNOWN	Independent Sector	Visib
1228	10617726	RDU5A	Hospital	UNKNOWN	Independent Sector	Visib

2458 rows × 22 columns

Since we will be using the coordinates of those facilities, keep only those rows that are non-null in both `Latitude` and `Longitude`.

In [4]:
```
all_med.dropna(subset=['Latitude','Longitude'], inplace= True)
```

Make Grid Coordinates for Medical Facilities

Provided for you in the next cell (which you can expand by clicking on the "...", and contract again after executing by clicking on the blue left border of the cell) is the lat/long to grid coordinates converter you have used earlier in the workshop. Use this converter to create grid coordinate values stored in `northing` and `easting` columns of the `all_med` data frame you created in the last step.

In [5]:
```
# https://www.ordnancesurvey.co.uk/docs/support/guide-coordinate-systems-great-britain

def latlong2osgbgrid_cupy(lat, long, input_degrees=True):
    '''
    Converts latitude and longitude (ellipsoidal) coordinates into northing and eastin

    Inputs:
    lat: latitude coordinate (N)
    long: longitude coordinate (E)
    input_degrees: if True (default), interprets the coordinates as degrees; otherwise

    Output:
    (northing, easting)
    '''

    if input_degrees:
        lat = lat * cp.pi/180
        long = long * cp.pi/180

    a = 6377563.396
    b = 6356256.909
    e2 = (a**2 - b**2) / a**2

    N0 = -100000 # northing of true origin
    E0 = 400000 # easting of true origin
    F0 = .9996012717 # scale factor on central meridian
    phi0 = 49 * cp.pi / 180 # latitude of true origin
    lambda0 = -2 * cp.pi / 180 # longitude of true origin and central meridian

    sinlat = cp.sin(lat)
    coslat = cp.cos(lat)
    tanlat = cp.tan(lat)

    latdiff = lat-phi0
    longdiff = long-lambda0

    n = (a-b) / (a+b)
    nu = a * F0 * (1 - e2 * sinlat ** 2) ** -.5
    rho = a * F0 * (1 - e2) * (1 - e2 * sinlat ** 2) ** -1.5
    eta2 = nu / rho - 1
    M = b * F0 * ((1 + n + 5/4 * (n**2 + n**3)) * latdiff -
```

```
                (3*(n+n**2) + 21/8 * n**3) * cp.sin(latdiff) * cp.cos(lat+phi0) +
                15/8 * (n**2 + n**3) * cp.sin(2*(latdiff)) * cp.cos(2*(lat+phi0)) -
                35/24 * n**3 * cp.sin(3*(latdiff)) * cp.cos(3*(lat+phi0)))
    I = M + N0
    II = nu/2 * sinlat * coslat
    III = nu/24 * sinlat * coslat ** 3 * (5 - tanlat ** 2 + 9 * eta2)
    IIIA = nu/720 * sinlat * coslat ** 5 * (61-58 * tanlat**2 + tanlat**4)
    IV = nu * coslat
    V = nu / 6 * coslat**3 * (nu/rho - cp.tan(lat)**2)
    VI = nu / 120 * coslat ** 5 * (5 - 18 * tanlat**2 + tanlat**4 + 14 * eta2 - 58 * t

    northing = I + II * longdiff**2 + III * longdiff**4 + IIIA * longdiff**6
    easting = E0 + IV * longdiff + V * longdiff**3 + VI * longdiff**5

    return(northing, easting)
```

In [6]:
```
med_lat = cp.asarray(all_med['Latitude'])
med_long = cp.asarray(all_med['Longitude'])
northing, easting = latlong2osgbgrid_cupy(med_lat, med_long)
all_med['northing'], all_med['easting'] = northing, easting
```

Find Closest Hospital or Clinic for Infected

Fit `cuml.NearestNeighbors` with `all_med` 's `northing` and `easting` values, using the named argument `n_neighbors` set to `1` , and save the model as `knn` .

In [7]:
```
knn = cuml.NearestNeighbors(n_neighbors = 1)
knn.fit(all_med[['northing','easting']])
```

Out[7]:
```
NearestNeighbors()
```

Save every infected member in `gdf` into a new dataframe called `infected_gdf` .

In [8]:
```
infected_gdf = gdf.loc[gdf['infected'] == 1]
```

Create `northing` and `easting` values for `infected_gdf` .

In [9]:
```
infected_gdf_lat = cp.asarray(infected_gdf['lat'])
infected_gdf_long = cp.asarray(infected_gdf['long'])
northing, easting= latlong2osgbgrid_cupy(infected_gdf_lat,infected_gdf_long)
infected_gdf['northing'], infected_gdf['easting'] = northing, easting
```

Use `knn.kneighbors` with `n_neighbors=1` on `infected_gdf` 's `northing` and `easting` values. Save the return values in `distances` and `indices` .

In [12]:
```
X = infected_gdf[["northing", "easting"]]
X.head()
```

Out[12]:

	northing	easting
1346586	424489.783814	371619.678741
1350932	418820.687944	371876.492369
1352085	422394.398940	367721.000265
1352799	422416.821887	367723.973098
1357529	425808.109929	374076.557677

In [14]:
```
distances, indices = knn.kneighbors(X)
```

Check Your Solution

`indices`, returned from your use of `knn.kneighbors` immediately above, should map person indices to their closest clinic/hospital indices:

In [15]:
```
indices.head()
```

Out[15]:
```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

Here you can print an infected individual's coordinates from `infected_gdf`:

In [16]:
```
infected_gdf.iloc[0] # get the coords of an infected individual (in this case, indivic
```

Out[16]:
```
lat             53.715826
long            -2.430079
infected         1.000000
northing    424489.783814
easting     371619.678741
Name: 1346586, dtype: float64
```

You should be able to used the mapped index for the nearest facility to see that indeed the nearest facility is at a nearby coordinate:

In [17]:
```
all_med.iloc[1234] # printing the entry for facility 1234 (replace with the index ider
```

Out[17]:

OrganisationID	OrganisationCode	OrganisationType	SubType	Sector	OrganisationStatus	IsPimsMa

0 rows × 24 columns

Please Restart the Kernel

...before moving to the next notebook.

In [ ]:
```
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```