

# CSCI 3412 Algorithms

## Homework 4

Matt Sullivan

### Part 1: Program Description

In problem 4, we are trying to demonstrate how a greedy algorithm to find a route between stars in a radius surrounding earth is an effective way of determining a “good” (not necessarily perfect) route between the stars. To do this, we are going to take an online database of stars (the HYG or Hippocarcos, Yale, Gliese database), which is in CSV format. We will read in this data and design a traversal algorithm using the greedy principle of using locally available data to find the optimal solution in that case.

### Part 2: Algorithm Design

I wrote my code in Python. I imported the math library for the sqrt function, as well as matplotlib for practice in making graphs for my data.

The first thing I did is define class Star, which includes ID number, distance from earth, x, y, and z coordinates, 5 different name values (to account for having different names), then I included a distance traveled (from last star) and total distance traveled (entire journey) for ease of keeping track and printing (I initialized both to 0). I used a series of if/else statements to name the star (prioritizing proper name, then bayer flamsteed, then gliese, then harvard revised, then henry draper). If a star had no proper name, I set the name to “Unnamed star X” with X being the ID number of the star. I set the star to return its name if it needed to be referenced or printed, and set the less than value based on the distance of the star from earth. This made it simple to set the origin point later in the code, by finding the smallest value of the array.

I then defined readFromFile, which was a custom built csv parser that I build prior to Dr. Williams suggesting we find one online. It's very simple, just ignores the first line, then only retrieves the information in the fields I want, makes a star object using the data, then appends the star to a star list.

I then demonstrated how many stars were in the list total, how many were in 100 parsecs, and how many were within 10.

GreedyStarList was a simple algorithm with an input of an origin point and an array of stars. It calculated the distance using the 3d pythagorean theorem on the difference between the x, y, and z coordinates of the origin point and each other star in the array. If the distance was smaller than the min (which was originally set to a distance much greater than 20 parsecs), the distance became the new min. After it steps through the entire array, it deletes the target from the list (ensuring we only visit each star once), updates the target's distance traveled and current total distance, and then returns the target. In later versions, I added an iteration count, which is also returned with the target. This enables me to show how many times the function is called.

My final function was calculateStarRoute, which took an array of stars and printed out the route, using greedyStarRoute to create a list of stars in the order they were visited, then printing out the route by stepping through this list. I also incremented the iteration count each time, letting me record the total number of iterations in the program. Finally, I created a plot using matplotlib showing the route taking in 3d, though I didn't have the time to figure out how to animate it star by star. (I did manage to get it to spin though, first time creating a plot in python)

Code w/comments posted below:

```
# -*- coding: utf-8 -*-  
"""
```

Created on Tue Apr 17 20:47:26 2018

@author: oneey\_000

By Matt Sullivan

```
"""
```

```
import math          # used for sqrt function  
import matplotlib.pyplot as plt  
from mpl_toolkits.mplot3d import Axes3D  
from matplotlib import animation
```

```
class star:
```

```
    def __init__(self, idnum, HD, HR, gliese, BayerFlamsteed, proper, dist, xcoord, ycoord, zcoord):  
        self.idnum= int(idnum)  
        self.dist = float(dist)  
        self.xcoord = float(xcoord)  
        self.ycoord = float(ycoord)  
        self.zcoord = float(zcoord)  
        self.HD = str(HD)  
        self.HR = str(HR)  
        self.gliese = str(gliese)  
        self.BayerFlamsteed = str(BayerFlamsteed)  
        self.proper = str(proper)  
        self.distanceTravelled=0  
        self.totalDistanceTravelled=0  
        if self.proper:          #Sets name of star, prioritizing Proper name, BayerFlamsteed,  
Gliese, HR, HD, or generated name  
            self.name=proper  
        elif self.BayerFlamsteed:  
            self.name = str(BayerFlamsteed)  
        elif self.gliese:  
            self.name = "Gliese " + str(gliese)  
        elif self.HR:  
            self.name = "Harvard Revised " + str(HR)  
        elif self.HD:  
            self.name = "Henry Draper " + str(HD)  
        else:  
            self.name = "Unnamed Star " + str(idnum)  
  
    def __repr__(self):          #Sets the return value of the star object  
        return (self.name)  
    def __str__(self):          #Sets the return value of the object in print statements  
        return (self.name)  
    def __lt__(self,other):      #Sets the less than value of the star object  
        return (self.dist<other.dist)
```

```

def readFromFile(readFile):    #CSV parser I wrote for to read stars from file
    starList=[]
    myFile = open(readFile, "r")
    next(myFile)              #Skips first line
    for line in myFile:
        workLine=line.split(",")
        newStar=star(workLine[0], workLine[2], workLine[3], workLine[4], workLine[5], workLine[6],
workLine[9], workLine[17], workLine[18],workLine[19])
        starList.append(newStar)
    return starList

totalStarList = readFromFile("hygxyz.csv")
medStarList=[]
smallStarList=[]

print("Total Star List Size: " + str(len(totalStarList)))

for entry in totalStarList:
    if entry.dist <=100:
        medStarList.append(entry)

print("# Stars within 100 parsecs: " + str(len(medStarList)))

for entry in medStarList:
    if entry.dist <=10:
        smallStarList.append(entry)

print("# Stars within 10 parsecs: " + str(len(smallStarList)))

def greedyStarRoute(origin, starList):    #Program to select nearest star
    iteration=0
    currentTotalDistance=10000000        #Arbitarily large number set as default
    for destination in starList:          #Steps through list
        iteration+=1
        xDist=(destination.xcoord - origin.xcoord)**2 #Gets difference of x values squared
        yDist=(destination.ycoord - origin.ycoord)**2 #gets difference of y values squared
        zDist=(destination.zcoord - origin.zcoord)**2 #gets difference of z values squared
        totalDist = math.sqrt(xDist + yDist + zDist)  #Gets the square root of all of those values added
        if totalDist < currentTotalDistance:          #if it's smaller than the smallest distance
            currentTotalDistance=totalDist            #set the current distance to the smallest distance
            target=destination                        #the target is the destination
            starList.remove(target)                   #Delete the closest target from the list
            target.distanceTravelled = currentTotalDistance #Set the target's distance travelled variable to the
current leg
            target.totalDistanceTravelled = currentTotalDistance+origin.totalDistanceTravelled #Update the
total distance travelled
    return target, iteration                        #return the target

def calculateStarRoute(starList):

```

```

origin = min(starList)          #Sets the origin
routeInOrder=[]
totalDistance = 0
totalIt=0
while len(starList)>0:          #While there's still stars left
    nextStop, numIt = greedyStarRoute(origin, starList) #Next stop is the target of greedyStarRoute
    routeInOrder.append(nextStop)          #Append the stop to the route
    origin = nextStop                      #Sets the next stop to the origin value of greedyStarRoute
    totalDistance += nextStop.distanceTravelled
    totalIt+=numIt

print(len(routeInOrder))

for index, entry in enumerate(routeInOrder):
    if index < len(routeInOrder)-1:
        print(entry.name + " -> " + routeInOrder[index+1].name + " Distance: " + str('%s' %
float('%0.3g' % routeInOrder[index+1].distanceTravelled)) + " Total Distance Travelled: " + str('%s' %
float('%0.5g' % routeInOrder[index+1].totalDistanceTravelled)))

print("Total Distance Travelled: " + str('%s' % float('%0.5g' % totalDistance)))
print("Number of iterations Greedy: " + str(totalIt))

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
xline=[]
yline=[]
zline=[]
for element in routeInOrder:
    xline.append(element.xcoord)
    yline.append(element.ycoord)
    zline.append(element.zcoord)
ax.plot(xline,yline,zline)
plt.show()

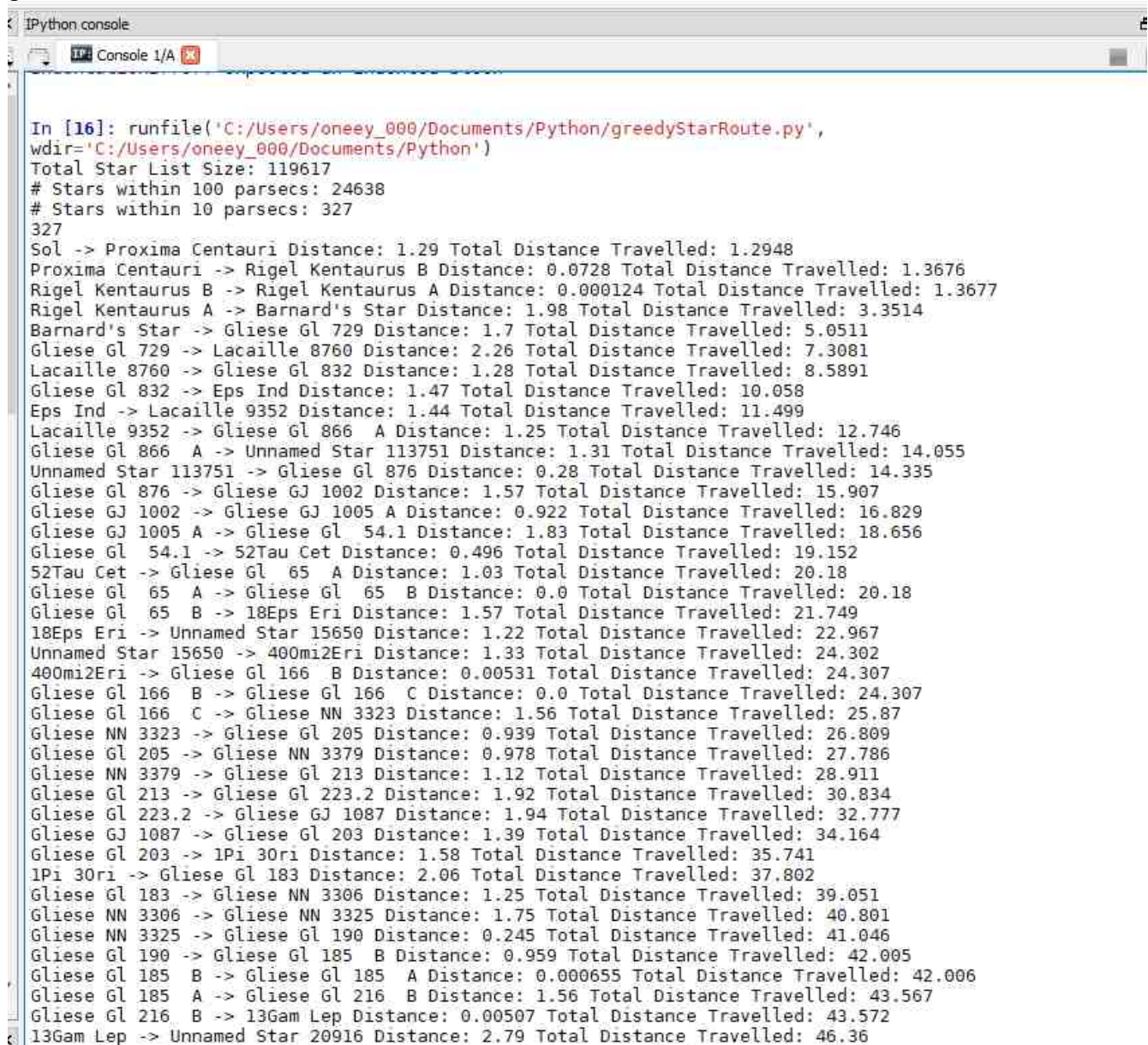
# for angle in range(0, 360):    #Rotating plot of star route, uncomment to enable
#     ax.view_init(30, angle)
#     plt.draw()
#     plt.pause(.001)

calculateStarRoute(smallStarList)

```

## Part 3 Sample Runs

Here is a picture of the output of my code. The code matches the example output in the handout (I made the output of my code match the sample for ease of comparison). I've also included the figure I generated at the end.



```
In [16]: runfile('C:/Users/oneey_000/Documents/Python/greedyStarRoute.py',
wdir='C:/Users/oneey_000/Documents/Python')
Total Star List Size: 119617
# Stars within 100 parsecs: 24638
# Stars within 10 parsecs: 327
327
Sol -> Proxima Centauri Distance: 1.29 Total Distance Travelled: 1.2948
Proxima Centauri -> Rigel Kentaurus B Distance: 0.0728 Total Distance Travelled: 1.3676
Rigel Kentaurus B -> Rigel Kentaurus A Distance: 0.000124 Total Distance Travelled: 1.3677
Rigel Kentaurus A -> Barnard's Star Distance: 1.98 Total Distance Travelled: 3.3514
Barnard's Star -> Gliese Gl 729 Distance: 1.7 Total Distance Travelled: 5.0511
Gliese Gl 729 -> Lacaille 8760 Distance: 2.26 Total Distance Travelled: 7.3081
Lacaille 8760 -> Gliese Gl 832 Distance: 1.28 Total Distance Travelled: 8.5891
Gliese Gl 832 -> Eps Ind Distance: 1.47 Total Distance Travelled: 10.058
Eps Ind -> Lacaille 9352 Distance: 1.44 Total Distance Travelled: 11.499
Lacaille 9352 -> Gliese Gl 866 A Distance: 1.25 Total Distance Travelled: 12.746
Gliese Gl 866 A -> Unnamed Star 113751 Distance: 1.31 Total Distance Travelled: 14.055
Unnamed Star 113751 -> Gliese Gl 876 Distance: 0.28 Total Distance Travelled: 14.335
Gliese Gl 876 -> Gliese GJ 1002 Distance: 1.57 Total Distance Travelled: 15.907
Gliese GJ 1002 -> Gliese GJ 1005 A Distance: 0.922 Total Distance Travelled: 16.829
Gliese GJ 1005 A -> Gliese Gl 54.1 Distance: 1.83 Total Distance Travelled: 18.656
Gliese Gl 54.1 -> 52Tau Cet Distance: 0.496 Total Distance Travelled: 19.152
52Tau Cet -> Gliese Gl 65 A Distance: 1.03 Total Distance Travelled: 20.18
Gliese Gl 65 A -> Gliese Gl 65 B Distance: 0.0 Total Distance Travelled: 20.18
Gliese Gl 65 B -> 18Eps Eri Distance: 1.57 Total Distance Travelled: 21.749
18Eps Eri -> Unnamed Star 15650 Distance: 1.22 Total Distance Travelled: 22.967
Unnamed Star 15650 -> 400mi2Eri Distance: 1.33 Total Distance Travelled: 24.302
400mi2Eri -> Gliese Gl 166 B Distance: 0.00531 Total Distance Travelled: 24.307
Gliese Gl 166 B -> Gliese Gl 166 C Distance: 0.0 Total Distance Travelled: 24.307
Gliese Gl 166 C -> Gliese NN 3323 Distance: 1.56 Total Distance Travelled: 25.87
Gliese NN 3323 -> Gliese Gl 205 Distance: 0.939 Total Distance Travelled: 26.809
Gliese Gl 205 -> Gliese NN 3379 Distance: 0.978 Total Distance Travelled: 27.786
Gliese NN 3379 -> Gliese Gl 213 Distance: 1.12 Total Distance Travelled: 28.911
Gliese Gl 213 -> Gliese Gl 223.2 Distance: 1.92 Total Distance Travelled: 30.834
Gliese Gl 223.2 -> Gliese GJ 1087 Distance: 1.94 Total Distance Travelled: 32.777
Gliese GJ 1087 -> Gliese Gl 203 Distance: 1.39 Total Distance Travelled: 34.164
Gliese Gl 203 -> 1Pi 30ri Distance: 1.58 Total Distance Travelled: 35.741
1Pi 30ri -> Gliese Gl 183 Distance: 2.06 Total Distance Travelled: 37.802
Gliese Gl 183 -> Gliese NN 3306 Distance: 1.25 Total Distance Travelled: 39.051
Gliese NN 3306 -> Gliese NN 3325 Distance: 1.75 Total Distance Travelled: 40.801
Gliese NN 3325 -> Gliese Gl 190 Distance: 0.245 Total Distance Travelled: 41.046
Gliese Gl 190 -> Gliese Gl 185 B Distance: 0.959 Total Distance Travelled: 42.005
Gliese Gl 185 B -> Gliese Gl 185 A Distance: 0.000655 Total Distance Travelled: 42.006
Gliese Gl 185 A -> Gliese Gl 216 B Distance: 1.56 Total Distance Travelled: 43.567
Gliese Gl 216 B -> 13Gam Lep Distance: 0.00507 Total Distance Travelled: 43.572
13Gam Lep -> Unnamed Star 20916 Distance: 2.79 Total Distance Travelled: 46.36
```

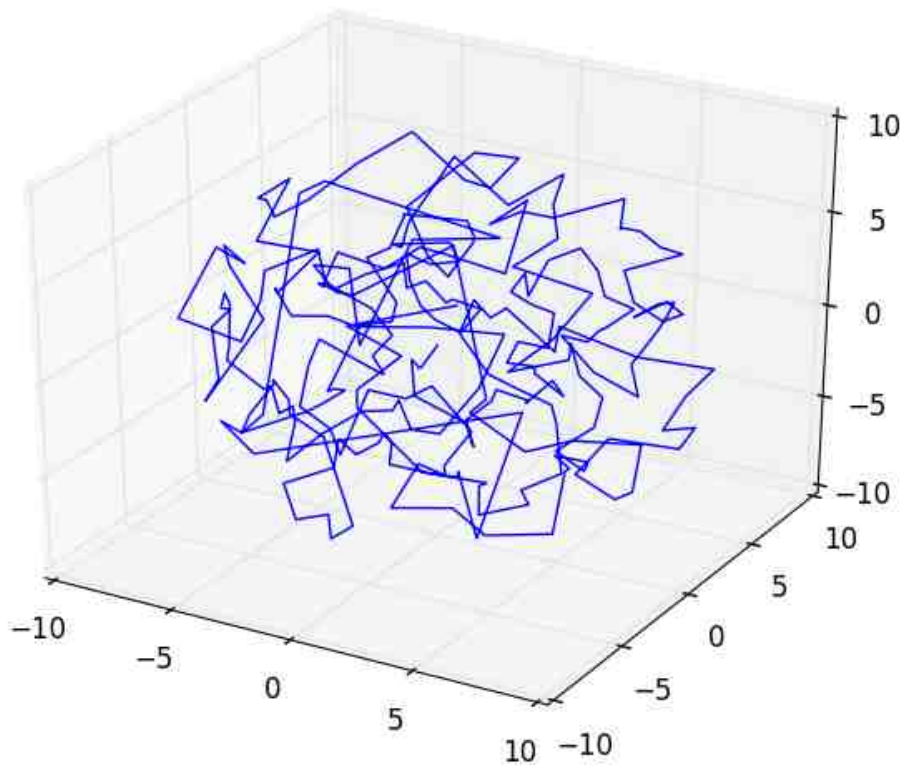
```

Van Maanen's Star -> Gliese Gl 83.1 Distance: 1.47 Total Distance Travelled: 421.95
Gliese Gl 83.1 -> 96 G. Psc Distance: 3.57 Total Distance Travelled: 425.52
96 G. Psc -> Gliese NN 3119 Distance: 3.86 Total Distance Travelled: 429.37
Gliese NN 3119 -> Gliese Gl 84 Distance: 2.11 Total Distance Travelled: 431.48
Gliese Gl 84 -> Gliese NN 3135 Distance: 2.07 Total Distance Travelled: 433.55
Gliese NN 3135 -> Gliese GJ 1028 Distance: 3.15 Total Distance Travelled: 436.7
Gliese GJ 1028 -> Gliese GJ 2012 Distance: 1.2 Total Distance Travelled: 437.9
Gliese GJ 2012 -> Gliese GJ 2005 Distance: 2.56 Total Distance Travelled: 440.47
Gliese GJ 2005 -> Gliese Gl 915 Distance: 2.23 Total Distance Travelled: 442.7
Gliese Gl 915 -> Gliese GJ 1001 Distance: 1.84 Total Distance Travelled: 444.54
Gliese GJ 1001 -> Gliese NN 3049 Distance: 1.29 Total Distance Travelled: 445.82
Gliese NN 3049 -> Gliese NN 4360 Distance: 4.66 Total Distance Travelled: 450.49
Gliese NN 4360 -> Gliese Gl 849 Distance: 4.04 Total Distance Travelled: 454.53
Gliese Gl 849 -> Gliese Gl 831 A Distance: 1.75 Total Distance Travelled: 456.28
Gliese Gl 831 A -> Gliese Gl 831 B Distance: 0.00174 Total Distance Travelled: 456.28
Gliese Gl 831 B -> Gliese Gl 791.2 Distance: 3.67 Total Distance Travelled: 459.95
Gliese Gl 791.2 -> Gliese GJ 1256 Distance: 1.5 Total Distance Travelled: 461.45
Gliese GJ 1256 -> Gliese GJ 1253 Distance: 7.06 Total Distance Travelled: 468.51
Gliese GJ 1253 -> Gliese Gl 661 B Distance: 5.12 Total Distance Travelled: 473.63
Gliese Gl 661 B -> Gliese Gl 661 A Distance: 0.00195 Total Distance Travelled: 473.63
Gliese Gl 661 A -> Gliese NN 3991 Distance: 0.963 Total Distance Travelled: 474.59
Gliese NN 3991 -> Gliese Gl 623 Distance: 1.45 Total Distance Travelled: 476.04
Gliese Gl 623 -> Gliese Gl 625 Distance: 1.64 Total Distance Travelled: 477.69
Gliese Gl 625 -> Unnamed Star 85346 Distance: 3.54 Total Distance Travelled: 481.23
Unnamed Star 85346 -> Gliese Gl 445 Distance: 5.56 Total Distance Travelled: 486.79
Gliese Gl 445 -> Gliese Gl 169.1B Distance: 3.39 Total Distance Travelled: 490.18
Gliese Gl 169.1B -> Gliese Gl 169.1A Distance: 0.00299 Total Distance Travelled: 490.18
Gliese Gl 169.1A -> Gliese Wo 9492 Distance: 7.92 Total Distance Travelled: 498.1
Gliese Wo 9492 -> Gliese NN 3855 Distance: 1.12 Total Distance Travelled: 499.22
Gliese NN 3855 -> Gliese Gl 678.1A Distance: 10.4 Total Distance Travelled: 509.64
Gliese Gl 678.1A -> Gliese GJ 1207 Distance: 2.24 Total Distance Travelled: 511.88
Gliese GJ 1207 -> 12 Oph Distance: 0.958 Total Distance Travelled: 512.84
12 Oph -> Unnamed Star 84326 Distance: 2.0 Total Distance Travelled: 514.83
Unnamed Star 84326 -> Gliese Gl 1 Distance: 11.1 Total Distance Travelled: 525.98
Gliese Gl 1 -> Gam Pav Distance: 6.26 Total Distance Travelled: 532.24
Gam Pav -> Unnamed Star 31220 Distance: 5.55 Total Distance Travelled: 537.78
Unnamed Star 31220 -> Unnamed Star 31215 Distance: 0.393 Total Distance Travelled: 538.18
Unnamed Star 31215 -> Gliese GJ 1123 Distance: 1.52 Total Distance Travelled: 539.7
Gliese GJ 1123 -> Gliese Gl 367 Distance: 5.23 Total Distance Travelled: 544.93
Gliese Gl 367 -> Gliese Gl 358 Distance: 0.868 Total Distance Travelled: 545.79
Gliese Gl 358 -> Gliese Gl 318 Distance: 2.34 Total Distance Travelled: 548.13
Gliese Gl 318 -> Gliese Gl 357 Distance: 2.57 Total Distance Travelled: 550.7
Gliese Gl 357 -> Gliese Gl 283 A Distance: 4.26 Total Distance Travelled: 554.97
Gliese Gl 283 A -> Gliese Gl 283 B Distance: 0.0 Total Distance Travelled: 554.97
Total Distance Travelled: 554.97
Number of iterations Greedy: 53628
C:\Anaconda3\lib\site-packages\matplotlib\backend_bases.py:2437: MatplotlibDeprecationWarning: Using
default event loop until function specific to this GUI is implemented
warnings.warn(str, mplDeprecation)

```

In [17]:





#### Part 4: Analysis

This greedy algorithm was fairly simple to write, and is much more efficient than the “optimal” solution. This is a variation of the traveling salesman problem, a well known problem belonging to the NP complete class. It is  $O(n!)$ . In contrast, our solution, using only the local variables known, comes to a  $O\left(\frac{x^2+x}{2}\right)$ , since the size of the starList shrinks by one with each iteration. This O value of course simplifies to  $O(x^2)$ , since the  $x^2$  overwhelms the  $x$  and the  $.5$  multiplier. This number of iterations is borne out in our iterations tracker which outputs 53628 iterations. I was unable to write a  $O(n!)$  version (using a much smaller sample size, of course), but the searching I've found on the internet estimates the value of  $327!$  to be  $7.93 \times 10^{681}$ , showing that our solution is **much** more efficient to compute (by many orders of magnitude), if not the ABSOLUTE optimal solution.

#### Part 5: Conclusion

In this exercise, we demonstrated how a greedy algorithm, while not finding the absolute best solution, found an acceptable solution in a much more effective and efficient amount of time using local information than the complete optimal solution could. While optimal solutions are nice, I doubt I could wait  $2 \times 10^{663}$  years for one (at 1 million iterations per second on my current machine). I mean, I could complete the route in 3618 years at 0.5 light speed, which is far faster than figuring out the route.

## Part 6: Reference

[https://en.wikipedia.org/wiki/Travelling\\_salesman\\_problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem) (where  $O(n!)$  information came from, apart from lectures)

<https://www.calculatorsoup.com/calculators/discretemathematics/factorials.php> (estimated  $327!$  value)

<https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/> (Tutorial for animating python chart)

<https://matplotlib.org/index.html> (All info about matplotlib)

[https://matplotlib.org/gallery/mplot3d/rotate\\_axes3d.html](https://matplotlib.org/gallery/mplot3d/rotate_axes3d.html) (how to rotate a python plot)