```python
## Lab 3: Part 1

# !pip install requests
import os
import requests
import zipfile
import json
import arcpy
from arcpy import env
from arcpy.sa import *

arcpy.env.overwriteOutput = True

# Step 1: Set working directory
directory = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data"
os.chdir(directory)
print(os.getcwd())
```

```
C:\Users\benla\Desktop\Grad_School\Classes\GIS5571_SpatialDataScience\
Labs\Lab2\Data
```

```python
# Step 2: Download landcover data
landcover_download = 
requests.get("https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn
_state_dnr/biota_landcover_nlcd_mn_2019/
tif_biota_landcover_nlcd_mn_2019.zip")
with open("./landcover.zip", 'wb') as file1:
    file1.write(landcover_download.content)

with zipfile.ZipFile("./landcover.zip", 'r') as landcover_zip:
        landcover_zip.extractall('landcover')

# Step 3: Download elevation data
elevation_download = 
requests.get("https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn
_state_dnr/elev_30m_digital_elevation_model/
fgdb_elev_30m_digital_elevation_model.zip")
with open("./elevation.zip", 'wb') as file2:
    file2.write(elevation_download.content)

with zipfile.ZipFile("./elevation.zip", 'r') as file2:
    file2.extractall('elevation')

# Step 4: Set up two points for start and end points. Buffer and clip
datasets to create field area.
start_and_end_points = [
    [-92.148796, 44.127985],
    [-92.044783, 44.054387],
]
```

```
arcpy.CreateFeatureclass_management(
    out_path = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Lab2_aprx\MyProject.gdb",
    out_name = "start_and_end_points",
    geometry_type = 'Point',
    spatial_reference = arcpy.SpatialReference(4326)
)

points_path = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Lab2_aprx\MyProject.gdb\
start_and_end_points"

with arcpy.da.InsertCursor(points_path, ['SHAPE@']) as cursor:
    for longitude, latitude in start_and_end_points:
        point = arcpy.Point(longitude, latitude)
        point_geometry = arcpy.PointGeometry(point,
arcpy.SpatialReference(4326))
        cursor.insertRow([point_geometry])

points_buffer = []
for i, point in enumerate(start_and_end_points):
    buffer_format = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Lab2_aprx\MyProject.gdb\
BufferFeature".format(i + 1)
    points_buffer.append(buffer_format)
    arcpy.Buffer_analysis(points_path, buffer_format, '10000 Meters')

# Step 5: Calculate slope
dem = arcpy.sa.Slope(r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\elevation\
elev_30m_digital_elevation_model.gdb\digital_elevation_model_30m",
"DEGREE", 1, "PLANAR", "METER")
dem.save(r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\slope")

# Step 6: Clip layer, reclassify data
slope_raster = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\slope"
clipped_slope_raster = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Lab2_aprx\MyProject.gdb\
clipped_slope_raster"

arcpy.management.Clip(slope_raster, '#', clipped_slope_raster,
buffer_format,'#', "ClippingGeometry", "NO_MAINTAIN_EXTENT")
slope_reclass = arcpy.sa.Reclassify("clipped_slope_raster", "Value",
"0 10 1;10 20 2;20 30 3;30 40 4;40 50 5;50 60 6;60 70 7;70 80 8;80 90
9;90 100 10")

# Step 7: Reclassifying data again
'11 10' # Open water
```

```
'21 0' # Developed, open space
'22 0' # Minimally developed
'23 0' # Moderately developed
'24 0' # Highy developed
'31 0' # Barren land
'41 1' # Decidious forest
'42 1' # Evergreen forest
'43 1' # Mixed Forest
'52 4' # Shrub/Scrub
'71 7' # Herbaceous
'81 7' # Hay/Pasture
'82 10'# Cultivated Crops
'90 2' # Woody Wetlands
'95 2' # Emergent Herbaceous Wetlands

landcover_path = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\landcover\
NLCD_2019_Land_Cover.tif"
landcover_output = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\clip_landcov"

arcpy.management.Clip(landcover_path, "#", landcover_output,
points_buffer[0], "#", "ClippingGeometry", "NO_MAINTAIN_EXTENT")

landcover_reclass = arcpy.sa.Reclassify("clip_landcov", "Value", "11
10; 21 0; 22 0; 23 0; 24 0; 31 0; 41 1; 42 1; 43 1; 52 4; 71 7; 81 7;
82 10; 90 2; 95 2", "DATA")

# Step 8: Generating 4 different cost surfaces
for loop_index, w in enumerate([0.1, 0.25, 0.5, 0.75]):
    landcover_weight = w
    slope_weight = 1-w
    cost_surface = arcpy.ia.RasterCalculator([landcover_reclass,
slope_reclass],
                                          ['landcover_reclass',
'slope_reclass'],

expression=f"({landcover_weight} * landcover_reclass) +
({slope_weight} * slope_reclass)")
    output_path = rf"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\
cost_surface_{loop_index}.tif"
    cost_surface.save(output_path)

# Add the cost surfaces to the map
aprx = arcpy.mp.ArcGISProject("CURRENT")
mp = aprx.listMaps("working_map_2")[0]

for loop_index in range(4):
    mp.addDataFromPath(output_path)
```

```
# Step 9: Optimal path for each cost surface

# Set workspace
export_directory = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports"
arcpy.env.workspace = export_directory

# Paths to the cost surface rasters
cost_surfaces = [
    os.path.join(export_directory, f"cost_surface_{i}.tif") for i in
range(4)
]

# Path to the starting and ending points feature class
points_path = r"C:\Users\benla\Desktop\Grad_School\Classes\
GIS5571_SpatialDataScience\Labs\Lab2\Lab2_aprx\MyProject.gdb\
start_and_end_points"

# Create feature layers for the start and end points using OBJECTID
start_point = arcpy.management.MakeFeatureLayer(points_path,
"start_point", "OBJECTID = 1")
end_point = arcpy.management.MakeFeatureLayer(points_path,
"end_point", "OBJECTID = 2")

# Generate optimal paths for each cost surface
for i, cost_surface in enumerate(cost_surfaces):
    # Set the output path for the cost distance, backlink, and optimal
path rasters
    cost_distance_raster = os.path.join(export_directory,
f"cost_distance_{i}.tif")
    backlink_raster = os.path.join(export_directory,
f"backlink_{i}.tif")
    optimal_path_raster = os.path.join(export_directory,
f"optimal_path_{i}.tif")

    # Use the CostDistance tool to calculate the cost distance raster
    cost_distance_result = arcpy.sa.CostDistance(start_point,
cost_surface)
    cost_distance_result.save(cost_distance_raster)

    # Use the CostBackLink tool to generate the backlink raster
    backlink_result = arcpy.sa.CostBackLink(start_point, cost_surface)
    backlink_result.save(backlink_raster)

    # Use the CostPath tool to calculate the optimal path from start
to end
    least_cost_path = arcpy.sa.CostPath(end_point,
cost_distance_raster, backlink_raster, "EACH_CELL")
    least_cost_path.save(optimal_path_raster)
```

```
    print(f"Optimal path for cost surface {i} saved to
{optimal_path_raster}")

# Add output to the map
aprx = arcpy.mp.ArcGISProject("CURRENT")
mp = aprx.listMaps("working_map_2")[0]
for i in range(4):
    path = os.path.join(export_directory, f"optimal_path_{i}.tif")
    mp.addDataFromPath(path)

print("Optimal paths created for each cost surface.")
```

```
Optimal path for cost surface 0 saved to C:\Users\benla\Desktop\
Grad_School\Classes\GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\
optimal_path_0.tif
Optimal path for cost surface 1 saved to C:\Users\benla\Desktop\
Grad_School\Classes\GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\
optimal_path_1.tif
Optimal path for cost surface 2 saved to C:\Users\benla\Desktop\
Grad_School\Classes\GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\
optimal_path_2.tif
Optimal path for cost surface 3 saved to C:\Users\benla\Desktop\
Grad_School\Classes\GIS5571_SpatialDataScience\Labs\Lab2\Data\Exports\
optimal_path_3.tif
Optimal paths created for each cost surface.
```