Title: **LAB 02, PART II.**

Notice: Dr. Bryan Runck
Author: **Michael Felzan**
Date: **March 6, 2021**

Project Repository: https://github.com/fezfelzan/GIS5572.git

Abstract

This report demonstrates an ETL which operates in an ArcGIS Pro Jupyter Notebooks that inputs data into a cost surface model and finds the least cost path from point A to point B, given certain weighted criteria. The input data for this path includes the start/end XY coordinate pairs, MN hydrography data, MN county shapefile, MN roadways data, MN land cover raster, and DEM rasters for all relevant counties (MN counties which the path could likely cross). After generating a weighted cost surface with boundaries (NO DATA pixels), a least cost path is generated from a cost distance raster and a back-link raster. The output achieved using the methodology described in this report seems to be valid, based on a visual analysis (described in the Results).

Problem Statement

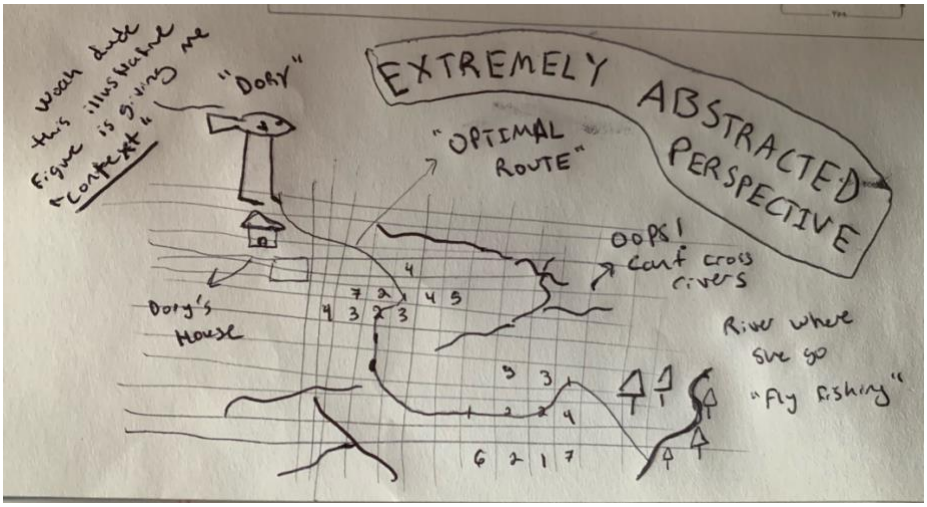


Figure 1: *Illustrative figure of the lab objective*

The objective of the second part of Lab 2 is to create an ETL for data to go into a cost surface model, along with creating the cost surface model itself. The cost surface model will describe how a fictional character (Dory) may reach her destination (Whitewater State Park) in the least distance possible, while considering that Dory:

- will not cross over rivers
- prefers not to cross through agricultural land
- prefers to take the route with the least slope possible

Table 1. *Defining the Problem*

#	Requirement	Defined As	Spatial Data	Attribute Data	Dataset	Preparation
1	Road network	Raw input dataset from MNDOT (county and local roads)	Road geometry		Mn GeoSpatial Commons	Extract only county and local roads; clip to study area
2	Rivers	Raw input dataset from MN Geospatial Commons	River (line) geometry	Type of body of water	Mn Geospatial Commons	Extract only rivers in study area
3	Incline	Slope calculated from DEM rasters (from MnGeo)	Raster	(change in) elevation	MnGeo	Mosaic/merge DEMs from all relevant counties into one; reclassify on 1-10 scale
4	Land cover	NLCD 2011 Land Cover raster		Classification of land cover	NLCD dataset; mn geospatial commons	Clip to study area; reclassify on 1-10 scale
5	Buffer zone (width) of river features	30 meters			Mn Geospatial Commons	
6	Buffer zone (width) of road features	100 meters			Mn Geospatial Commons	
7	Dory Start and end points	44.127985, -92.148796; 44.13170, -92.14856	XY coordinates		Google Maps	Bring cords into CSV; XY table to points

Input Data

The first data that's required is the geographic coordinates of "point A" and "point B." Next, we need a shapefile that represents the geographic extent of our study area (as we will be dealing with a large amount of dense data, so we do not want to create a Cost Surface map for the entire state of Minnesota). The MN County Boundaries shapefile will be used for this. Data for the rivers in MN is required, as Dory cannot cross through rivers on her way to the park. For this, the National Hydrology Data for MN is used. Data for the land cover types in MN is also necessar, as Dory prefers *not* to cross through agricultural land when she can. For the MN land cover data, the NLCD 2011 MN land cover raster was used. Additional data which reflects changes in elevation is required for analysis, as Dory prefers not to navigate areas of high slope when she can. Because Dory could likely cross through Winona, Wabasha, and Olmsted County to travel from her start point to her destination, DEM raster data for all three of those counties is used here. (data downloaded from the MN DNR FTP server).

Table 2. *Input Data*

#	Title	Purpose in Analysis	Link to Source
1	Minnesota Roads	Determining areas where Dory may cross over rivers (bridges)	Mn GeoSpatial Commons
2	MN Rivers	Determining boundaries for which Dory can't cross over	https://gisdata.mn.gov/dataset/f337e36f-83e0-4ca7-b47e-736d98d05ca5/resource/2717962b-8eed-4d93-9db1-88e83147f894
3	Land Cover	Determining agricultural land areas vs. non-agg land areas (preference of travelling over non agg land used in cost surface analysis)	https://gisdata.mn.gov/dataset/biota-landcover-nlcd-mn-2011
4	DEM Rasters for Wabasha, Winona, and Olmsted County	Determining amount of incline, to be used in weighted cost surface analysis	https://resources.gisdata.mn.gov/pub/data/elevation/lidar/county/
5	MM Counties shapefile	Creating a "study area extent" that other layers may be clipped to	https://gisdata.mn.gov/dataset/bdry-counties-in-minnesota
6	Dory's start/end XY coords	To be used in the Cost Distance, Back-Link, and Cost Path analyses	(given; https://www.google.com/maps)

Methods

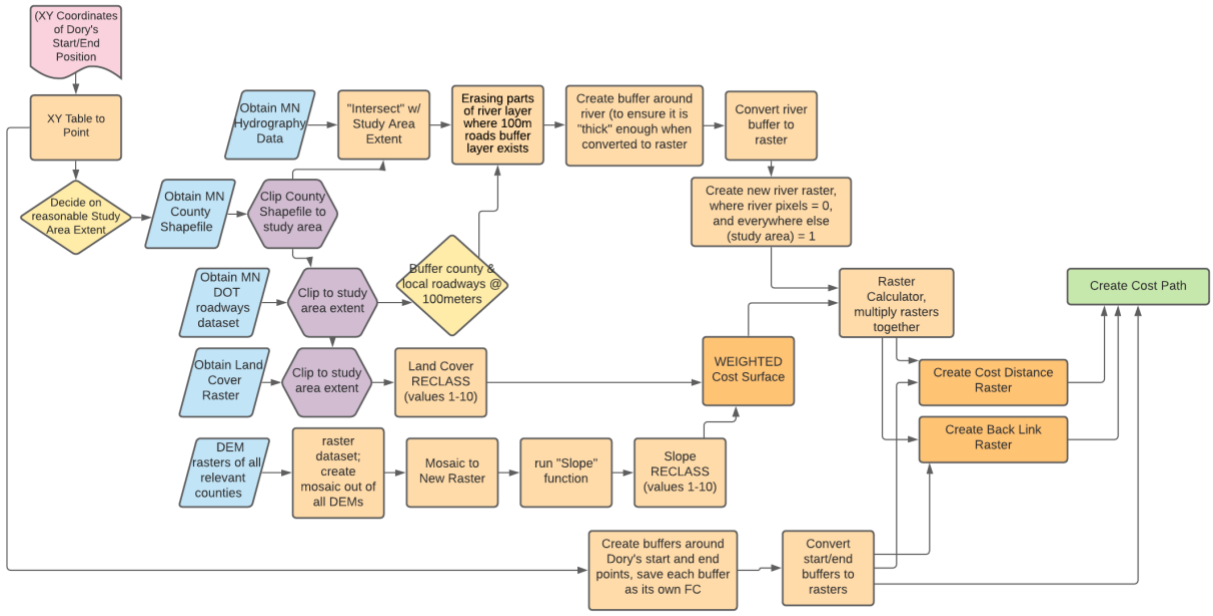


Figure 2. *Flow Diagram depicting the process of creating a Least Cost Path out of a Cost Distance Raster and a Back Link Raster. The input data for this path includes Start/End XY coordinate pairs, MN hydrography data, MN county shapefile, MN roadways data, MN land cover raster, and DEM rasters for all "relevant" counties (counties that Dory could feasibly cross thru in order to get from point A to point B).*

As visualized in **Figure 2**, there are many different datasets required in order to calculate the "least cost path" for Dory to get from point A to point B. The first data that's required is the geographic coordinates of "point A" and "point B." We know that Dory lives at 44.127985, -92.148796, and is navigating to "Whitewater State Park," though we were not given the coordinate pair for the park. However, we can find the decimal coordinates for Whitewater State Park by punching in the location on Google Maps and right clicking the pin (**Figure 3)**.

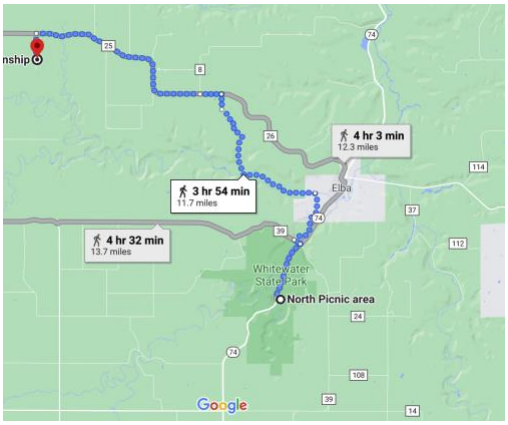


Figure 3: *Using Google Maps to find the decimal coordinate pairs of locations.*

After obtaining both coordinate pairs for Dory's start and end positions, we can write them out in a .CSV file, and import them into ArcPro using the XYTableToPoint() function:

XY Table to Point

```
arcpy.management.XYTableToPoint(r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\arcii_labii_lat_long
                                r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\latlongcoords",
                                "Long (X)",
                                "Lat (Y)",
                                None,
                                "GEOGCS['GCS_North_American_1983',DATUM['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,29
```

[4]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\latlongcoords.shp

Figure 4: *Plotting Dory's start and end points as XY coordinates in ArcPro*

Next, we need to obtain a shapefile that represents the geographic extent of our study area (as we will be dealing with a large amount of dense data, so we do not want to create a Cost Surface map for the entire state of Minnesota). To obtain a "geographic extent" bounding polygon, I decided to go with the three counties Dory could (likely) cross through on her way to the state park, which are Wabasha, Winona, and Olmsted. I imported a shapefile containing all MN counties, and then ran a "Select By Attributes" on the data, copying/exporting only the MN counties that had those county names:

(SELECT BY ATTRIBUTES TO GET "RELEVANT_MN_COUNTIES)

```
mn_county_shp = r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\shp_bdry_counties_in_minnesota (1)\m
```

```
relevant_mn_counties = arcpy.management.SelectLayerByAttribute(mn_county_shp,
                                                               "NEW_SELECTION",
                                                               "CTY_NAME = 'Winona' Or CTY_NAME = 'Wabasha' Or CTY_NAME = 'Olmst
arcpy.CopyFeatures_management(relevant_mn_counties, 'relevant_mn_counties')
```

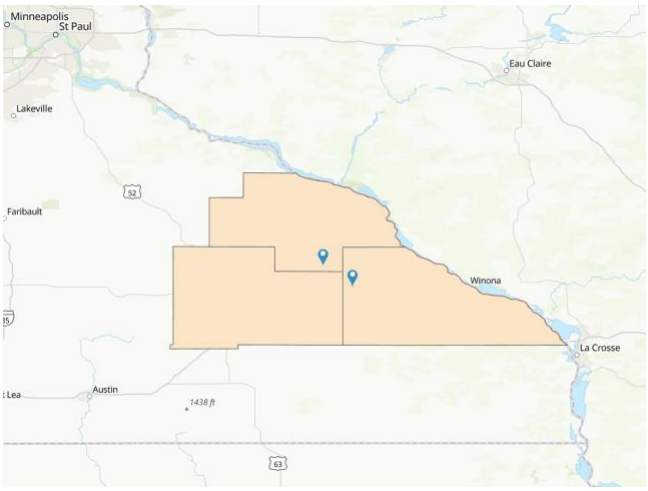


Figure 5: *Clipped MN County Boundaries shapefile (what will serve as the study area extent)*

Next, we need data for the rivers in MN, as Dory cannot cross through rivers on her way to the park. For this, I downloaded the National Hydrology Data for MN, and copied only *river* features that *intersected the study area* (three counties explained above) to a new layer.

```
arcpy.analysis.Intersect(r"Hydrography\NHDFlowline #;relevant_mn_counties #",
                         r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\relevant_mn_rive
                         "ALL",
                         None,
                         "INPUT")
```

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\relevant_mn_rivers

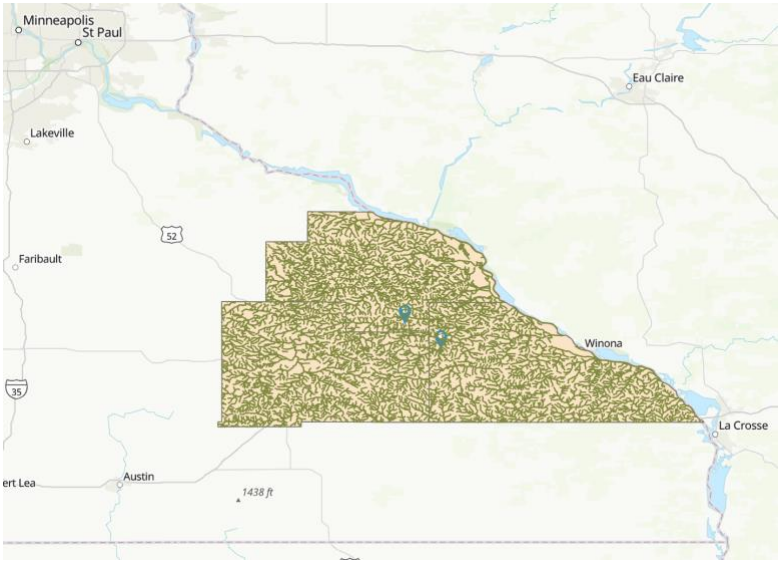


Figure 6: *MN rivers shapefile, clipped to study area extent.*

We also need data for the land cover types in MN, as dory prefers *not* to cross through agricultural land when she can. For the MN land cover data, I already had the NLCD 2011 MN land cover raster saved onto my computer, so I decided to use that data and clip it to the study area extent. The tool I used to clip the raster was arcpy's Clip (Raster) tool.

Clip Land Cover Raster to Study Area

```
path_to_landcover = r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\lc_mn_2011.gdb\lc_mn_2011"

arcpy.management.Clip(path_to_landcover,
                      "524966.6376 4853462.8394 637916.1448 4922619.9426",
                      r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\lc_mn_2011_Clip2",
                      "relevant_mn_counties",
                      "255",
                      "NONE",
                      "NO_MAINTAIN_EXTENT")
```

15]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\lc_mn_2011_Clip2

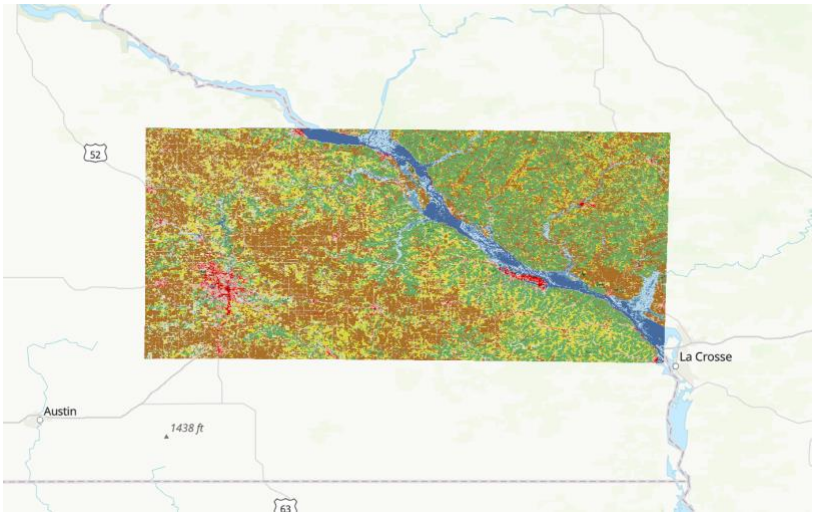


Figure 7: *NLCD 2011 MN Land Use raster, clipped to study area*

We need data which represents changes in elevation, as Dory prefers not to navigate areas of high slope when she can. Because Dory could likely cross through Winona, Wabasha, and Olmsted County to travel from her start point to her destination, we need DEM raster data for all three of those counties. I downloaded all three of these county DEM rasters from the MN DNR FTP server (under elevation data by county). Because these files are huge (even at the 3m level), I decided to resample each DEM to become a lower resolution. For the cell size of the new resolution, I chose to use the cell size of my land cover data, as I figured all of my data should be as accurate as the "most coarse"/least accurate dataset (the cell sizes for the NLCD 2011 land cover raster are somewhat large). I specifically used the bilinear sampling method:

Resampling each DEM raster to be lower resolution

```
winona_3mDEM = r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\elevation_data.gdb\dem_3m_m"

wabasha_3mDEM = r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\wabasha_elevation_data.gdb\dem_3m_m"

olmsted_3mDEM = r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\PART_TWO\olmsted_elevation_data.gdb\dem_3m_m"


arcpy.management.Resample(winona_3mDEM,
                          r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\Winona_dem_3m_m
                          "3 3",
                          "BILINEAR")
arcpy.management.Resample(wabasha_3mDEM,
                          r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\Wabasha_dem_3m_
                          "3 3",
                          "BILINEAR")
arcpy.management.Resample(olmsted_3mDEM,
                          r"C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\Olmsted_dem_3m_
                          "3 3",
                          "BILINEAR")
```

17]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\Olmsted_dem_3m_m_Resample

To merge all three DEMs into one layer, I created an empty raster dataset, and then mapped a mosaic of all three DEMs to that dataset. Once all three DEMs were in a single mosaic, I then exported the mosaic to a new raster. With all of the DEMs merged into one raster, I then used the arcpy.sa.Slope() function to output a raster with pixel values pertaining to the slope or steepness of all points in the study area. Once this slope raster was created, I then *reclassified* the raster on a 1 to 10 scale (as the land cover preference cost raster and the incline preference cost raster need to be on the same scale in order to perform map algebra functions on both).

Running "Slope" function on simplified DEM covering entire study area

```
out_raster = arcpy.sa.Slope("mosaic_rast",
                            "DEGREE",
                            0.304800609601219,
                            "PLANAR",
                            "METER"); out_raster.save(os.path.join(default_gdb, "slope_grid"))
```

Reclassifing slope to be on 1 - 10 scale:

```
out_raster = arcpy.sa.Reclassify("slope_grid",
                                 "VALUE",
                                 "0 7.127013 1;7.127013 14.254025 2;14.254025 21.381038 3;21.381038 28.508051 4;28.508051 35.635
                                 "DATA"); out_raster.save(os.path.join(default_gdb, "slope_reclass"))
```

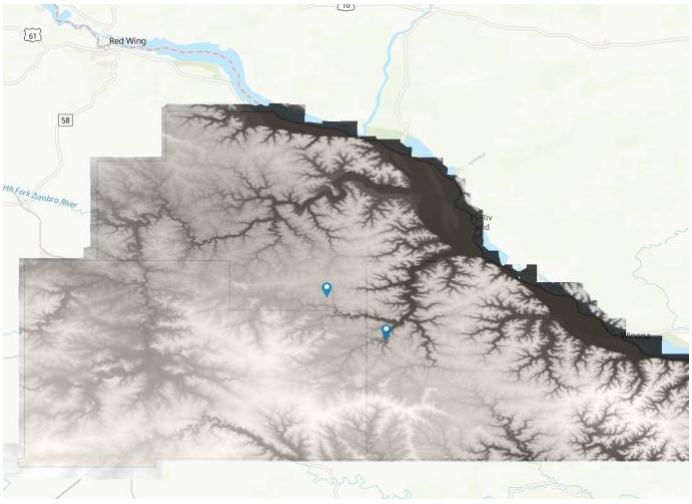


Figure 8: *Mosaic (merge) of the Winona County, Wabasha County, and Olmsted County 3m DEMs*

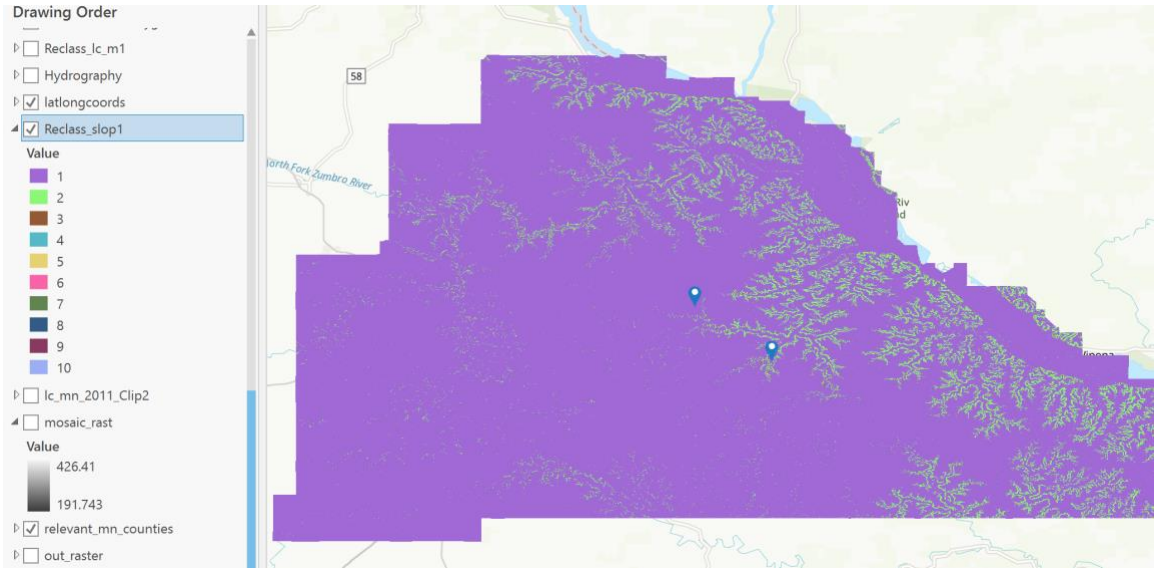
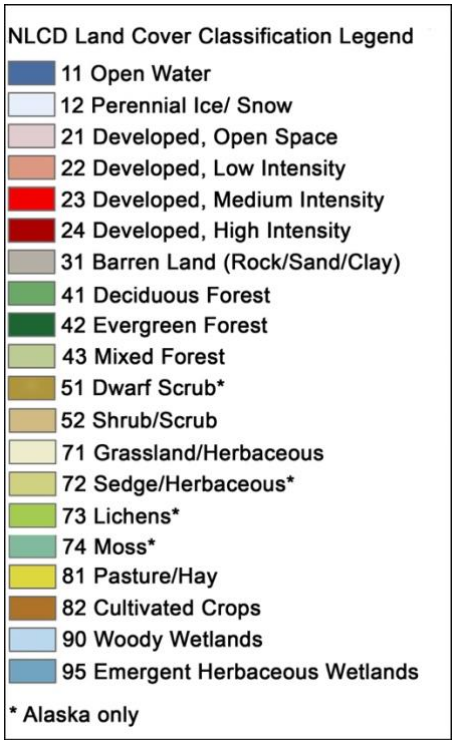


Figure 9: *Reclassifying the elevation data using a 1 – 10 classification scale.*

As stated above, the land use raster and the slope raster need to be on the same "scale" in order to apply map algebra functions between them. So now we must reclassify land use to be on a 1 – 10 scale. ***Note:*** this step requires some subjective decisions to be made… first we must decide what land uses are considered preferred vs. not preferred -- I chose to make 41,42, 43, 51, 52, and 71 "preferred" land use types, 81, 82, 90, 95 "not preferred" land use types (as 81 and 82 are agricultural land, and 90 and 95 seem muddy), the developed land use types (21, 22, 23, 24) as somewhere in the middle (as Dory would rather cross over a bridge/road than go thru mud), and open water (11) the 'max' least preferred. Next, we must assign values or 'weights' to the levels of preferred land classes – used '1' (most preferred/ least cost) as the grassland areas, '4' as the developed areas, '7' as the agricultural areas, and '10' for the open water areas (though this value does not *really* matter, as the areas containing rivers will later be set up to be boundaries that Dory cannot cross).



Reclassifying Land Use raster to be on 1 - 10 scale

Subjective decisions were made here --

1 = grassland areas (most preferred), 4 = developed areas (bridges...preferred over crossing thru mud or rivers), 7= agricultural land (not preferred), 10 = open water (wont cross anway, as rivers will be set as boundaries)

```
out_raster = arcpy.sa.Reclassify("lc_mn_2011_Clip2",
                                 "Value",
                                 "11 10;21 4;22 4;23 4;24 4;31 4;41 1;42 1;43 1;52 1;71 1;81 7;82 7;90 4;95 4",
                                 "DATA"); out_raster.save(os.path.join(default_gdb, "Reclass_lc_m1"))
```

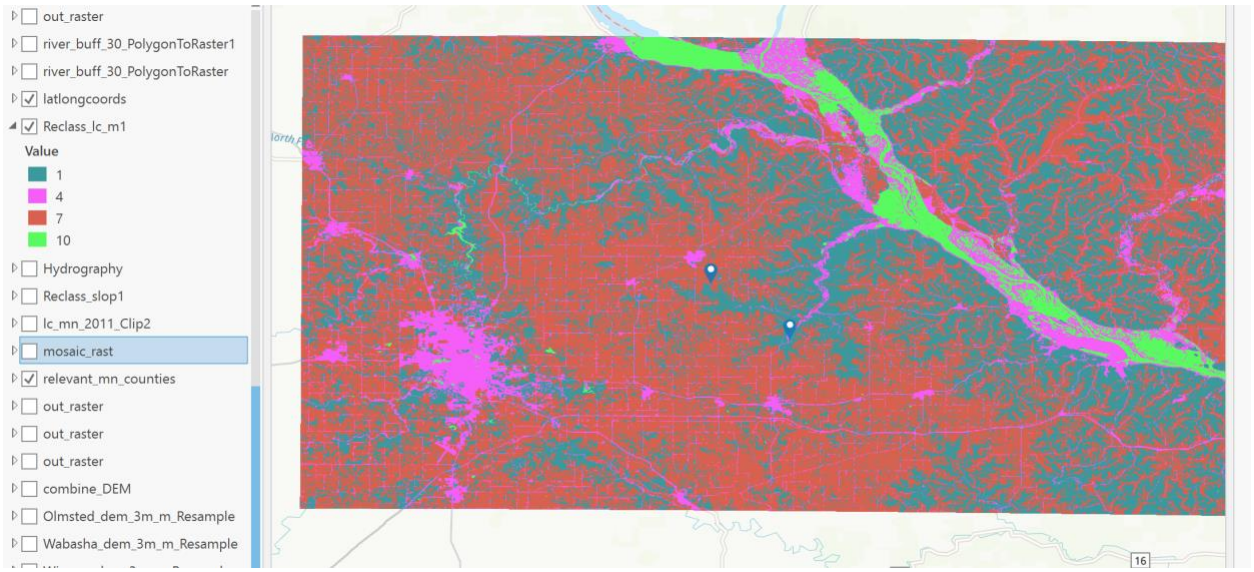


Figure 10: *Reclassifying Land Use raster using a 1 – 10 classification scale.*

Now that both our Land Use raster and Slope raster are on a 1-10 pixel classification system, we may add them together to create an initial Cost Surface raster. Here I made another subjective decision (influenced by the wording of the prompt), where I decided to *weight* land use type as being twice as important to Dory than the level of incline. To create a weighted cost surface raster, considering this criteria, I used the RasterCalculator() tool, and added .66*(Land Use) + .34*(Slope).

Creating WEIGHTED Cost Surface, using Land Use Reclass layer, and Slope Reclass layer.

The amount of weight given to slope vs. land use was subjective, but based on the description of Dory's preferences. I interpreted the prompt as, it is twice as important to Dory to navigate areas of land that are not muddy farmland, than navigating areas of high inclines/slopes. The below calculation reflects this weighted analysis

```
out_rc_multi_raster = RasterCalculator(["Reclass_lc_m1", "slope_reclass"],
                                       ["x", "y"], ".66*(x)+.34*(y)")
out_rc_multi_raster.save(os.path.join(default_gdb,"weighted_cost_surface3"))
```

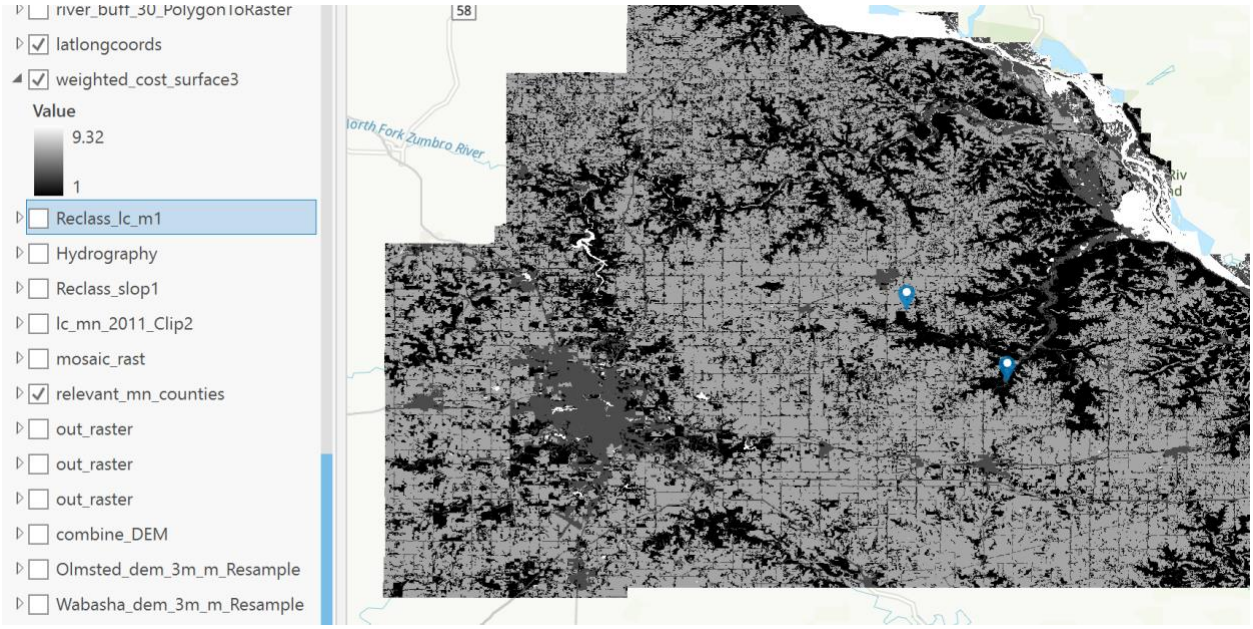


Figure 11: *Weighted Cost Surface raster, made from the individual Slope and Land Use cost surface rasters.*

As alluded to above, we ***also*** need a dataset which represents roadways in Minnesota, as it is assumed that county/ local roads which cross over rivers may be used by Dory as bridges. Following the process outlined in **Figure 2,** we must download a MN roadways shapefile, clip the roads layer to the study area extent, and extract only county and local roads from the layer (Dory will not get on the highway to cross a river). We will then erase from the 'rivers' feature class all areas which are occupied by county and local roads. However, if we just did this, Dory may not actually be able to cross over rivers where bridges 'should' be, as the tiny gaps left by deleting the (almost infinitesimally small) line features may be blocked by adjacent pixels. So, after obtaining our roads feature class, we will then ***buffer*** the roadways. I chose a buffer value

of 100m, and although this is an overexaggeration of how large a county road would be, the value ensures that Dory will be able to cross through the rivers in areas where roads overlay them.

Erasing parts of river layer where 100m roads buffer layer exists:

step 1: creating updated river layer by erasing county roads

```
arcpy.analysis.Erase("relevant_mn_rivers",
                     "CountyBuffer",
                     os.path.join(default_gdb, "rivers_erase1"),
                     None)
```

34]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\rivers_erase1

I also buffered the (newly edited) river features to be 30m in width, to ensure Dory could not slip thru the pixel gaps in the river to reach her destination.

The next step was to convert the erased/buffered river polygon feature class into a raster. This river raster must occupy the whole study area extent. I chose to value the pixels occupied by river as "0", and everywhere else as "1", so I could multiply this raster against the previously constructed cost surface raster (all values in the cost surface raster would remain the same, except pixels containing river would be set to 0). Finally, once a cost surface raster is achieved that had river areas classified as 0, we are able to use the arcpy.sa.SetNull() function to set all pixel values with a given classification as NO DATA. This is crucial because the Cost Path function treats NO DATA pixels as ***path boundaries*.**

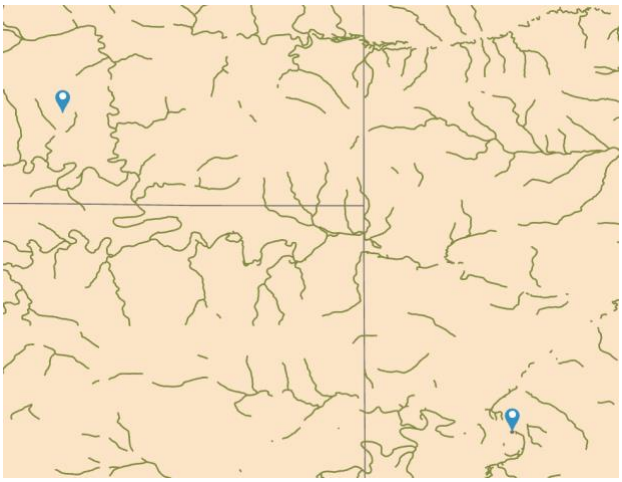


Figure 12: *River features* ***erased*** *in areas where county/local roads cross them (100m buffer)*

Setting all 0 pixel values (river) as NULL (so that the rivers will become walking boundaries)

```
out_raster = arcpy.sa.SetNull("multiplied_zeros_and_ones",
                              "multiplied_zeros_and_ones",
                              "VALUE = 0"); out_raster.save("SetNull_mult1")
```

Figure 13: *Weighted cost surface raster, where 'river pixels' have been set to NO DATA to serve as boundaries for Dory's cost path.*

One of the last pieces we need to complete our cost path analysis is Dory's 'start' and 'end' points to be *rasters* instead of *points*, and for each raster to be its own feature class. To accomplish this, I ran a 25m buffer around the XY coord pairs feature class, and then ran Select By Attributes/ copied each point-buffer feature into separate feature classes. After that, I used the Polygon to Raster tool to convert the buffer polygons into rasters, and used each points OBJECTID as the raster values. In order for a raster to be used as a start/end point, the raster must have a non-zero integer value. Because the points had OBJECTIDs of 1 and 2, I decided creating the rasters on this field was the most convenient.

Creating a buffer around Dory's start/end X/Y coords, so that these buffer areas can be transformed into rasters

```
arcpy.analysis.Buffer("latlongcoords",
                      "points_buffer",
                      "25 Meters",
                      "FULL",
                      "ROUND",
                      "NONE", None,
                      "PLANAR")
```

]3]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\points_buffer

Transporting "start place" buffered coord into its own feature class -- transforming isolated start place buffer area into raster.

```
startplace = arcpy.management.SelectLayerByAttribute("points_buffer",
                                                     "NEW_SELECTION",
                                                     "OBJECTID = 1")
arcpy.CopyFeatures_management(startplace, 'START1')
```

4]:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\START1

```
arcpy.conversion.PolygonToRaster("START1",
                                 "OBJECTID",
                                 #object id used, because value just has to be non-0 int
                                 "STARTPOINT",
                                 "CELL_CENTER",
                                 "NONE",
                                 "Reclass_lc_m1")
```

|:

Output

C:\Users\michaelfelzan\Documents\arc ii labby ii\arciiptii clean\arciiptii clean.gdb\STARTPOINT

Finally, as depicted in the **Figure 2.** flow map, now that we have our individual start/endpoint rasters, and a weighted cost surface raster (with NO DATA boundaries set up wherever rivers occur w/o bridges to cross them), we can now construct our cost path. The first step is to calculate the **Cost Distance Raster**, which uses the **start point** and the **weighted cost surface**:

Creating Cost Distance Raster

(uses start point)

```
out_distance_raster = arcpy.sa.CostDistance("STARTPOINT",
                                            "SetNull_mult1",
                                            # SetNull is the weighted cost surface
                                            # with null values where water is
                                            None, None, None, None, None, None,
                                            ''); out_distance_raster.save("COSTDIST")
```

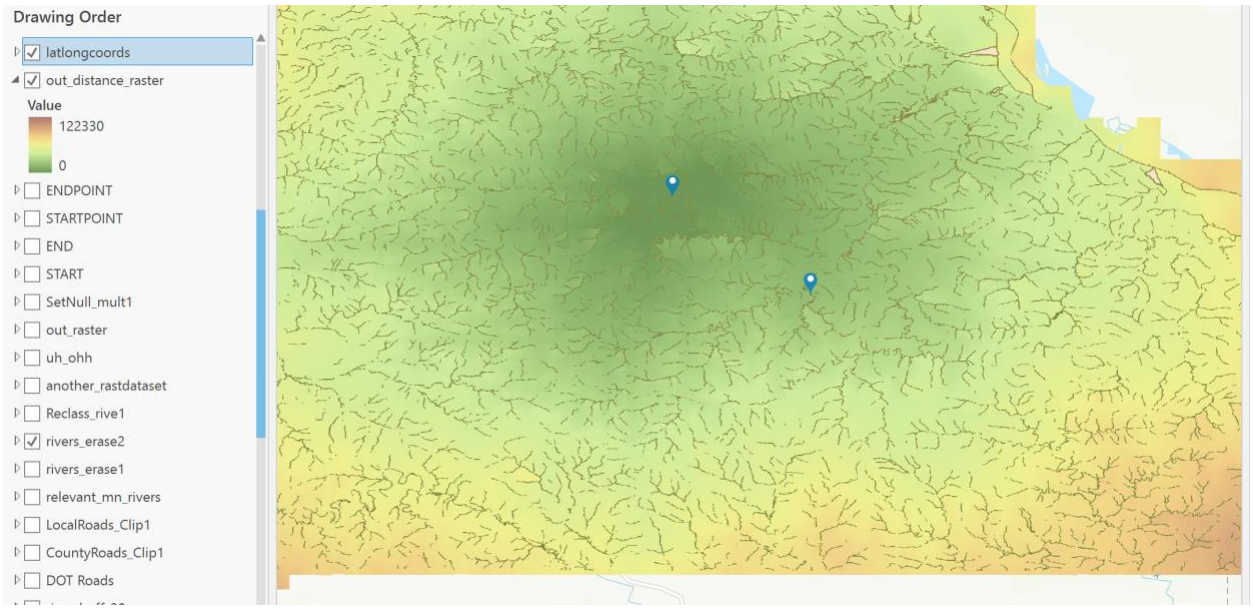


Figure 14: *Visualization of the Cost Distance raster*

Next, we need to create a **Cost Back Link Raster**, which is created using again the **start point**, and the **weighted cost surface**.

Creating Back Link Raster

(uses start point)

```
out_backlink_raster = arcpy.sa.CostBackLink("STARTPOINT",
                                            "SetNull_mult1",
                                            # SetNull is the weighted cost surface
                                            # with null values where water is,
                                            None, None, None, None, None, None, ''); out_backlink_raster.save("BACKLINK")
```

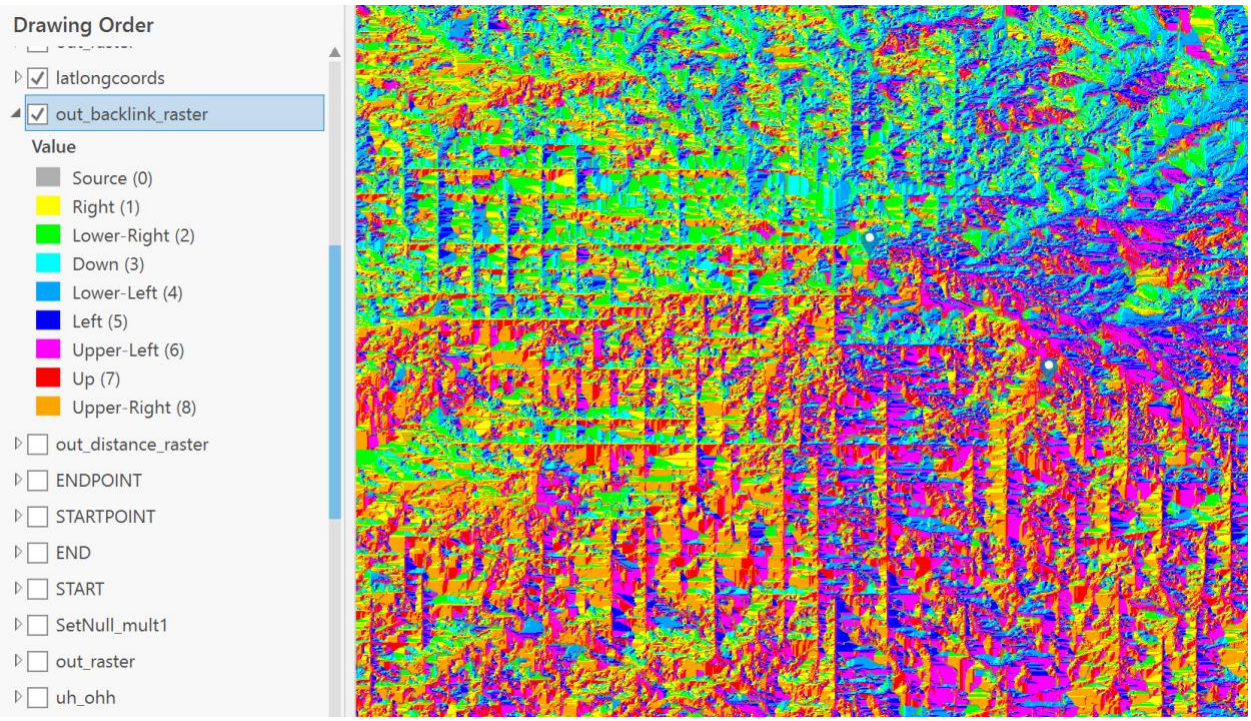


Figure 15: *Visualization of the Back Link raster*

And at last, we can calculate the **Cost Path,** which is created using the **end point** instead of the start point, the **cost distance raster**, and the **cost back link raster**.:

Creating Cost Path

(uses END point)

```
out_raster = arcpy.sa.CostPath("ENDPOINT",
                               "COSTDIST",
                               "BACKLINK",
                               "EACH_CELL",
                               "Value",
                               "INPUT_RANGE"); out_raster.save("CostPat_END1")
```

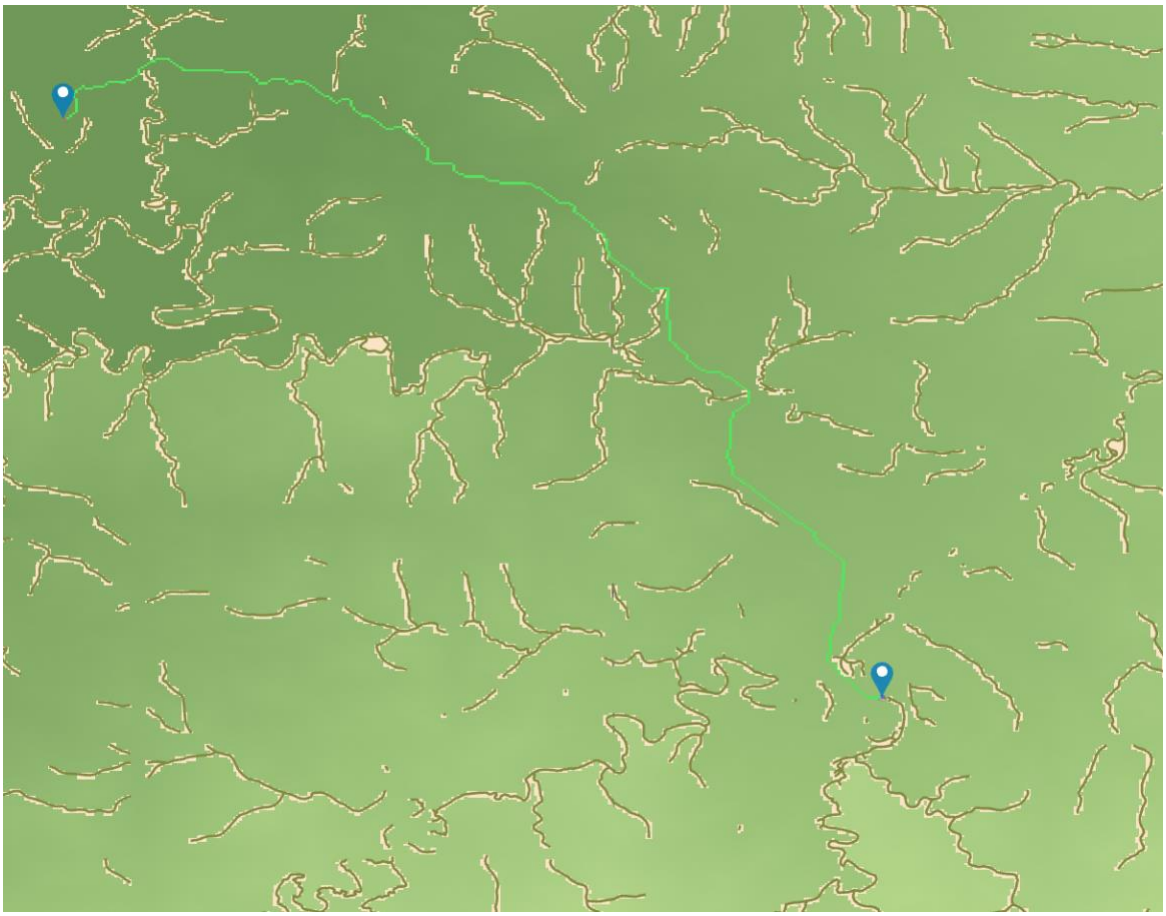


Figure 16: *Dory's optimal path (shown as the bright green line)*

Results/ Results Verification

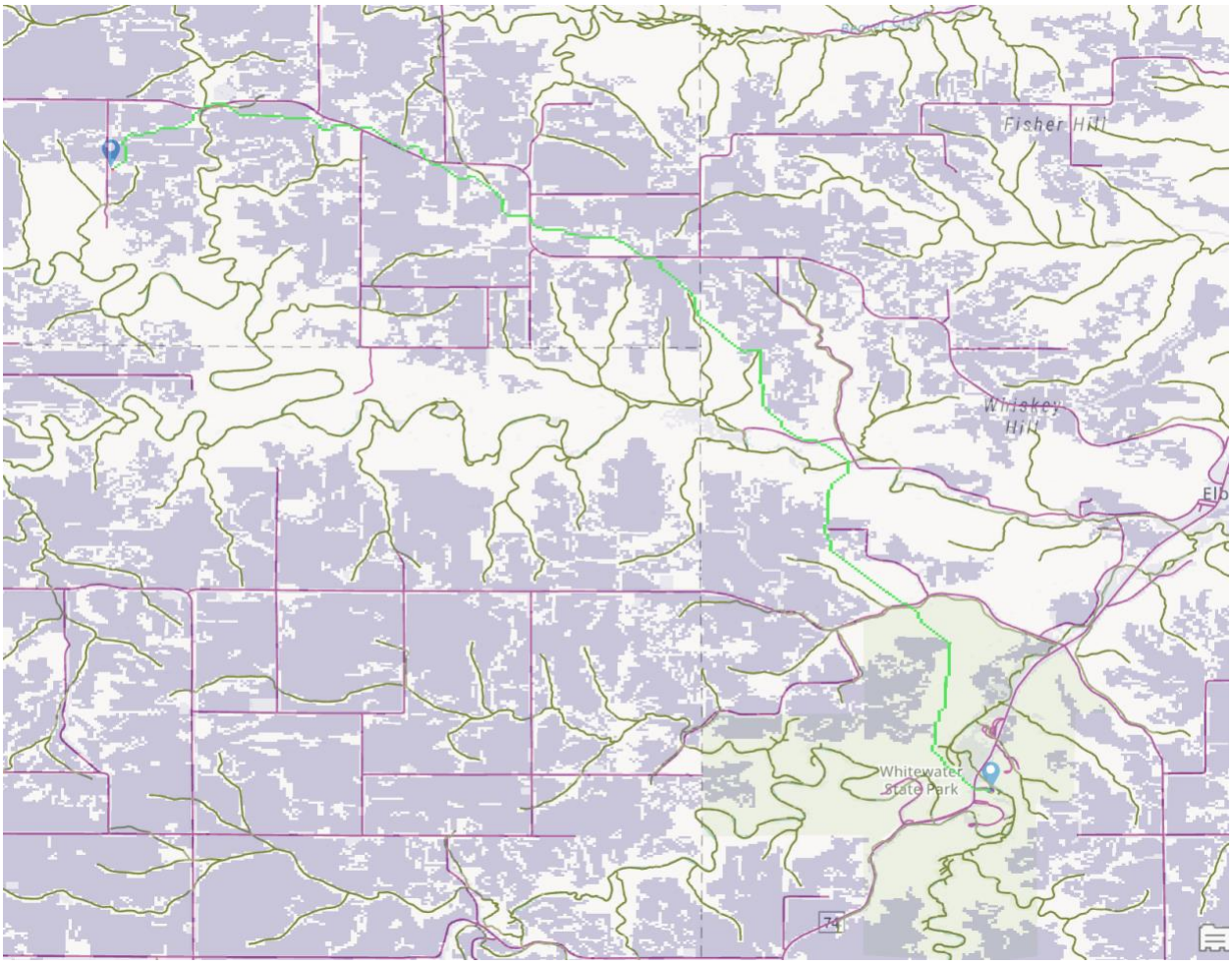


Figure 16: *Dory's optimal path (shown as the bright green line), on top of the cost surface raster at 50% transparency. Rivers and roadways are shown here.*

As shown in the above **Figure 16.**, I was able to achieve a least cost path for Dory, using the weighted parameters described in the Methods. Looking at **Figure 16**, it appears that Dory's path stays along low-cost areas (shown in light/beige), and she only crosses rivers (dark yellow) where there are roads overpassing them (shown in pink). From this qualitative analysis, it appears that this cost path is correct, based on the parameters I inputted. However, some subjective decisions we made during the calculation of this cost path that could have affected the path's accuracy. The elevation DEM rasters were downsampled quite a bit to reduce file size (though I believe the decision could be justified, as all data used in a calculation can only be as accurate as the least accurate component). I believe buffering the roadways by 100m to erase the intersecting river sections may have been an overexaggeration, though I do not believe a smaller buffer size would have substantially changed the trajectory of the path. Also, the decision to weight land cover as twice as important to Dory than slope was subjective, and my results may have greatly differed without that weighted calculation.

Discussion and Conclusion

This report demonstrates how an ETL methodology which operates in ArcPro Jupyter Notebooks that inputs a variety of different types of data, converts the data into rasters, reclassifies the rasters to be on the same classification scale, and calculates a weighted cost surface raster which reflects all of the different data components (and their relative costs). This report successfully demonstrates how to calculate a least cost path, via generating a cost distance raster and a cost back-link raster from a weighted cost surface raster.

References

(See links to datasets provided in Table 2)

Self-score

Category	Description	Points Possible	Score
Structural Elements	All elements of a lab report are included **(2 points each)**: Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score	28	
Clarity of Content	Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5 minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**.	24	
Reproducibility	Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified.	28	
Verification	Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**.	20	
		100	