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 [ ]:
```
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 [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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 [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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

In [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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 [ ]:

```
# https://www.ordnancesurvey.co.uk/docs/support/guide-coordinate-systems-great-britain.pdf

def latlong2osgbgrid_cupy(lat, long, input_degrees=True):
    '''
    Converts latitude and longitude (ellipsoidal) coordinates into northing and easting (grid) coordinates, using a Transverse Mercator projection.

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

    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 + tanla
```

```
t**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 + 1
4 * eta2 - 58 * tanlat**2 * eta2)

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

    return(northing, easting)
```

In [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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 [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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

In [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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

In [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

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

In [ ]:
```
ANSWER OR CODE COPYRIGHTED BY NVIDIA DEEP LEARNING INSTITUTE
```

Check Your Solution

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

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

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

In [ ]:
```
infected_gdf.iloc[0] # get the coords of an infected individual (in t
his case, individual 0)
```

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 [ ]:
```
all_med.iloc[1234] # printing the entry for facility 1234 (replace wi
th the index identified as closest to the individual)
```

Please Restart the Kernel

...before moving to the next notebook.

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