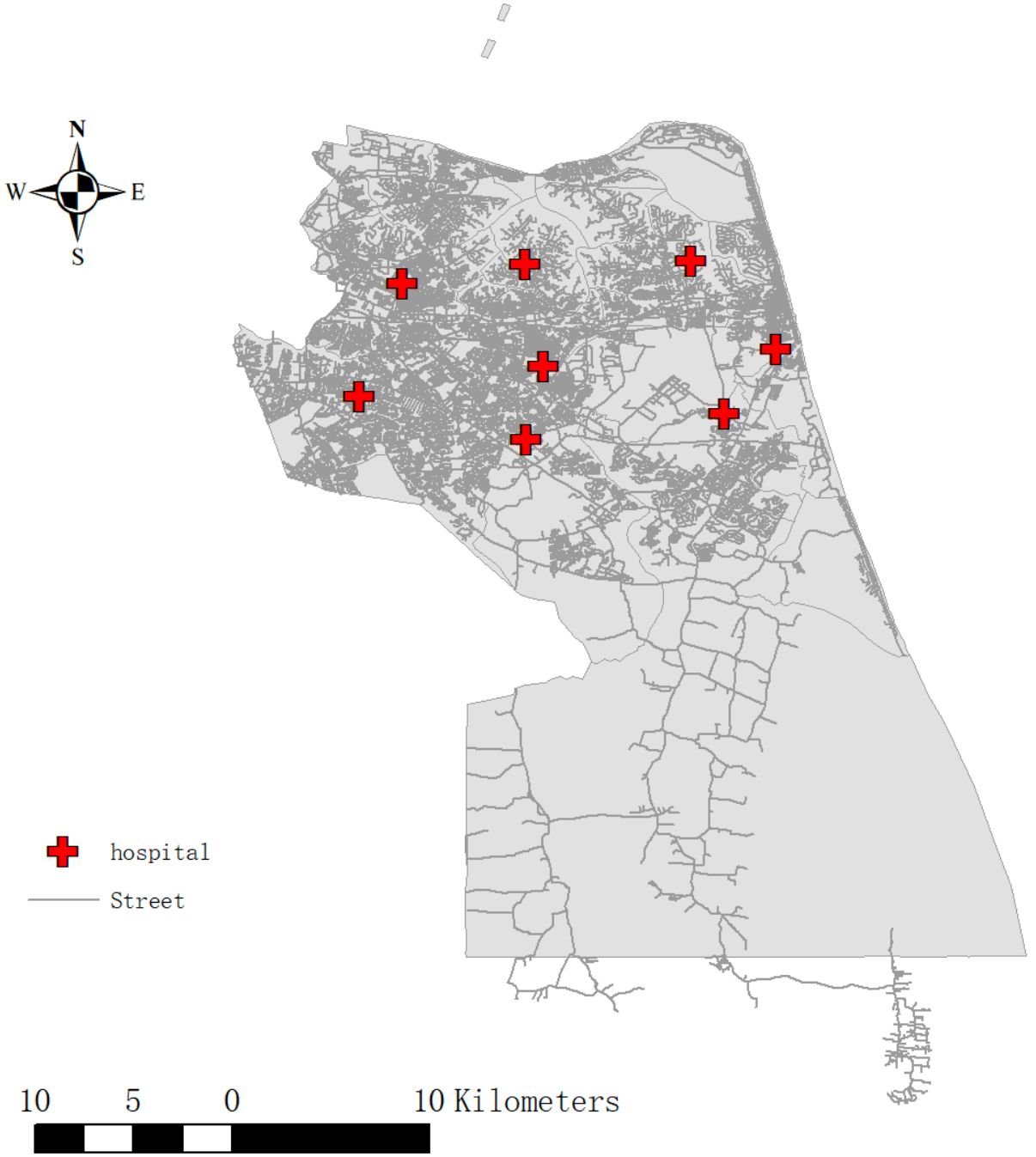
Virginia beach emergency call ---optimal hospital and route

This final project, I developed is for selecting optimal hospital and make the best route for the ambulance drivers.

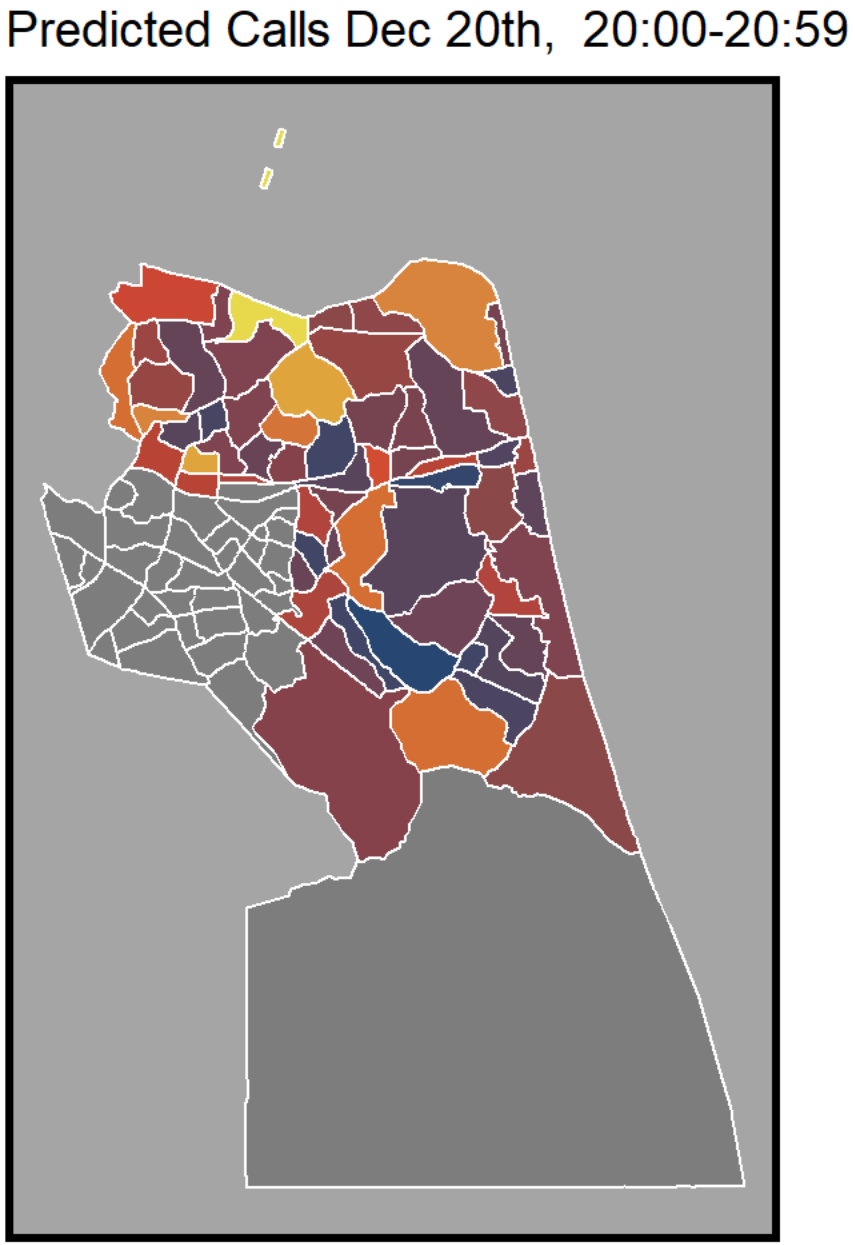
Fangnan Du

Final Project



Background

- Nowadays, because of the increasing pressure for living, people’ s health issue becomes one of the main concerns for a city’ s health construction and future health infrastructure's development.
- In my previous project, I did the emergency call prediction in the Virginia beach using the Poisson regression. And the outcome is the when and where will the emergency call will happen. This prediction can give the ambulance drivers a great tool to reduce the response time.
- However, although the ambulance driver has know the emergency calls , but when they get there, it is a problem to choose the optimal hospitals and the best route to the hospitals.
- For the optimal hospital, the most common way to choose which hospital to go is based on the experience for the drivers. And for the most efficient route, the ambulance driver mostly will choose the closest and shortest way to go instead of considering other factors, such as the traffic flow.
- In this project, I will create a tool for the ambulance driver to choose the optimal hospital to go and the best route to go.



Introduction

- In this project, the tool is for selecting the hospital and choosing the best route.
- There are reasons for creating this tool for these two purpose:

  - The purpose for selecting the hospital

  1. When a emergency call comes, it will come through the emergency process center, and then make clear instruction to the ambulance driver. So it will definitely cost the time. What I developed, the tool can solve this problem. **Because when the emergency call comes, it will automatically sent the location for the ambulance driver**, without being noticed by the emergency process center. So in this way, the time will be saved. In this emergency call case, the time is the most valuable thing.
  2. When they got there, the ambulance driver can determine the **priority level** for the patient. In the normal case, the priority level is predetermined by the emergency process center. But they determine the priority level just based on the calling. And for the ambulance driver, they can on scene and then determine the priority just with a single click. In this way, the priority will be much more accurate and the resource of high priority level hospital will not be wasted.

Introduction

- In this project, the tool is for selecting the hospital and choosing the best route.
- There are reasons for creating this tool for these two purpose:

  ➢ The purpose for choosing the best route

  When the ambulance driver has arrived the emergency scene, the most important thing to be is to determine the most time-efficient way to go to the determined hospital. However, in the traditional way, the ambulance drive mostly will choose the shortest route subconsciously. But as we all know, if we count the traffic flow, the population density and the rain even the elevation, the time will be saved much more.

  So it is very important to create a traffic impedance for the best routing

  In this project, as a demo, I create a traffic impedance of two factors: traffic flow and the population density.

DATA

- The most important function of this tool is to find the optimal route, which is mainly based on the network analysis. But before we run the arcpy to use the network analysis, we have to set up the data set which is required for the network analysis.

  - In order to use the network analysis, the dataset is used as following:

  1. Network dataset
  2. Facilities feature classes
  3. Incidents feature classes

Network dataset

For the network dataset:

As for the network dataset, we should first create a network dataset of the street in ArcCatalog.

The further analysis will based on the dataset which is created above.

We can have the access of the street data from Virginia open data (https://data.vbgov.com) .

Facility features dataset

The facility data set should be pre processed by Arcgis.
In this step, instead of creating a tool, I create a model builder tool to visualize my process, in order to delivering a clear map of my processes.

And this dataset eventually should have the impedance field, which is calculated by traffic flow and population density.
This will be explained later.

Incidents dataset

In this scenario, because the main purpose of this tool is used from a app. So the incidents location will be extracted from the app location sharing.

And the incident we are using is just 1 incident, which is the emergency call. Because 1 ambulance only can go to respond 1 emergency call.

Facility features dataset

Goal: add weight field(impedance)

Traffic flow

Population density

The hospital dataset is found as well as from the Virginia beach open data source.
And the most important thing to do is add the weight field to the hospitals.

The traffic flow dataset is from the Virginia beach.
But the raw data is excel. It indicates the average traffic flow for one year.
So, in this case, I use the traffic flow to convert to the point and do the kernel density. So we will get the traffic flow raster for all virginia beach.
(https://www.vbgov.com/government/departments/public-works/traffic/Pages/traffic-count-data.aspx)

City of Virginia Beach
ADT's: 2013 - 2017

View In Excel

Count Station	Street Name	From	To	Latitude	Longitude	2013	Date	2014
371	19th	Birdneck Rd.	Parks Ave.	36.8440056853693	-75.9932899475098	4,540	Aug-13	3,981
372	19th	Parks Ave.	Baltic Ave.	36.8459375057883	-75.9824752807617	3,995	Aug-13	4,068
818	21st St.	Baltic Ave.	Arctic Ave.	36.8485131904247	-75.9794282913208	11,308	Aug-13	11,577
819	22nd St.	Arctic Ave.	Baltic Ave.	36.8495176839139	-75.9795892238617	9,022	Aug-13	11,060
351	22nd St.	Atlantic Ave.	Pacific Ave.	36.8500929007383	-75.976231098175	3,407	Aug-13	3,334
133	24th St.	Barberton Dr.	Parks Ave.	36.852376554684	-75.9929358959198	3,785	Aug-13	3,589
182	30th St.	Arctic Ave.	Pacific Ave.	36.8579888165699	-75.979999601841	9,079	Jul-13	
353	9th St.	Pacific Ave.	Atlantic Ave.	36.8375659314896	-75.9727442264557	3,548	Aug-13	3,504
400	Albright Dr.	Lynnhaven Pkwy.	Kempsville Rd.	36.7846707580299	-76.1766350269318	6,208	Dec-13	6,100
312	Aragona Blvd.	Virginia Beach Blvd.	Kellam Rd.	36.8494404156525	-76.1462187767029	3,417	Sep-13	3,485
314	Aragona Blvd.	Witchduck Rd.	Haygood Rd.	36.8620083592744	-76.1457145214081	7,026	Sep-13	7,181
349	Arctic Ave.	19th St.	21st St.	36.8478692774002	-75.978194475174	3,846	Aug-13	4,000
708	Atlantic Ave.	17th St.	Laskin Rd.	36.8460233633408	-75.9744501113892	8,551	Aug-13	7,321
710	Atlantic Ave.	37th St.	Bay Colony Drive	36.8709952317855	-75.9825074672699	21,405	Aug-13	18,444
451	Atlantic Ave.	83rd St.	Fort Story	36.9108526090908	-75.9942448139191	6,525	Aug-13	4,090
707	Atlantic Ave.	9th St.	17th St.	36.8408202215598	-75.9730231761932	7,315	Aug-13	6,972
837	Atlantic Ave.	Bay Colony Drive	83rd St.	36.8971086698319	-75.9903717041016	12,598	Jul-13	12,905

Facility features dataset

Goal: add weight field(impedance)

Traffic flow

Population density

The population density is found in the tract 2000 of virginia beach, and the process step is to use the polygon to raster and choose the popden field as value to get the raster data for entire virginia beach.

Weight=2*Traffic flow +population density

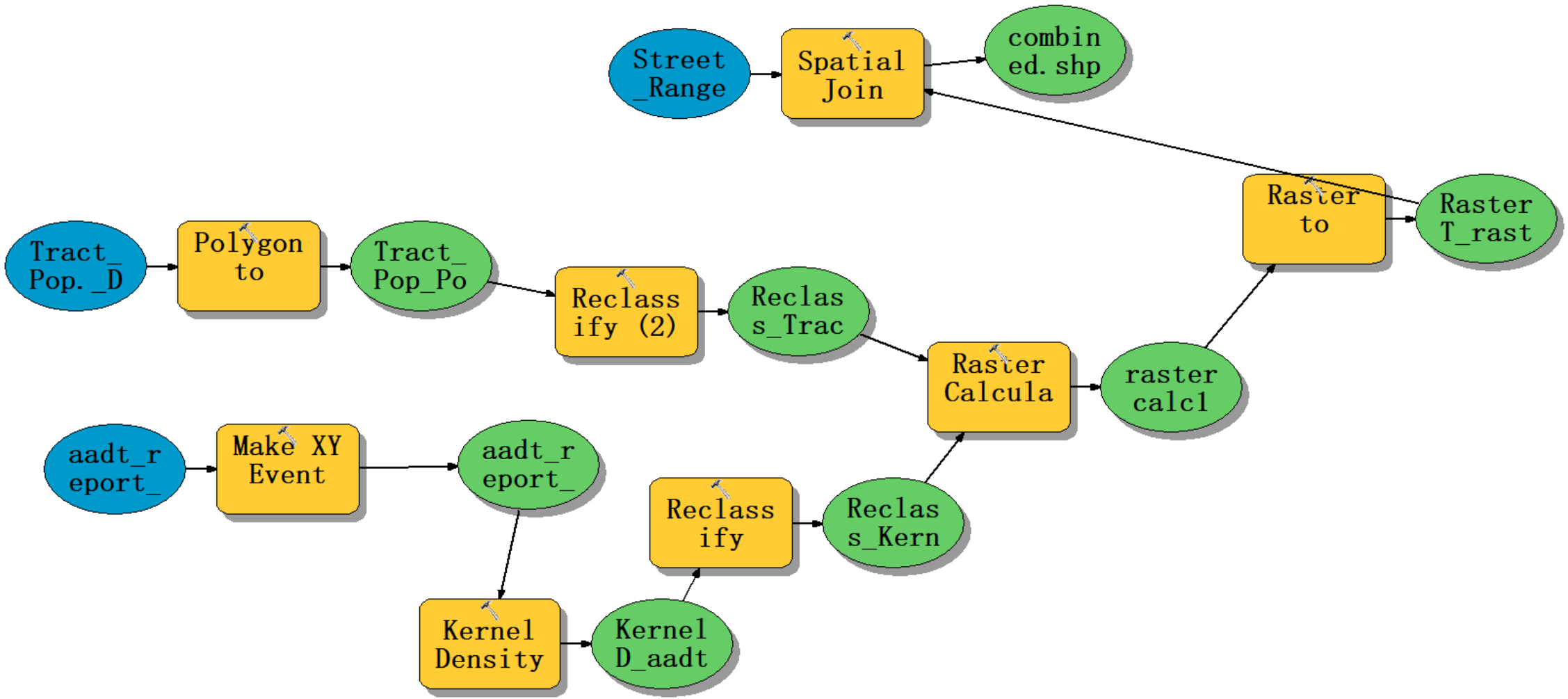
In this step, I used the my own equation to give the weight on the traffic flow and the population density. This weight is not for accuracy, just for demo. Because, if we want to do this for accuracy, we need to add more variables, such as rain and elevation.
And in this step, we used the raster calculator to do this process.

Attach the weight to the hospital, to measure the average hardness to get to this particular hospital (using the extra value)

In step, we are only for the demo. Because when we apply this to reality, we should use the real time traffic flow and calculate the cost of entire route.

Using model builder to do it

Arcpy code for doing it is in the appendix 2



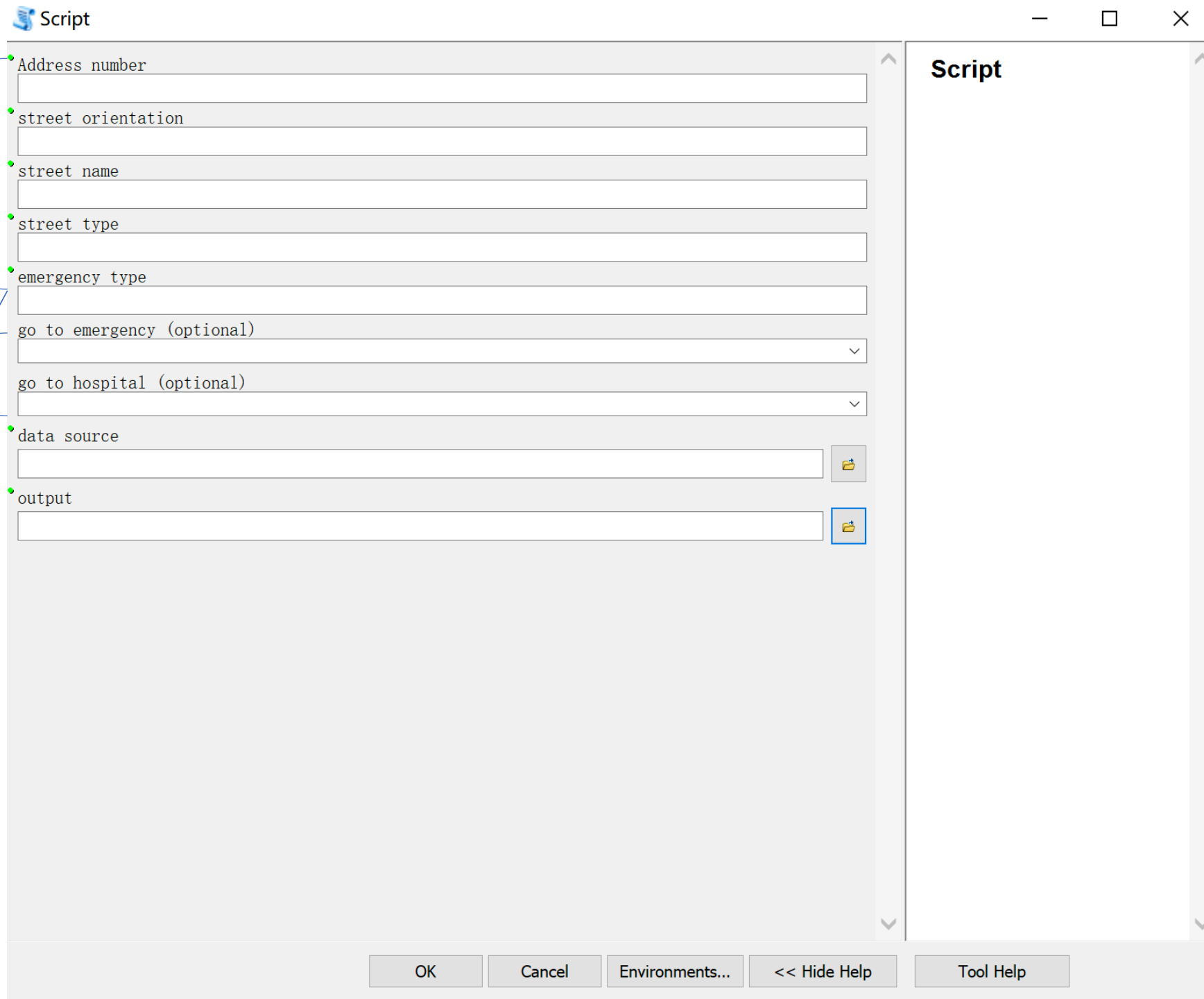
Parameters

This is the location parameters

In the real life, this parameters will obtained automatically.

The emergent type is set for the ambulance driver to give, which can give a more accurate result.

This is the direction of the ambulance goes: whether it goes to the emergency points or the hospital.



Process

1. Set the parameter

2. Create emergency location and select hospital based on priority level

```
#--------------------emergency location create and hospital selection----------
arcpy.MakeFeatureLayer_management(Emergency, "lyr")     # Make a layer from the feature class
# Incident address selection
Start = NO+" "+direct+" "+street+" "+type
Start = str(Start)
#select point by location name
arcpy.SelectLayerByAttribute_management("lyr", "NEW_SELECTION", '"Address" = 'Start'')
# copy the selecttion to a new point shapefile
arcpy.CopyFeatures_management("lyr", "Incidents")
Incidents = "Incidents"
arcpy.AddMessage('Incident Address:' + Start)

    # Hospital Selection
    # When the ambulance recieved the message, there will be a prioprity type, which indicates the hospital
TYPE
    # if the priority is 1 ,then choose the highest level hospital
    if EmergencyType == "1":
        arcpy.Select_analysis(Hospitals, "HosSelection", '"CallPriority" = '1'')
    elif EmergencyType == "2":
        arcpy.Select_analysis(Hospitals, "HosSelection", '"CallPriority" = '2'')
    elif EmergencyType == "2":
        arcpy.Select_analysis(Hospitals, "HosSelection", '"CallPriority" = '3'')
    else:
```

Process

3. Using the median weight to choose the optimal hospital

After we have get the weight from the previous step. In this secton. We will use the weight to find the optimal hospital.

The script uses the Near tool to calculate distance and then adds and calculates a new field for the weighted rank using this equation: $weight = rank\ /(distanc\ e\ )\text{^}(1/2)$

This method is learned from Henry Bernberg report in last assignments.

```
#----------add the weighted impedance to select the best routed hospital before network analyse
#calculate the distance from the hospital to emergency call
arcpy.Near_analysis(Hospitals, Incidents)
## Add a weight field for impedance( the impedance is calculated by another script which showed at the end of the report)
arcpy.AddField_management(Hospitals, "weight", "DOUBLE")
# Calculate the distance based weigh of each selected hospital based on their median ranking
arcpy.CalculateField_management(Hospitals, "weight", "( [median] / (Sqr ( [NEAR_DIST] ) ) ) * 100", "VB", "")
# Determine the maximum wieghted ranking amongst the hospitals
arcpy.Statistics_analysis(Hospitals, "Hos_max", "weight MAX", "")
# Join the maximum wiegh back to the hospitals locations
arcpy.JoinField_management(Hospitals, "weight", "Hos_max", "MAX_weight", "")
# Select the hospitals locaiton with the highest weighted rank or equally 2 or more highest weighted rank
Hospitals = arcpy.Select_analysis(Hospitals, "Hospitals", '"FREQUENCY" > 0')
```

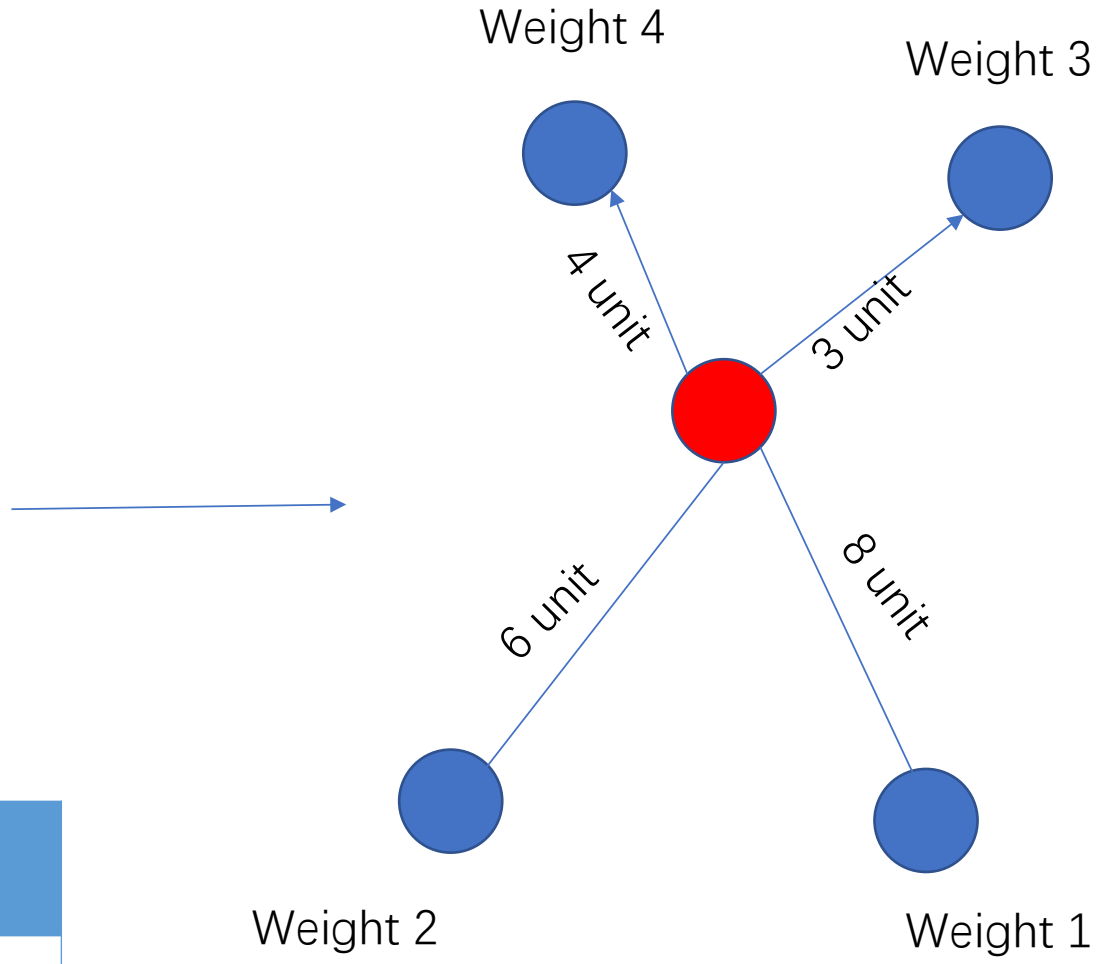
Process

3. Using the median weight to choose the optimal hospital

The script uses the Near tool to calculate distance and then adds and calculates a new field for the weighted rank using this equation: $weight = rank\ /(distanc\ e\ )\hat{}(1/2)$

This method is learned from Henry Bernberg report in last assignments.

HOSPITAL	MEDIAN WEIGHTED
A	=4/ (4^0.5)
B	=3/ (3^0.5)
C	=2/ (6^0.5)
D	=1/ (8^0.5)



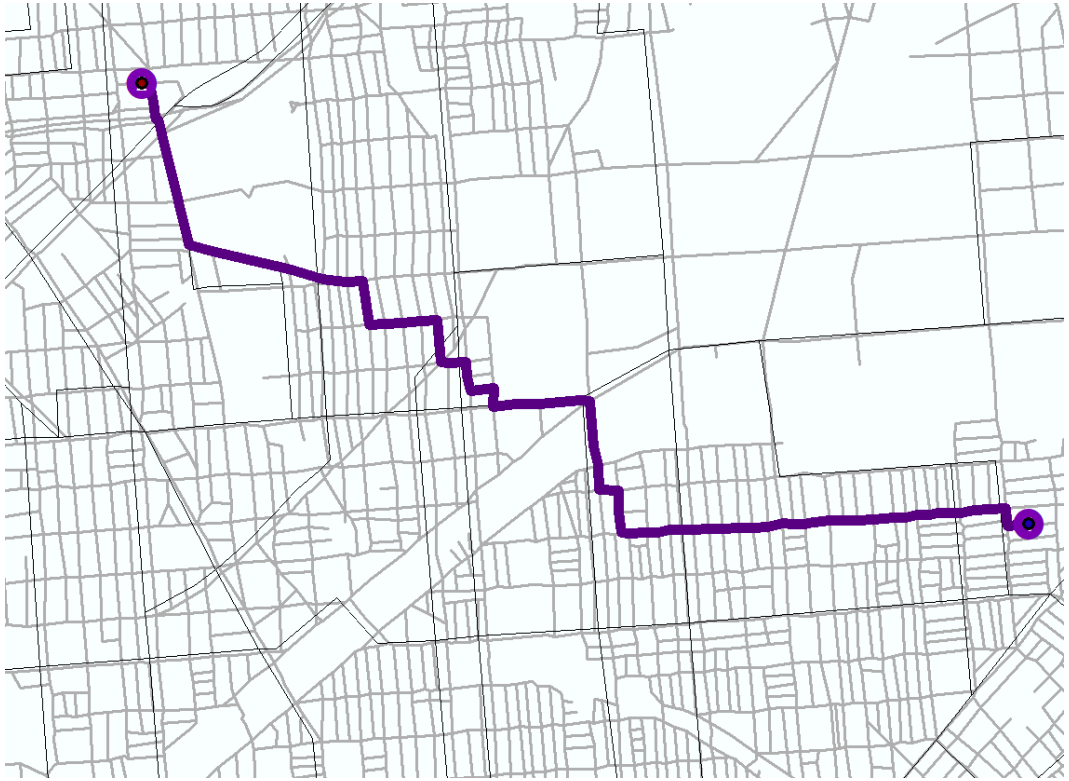
Process

The script is showed at appendix 1

4. after selected 1 or more equally weighted optimal hospitals, run the net work analyze to find the route.

4. Result display

As showed right, the tool has route a optimal hospital and a optimal route for the ambulance driver.



Discuss

In this final project, I have learned a lot from the network analysis using the Arcpy. And also I realized there are a lot of limitations for this tool.

Limitation:
1 traffic flow is not for the real time. And when I calculate the weight, there are only two variables, one is the traffic flow, another is the population density. So if we wan tot improve the accuracy, we should add more valuable and contributable variables.

2 In the final project, we only calculate the impedance based on the weight, which means that the whole impedance is not calculate. So in next time, I will use the impedance of all the route, which means I will add the impedance in the network analyze, instead of before it just like this project.

Reference:

The final Arcpy project—Wei Ying
The final Arcpy project—Henry Bernberg

APPENDIX 1

"""

FANGNAN DU

FINAL PROJECT

This script is for developing a tool for the ambulance driver to find the optimal hospital and the best route.

To be specific, the script is just for the virginia beach. Because i did a project for the emergency call propbility in virginia beach. So we will know where will the calls may happen. but after that, we need to know which hospital to go and what is the best route for it. So in this final project, I want to finish the following part of the project, which is helping the ambulance driver to find the optimal hospital and best route.

Data required:

-network dataset of virginia beach

-hospital dataset of virginia beach

-emergency calls dataset of virginia beach

To create an ArcToolbox tool with which to execute this script, do the following.

1 In ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new one.

2 Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.

3 Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.

4 In this Add Script dialog box, use Label to name the tool being created, and press Next.

5 In a new dialog box, browse to the .py file to be invoked by this tool, and press Next.

6 In the next dialog box, specify the following inputs (using dropdown menus wherever possible) before pressing OK or Finish.

DISPLAY NAME	DATA TYPE	PROPERTY>DIRECTION>VALUE
Address Number	Long	Input(required)

```
        Street Orientation          String              Input(optional)
        Street Name                 String              Input(required)
        Street type                 String              Input(optional)
        EmergencyType               String              Input(required)
        Go to emergency             String              Input(optional)
        Go to hospital              String              Input(optional)
        Data Source                 Folder              Input(required)
        Output                      workspace           Output(required)

"""
#Import necessary system modules
import arcpy
import sys, string, traceback
import os.path as op
try:
    #Set up the environment and allow the overwite
    arcpy.env.overwriteOutput = True
    #Check out the Network Analyst extension license
    arcpy.CheckOutExtension("Network")
    #-------------------------set the parameters--------------
    #Input Data source
    DataSource = arcpy.GetParameterAsText(0)
    #Before we run this model, we need to building the data set for the network Analyst
    NetworkDataset = DataSource + "\Street_ND.nd"       # network dataset
    Emergency = DataSource + "\Emergency.shp"       # Basic point layers of specific emergency Point
    Hospitals = DataSource + "\Hospitals.shp"       # Point layer for all hospital locations
    outGeodatabase = arcpy.GetParameterAsText(1)    # output workspace route
```

```
# Emergency call Location
NO = arcpy.GetParameterAsText(2)
direct = arcpy.GetParameterAsText(3)
street = arcpy.GetParameterAsText(4)
type = arcpy.GetParameterAsText(5)
# Hospital requirement
EmergencyType = arcpy.GetParameterAsText(6)
Ambulance = arcpy.GetParameterAsText(7)
DriveTo = arcpy.GetParameterAsText(8)


#--------------------emergency location create and hospital selection----------
arcpy.MakeFeatureLayer_management(Emergency, "lyr")     # Make a layer from the feature class
# Incident address selection
Start = NO+" "+direct+" "+street+" "+type
Start = str(Start)
#select point by location name
arcpy.SelectLayerByAttribute_management("lyr", "NEW_SELECTION", '"Address" = 'Start'')
# copy the selecttion to a new point shapefile
arcpy.CopyFeatures_management("lyr", "Incidents")
Incidents = "Incidents"
arcpy.AddMessage('Incident Address:' + Start)

# Hospital Selection
# When the ambulance recieved the message, there will be a prioprity type, which indicates the hospital TYPE
# if the priority is 1 ,then choose the highest level hospital
if EmergencyType == "1":
```

```
    arcpy.Select_analysis(Hospitals, "HosSelection", "'CallPriority" = '1'")
elif EmergencyType == "2":
    arcpy.Select_analysis(Hospitals, "HosSelection", "'CallPriority" = '2'")
elif EmergencyType == "2":
    arcpy.Select_analysis(Hospitals, "HosSelection", "'CallPriority" = '3'")
else:
    arcpy.CopyFeatures_management(Hospitals,"HosSelection")


#----------add the weighted impedance to select the best routed hospital before network analyse
#calculate the distance from the hospital to emergency call
arcpy.Near_analysis(Hospitals, Incidents)
## Add a weight field for impedance( the impedance is calculated by another script which showed at the end of the report)
arcpy.AddField_management(Hospitals, "weight", "DOUBLE")
# Calculate the distance based weigh of each selected hospital based on their median ranking
arcpy.CalculateField_management(Hospitals, "weight", "( [median] / (Sqr ( [NEAR_DIST] ) ) ) * 100", "VB", "")
# Determine the maximum wieghted ranking amongst the hospitals
arcpy.Statistics_analysis(Hospitals, "Hos_max", "weight MAX", "")
# Join the maximum wiegh back to the hospitals locations
arcpy.JoinField_management(Hospitals, "weight", "Hos_max", "MAX_weight", "")
# Select the hospitals locaiton with the highest weighted rank or equally 2 or more highest weighted rank
Hospitals = arcpy.Select_analysis(Hospitals, "Hospitals", "'FREQUENCY" > 0')

#----------after selected 1 or more equally weighted optimal hospitals, run the net work analyse to find the rount--------
# Set up some constant
# network analysis route layer
outRoutes = "Routes"
```

```
    # network analysis direction data
    outDirections = "Directions"
    # network analysis selected hospital
    outClosestHospital = "ClosestHospital"
    # travel cost measure unit
    measurement_units = "Minutes"
    # travel from hospital to incident site
    Travel_Direction = "TRAVEL_FROM "

    #Direction Set
    Populate_Directions = "True"
    Directions_Language = "en"
    Directions_Distance_Units = "Mile"
    Directions_Style_Name = "NA Desktop"

    # Target point layers
    Incidents = "Incidents"
    Facilities = Hospitals
    Maximum_Facilities_to_Find = 1

    # Run FindClosestFacilities. Choose to find only the closest facility.

    arcpy.na.FindClosestFacilities( Incidents, Facilities, measurement_units, NetworkDataset, outGeodatabase, outRoutes,outDirections,outClosestHospital,
Number_of_Facilities_to_Find=1)
    #-----------------------results DISPLAY---------------------------
    # Add the hospital, route and direction feature classes to a new dataframe
    currentMap = arcpy.mapping.MapDocument("CURRENT")
```

```
currentDataFrame = currentMap.activeDataFrame
layerToBeDisplayed1   = arcpy.mapping.Layer(outRoutes)
arcpy.mapping.AddLayer(currentDataFrame, layerToBeDisplayed1,"TOP")
layerToBeDisplayed2   = arcpy.mapping.Layer(outClosestHospital)
arcpy.mapping.AddLayer(currentDataFrame, layerToBeDisplayed3,"TOP")

del currentMap

sequenceOfShapefileRecords = arcpy.SearchCursor(outClosestHospital)
# Loop through that list, printing each field's type and name
arcpy.AddMessage ( " Optimal hospital is found.")

# Get the ID of finded hospital in order to find the specific info in orginal feature
for nextRecord in sequenceOfShapefileRecords:
    ID = str(nextRecord.getValue(ORIG_FID))
del sequenceOfShapefileRecords

sequenceOfShapefileRecords = arcpy.SearchCursor(Facilities)
# Print out hospital information
for nextRecord in sequenceOfShapefileRecords:
      if str(nextRecord.getValue(FID)) == ID:
          arcpy.AddMessage ("Hospital Name:"+ str(nextRecord.getValue(Facilities))
          arcpy.AddMessage ("Hospital Address: " + str(nextRecord.getValue(Street))
del sequenceOfShapefileRecords

# Print out direction text
Arcpy.AddMessage ("Route Direction:\n")
```

```
    sequenceOfShapefileRecords = arcpy.SearchCursor(outDirections)
    for nextRecord in sequenceOfShapefileRecords:
        arcpy.AddMessage( str(nextRecord.getValue(Text)))

    # Travel distance & time
    del sequenceOfShapefileRecords
    sequenceOfShapefileRecords = arcpy.SearchCursor(outRoutes)
    for nextRecord in sequenceOfShapefileRecords:
        arcpy.AddMessage( "\n"+ str(nextRecord.getValue(Total_Minutes)))
        arcpy.AddMessage( str(nextRecord.getValue(Total_Miles)))

except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage     = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

#APPENDIX 2
Code for doing the data preparation
To calculate the weight for the data preparation

```
# Import arcpy module
#Import necessary system modules
import arcpy
import sys, string, traceback
import os.path as op
try:
    import arcpy
    # Local variables:
    Street_Ranges = "Street_Ranges"
    aadt_report_2013_2017_5_csv = "aadt_report_2013-2017-5.csv"
    aadt_report_2013_2017_5_Layer = "aadt_report_2013-2017-5_Layer"
    KernelD_aadt5 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\KernelD_aadt5"
    Reclass_Kern3 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\Reclass_Kern3"
    Tract_Pop__Density_per_mi² = "Tract_Pop._Density_per_mi²"
    Tract_Pop_PolygonToRaster4 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\Tract_Pop_PolygonToRaster4"
    Reclass_Trac1 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\Reclass_Trac1"
    rastercalc1 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\rastercalc1"
    RasterT_rasterc1 = "C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\RasterT_rasterc1"
    combined_shp = "C:\\Users\\49469\\Desktop\\arcpy\\combined.shp"

    # Process: Make XY Event Layer
    arcpy.MakeXYEventLayer_management(aadt_report_2013_2017_5_csv, "Field6", "Field5", aadt_report_2013_2017_5_Layer,
"GEOGCS['GCS_WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_1984',6378137.0,298.257223563]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]];-400 -400
1000000000;-100000 10000;-100000 10000;8.98315284119522E-09;.001;.001;IsHighPrecision", "")

    # Process: Kernel Density
    arcpy.gp.KernelDensity_sa(aadt_report_2013_2017_5_Layer, "NONE", KernelD_aadt5, "1.44286400212047E-03", "", "SQUARE_MAP_UNITS", "DENSITIES", "PLANAR")

    # Process: Reclassify
    arcpy.gp.Reclassify_sa(KernelD_aadt5, "VALUE", "0 434 1;434 4000 2;4000 400000 3", Reclass_Kern3, "DATA")

    # Process: Polygon to Raster
    arcpy.PolygonToRaster_conversion(Tract_Pop__Density_per_mi², "PopSqMi", Tract_Pop_PolygonToRaster4, "CELL_CENTER", "NONE", ".0014")
```

```
    # Process: Reclassify (2)
    arcpy.gp.Reclassify_sa(Tract_Pop_PolygonToRaster4, "VALUE", "0 58 1;58 2000 2;2000 12000 3", Reclass_Trac1, "DATA")

    # Process: Raster Calculator
    arcpy.gp.RasterCalculator_sa("(\"%Reclass_Kern3%\"+2*\"%Reclass_Trac1%\") / 2", rastercalc1)

    # Process: Raster to Polygon
    arcpy.RasterToPolygon_conversion(rastercalc1, RasterT_rasterc1, "SIMPLIFY", "Value")

    # Process: Spatial Join
    arcpy.SpatialJoin_analysis(Street_Ranges, RasterT_rasterc1, combined_shp, "JOIN_ONE_TO_ONE", "KEEP_ALL", "OBJECTID_1 \"OBJECTID_1\" true true false 10 Long 0
10 ,First,#,Street_Ranges,OBJECTID_1,-1,-1;NAME \"NAME\" true true false 80 Text 0 0 ,First,#,Street_Ranges,NAME,-1,-1;TYPE \"TYPE\" true true false 80 Text 0
0 ,First,#,Street_Ranges,TYPE,-1,-1;DIRECTION \"DIRECTION\" true true false 80 Text 0 0 ,First,#,Street_Ranges,DIRECTION,-1,-1;FULL_NAME \"FULL_NAME\" true true false 80 Text 0
0 ,First,#,Street_Ranges,FULL_NAME,-1,-1;FROM_ \"FROM_\" true true false 10 Long 0 10 ,First,#,Street_Ranges,FROM_,-1,-1;TO_ \"TO_\" true true false 10 Long 0
10 ,First,#,Street_Ranges,TO_,-1,-1;RIGHT_FROM \"RIGHT_FROM\" true true false 10 Long 0 10 ,First,#,Street_Ranges,RIGHT_FROM,-1,-1;RIGHT_TO \"RIGHT_TO\" true true false 10 Long 0
10 ,First,#,Street_Ranges,RIGHT_TO,-1,-1;LEFT_FROM \"LEFT_FROM\" true true false 10 Long 0 10 ,First,#,Street_Ranges,LEFT_FROM,-1,-1;LEFT_TO \"LEFT_TO\" true true false 10 Long 0
10 ,First,#,Street_Ranges,LEFT_TO,-1,-1;ST_FROM \"ST_FROM\" true true false 80 Text 0 0 ,First,#,Street_Ranges,ST_FROM,-1,-1;ST_TO \"ST_TO\" true true false 80 Text 0
0 ,First,#,Street_Ranges,ST_TO,-1,-1;CLASS \"CLASS\" true true false 80 Text 0 0 ,First,#,Street_Ranges,CLASS,-1,-1;ST_RANGE \"ST_RANGE\" true true false 80 Text 0
0 ,First,#,Street_Ranges,ST_RANGE,-1,-1;SHAPElen \"SHAPElen\" true true false 24 Double 15 23 ,First,#,Street_Ranges,SHAPElen,-1,-1;Shape_Length \"Shape_Length\" false false true 0 Double 0
0 ,First,#,C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\RasterT_rasterc1,Shape_Length,-1,-1;Shape_Area \"Shape_Area\" false false true 0 Double 0
0 ,First,#,C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\RasterT_rasterc1,Shape_Area,-1,-1;ID \"ID\" true true false 0 Long 0
0 ,First,#,C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\RasterT_rasterc1,ID,-1,-1;GRIDCODE \"GRIDCODE\" true true false 0 Long 0
0 ,First,#,C:\\Users\\49469\\Documents\\ArcGIS\\Default.gdb\\RasterT_rasterc1,GRIDCODE,-1,-1", "WITHIN", "", "")

except Exception as e:
    # If unsuccessful, end gracefully by indicating why
    arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
    # ... and where
    exceptionreport = sys.exc_info()[2]
    fullermessage    = traceback.format_tb(exceptionreport)[0]
    arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```