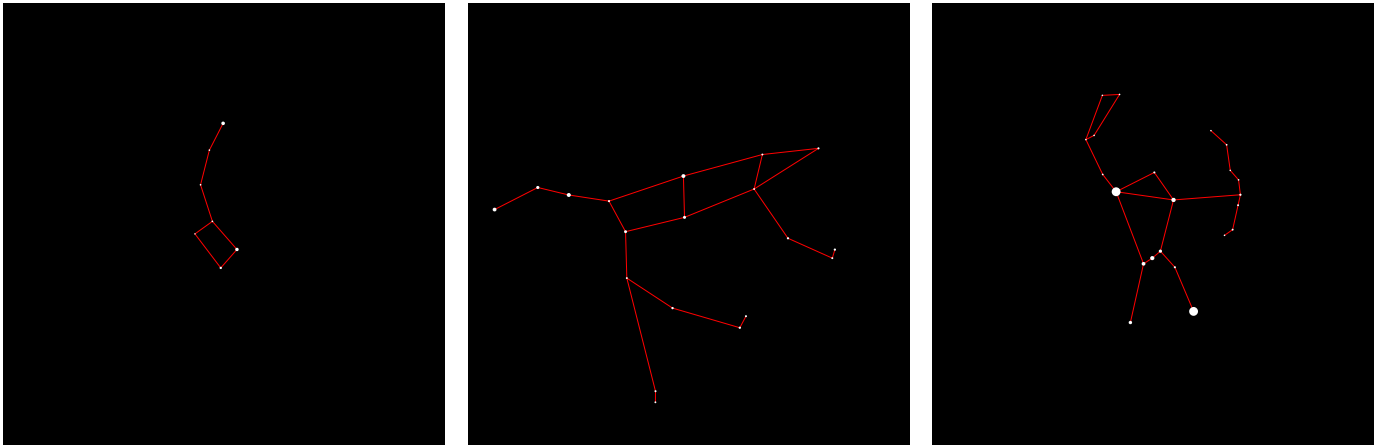


CSCI-1200 Data Structures — Fall 2019

Homework 10 — Constellation Hash Table

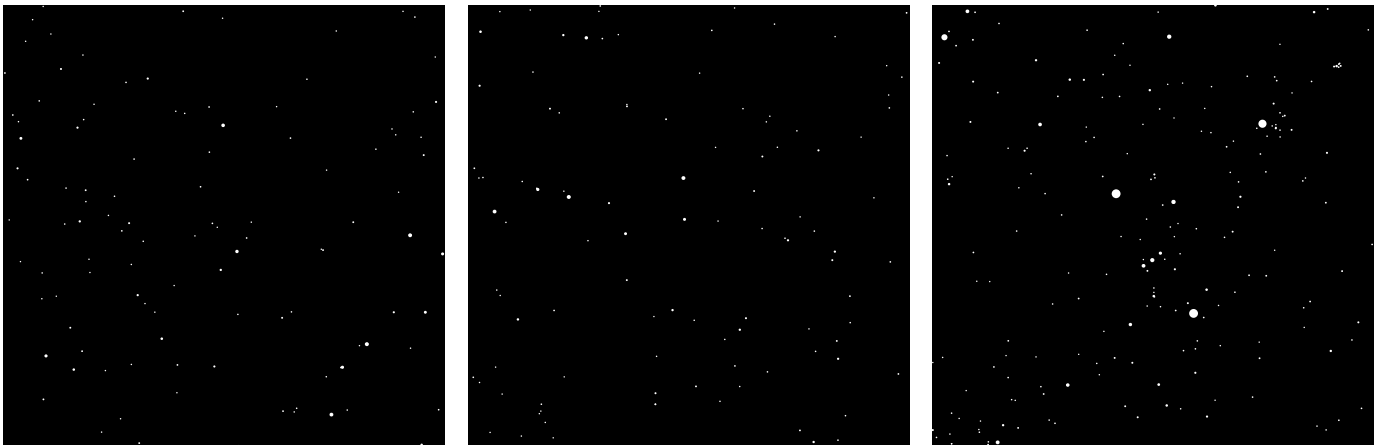
In this assignment we will search the stars of the night sky for known constellations. This is a challenging problem and you are welcome to use any data structures, algorithms, and techniques we have learned this semester. We hope that you experiment with using *hash tables* as one tool for this problem.

Below are three of the 88 modern constellations recognized by the International Astronomical Union (IAU). From left to right they are *Ursa Minor* (also known as the *Little Dipper*), *Ursa Major* (a subset of this constellation, an *asterism*, is also popularly known as the *Big Dipper*), and *Orion*. For clarity in these initial diagrams we only draw the brightest stars that are used to form the connect-the-dots constellation diagrams. But the night sky contains many, many more stars.



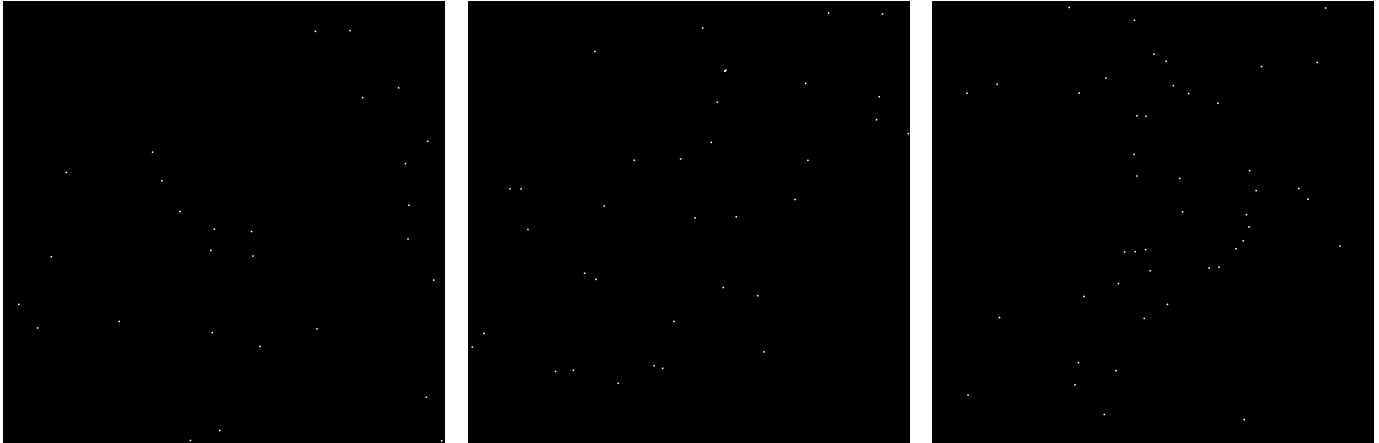
The stars are mapped to a sphere surrounding the Earth, and every star has a unique name and belongs to exactly one of these 88 constellations (though not all stars in the constellation are used to connect the dots for the diagram). Your task for this assignment is to write a program that will take a synthetic image of the night sky as input, along with a library of known constellations, and identify which portion of the night sky we are observing. Once identified, we can draw the constellation diagram on the image and label the visible stars by name.

In the images below we visualize the same three constellations surrounded by many of their neighboring stars. Can you connect the dots to trace the same diagrams? Note that images of the night sky are challenging to visualize because of the high dynamic range from the inky black sky and the large range of dim and bright stars. We approximate this imagery by drawing the brighter stars as disks with larger radii.



Finding constellations can be especially challenging because the orientation (position & rotation) of the constellation changes with the observer's position on the Earth and the season. Furthermore, if we used actual photographs of the night sky, the stars might be over- or under- exposed causing us to miss some stars completely and making measurements of relative star brightness unreliable.

The images below show some of the brightest stars from the previous visualizations, but all of these stars are displayed with the same radius/brightness. Additionally, the images have been rotated. Can you still find the constellations in their rotated orientations and without relying on hints from the relative star brightness?



The three stars of Orion's belt are not as obvious when they are equal brightness to their neighbors! But we can still find them by matching distances and angles between neighboring stars.

Program Input and Output & Command Line Arguments

Your program will take in an input file with unidentified star data – an example is shown below on the left. Each row lists a star's x,y position within the synthetic image, the *apparent magnitude*, and the *star name*. The apparent magnitude is the star's brightness as observed from Earth – note that smaller numbers are brighter! In our input files, a value of -1 indicates that this data is unknown or unreliable. Star names beginning with STAR_ are unknown and your task is to identify them.

example_input.txt

```

star 784.74 66.13 -1.00 STAR_00001
star 34.89 685.95 -1.00 STAR_00002
star 813.38 217.91 -1.00 STAR_00003
star 108.37 578.21 -1.00 STAR_00004
star 957.61 896.12 -1.00 STAR_00005
star 489.87 971.57 -1.00 STAR_00006
star 895.22 195.74 -1.00 STAR_00007
star 562.02 521.11 -1.00 STAR_00008
star 142.30 387.11 -1.00 STAR_00009
star 423.50 994.14 -1.00 STAR_00010
star 961.07 316.81 -1.00 STAR_00011
star 565.64 576.59 -1.00 STAR_00012
star 992.62 995.02 -1.00 STAR_00013
star 262.11 724.67 -1.00 STAR_00014
star 910.54 367.30 -1.00 STAR_00015
star 358.82 406.26 -1.00 STAR_00016
star 918.46 461.54 -1.00 STAR_00017
star 400.15 475.86 -1.00 STAR_00018
star 473.01 749.98 -1.00 STAR_00019
star 477.69 515.45 -1.00 STAR_00020
star 77.51 739.39 -1.00 STAR_00021
star 581.32 781.21 -1.00 STAR_00022
star 974.54 631.06 -1.00 STAR_00023
star 469.79 563.48 -1.00 STAR_00024
star 710.01 741.47 -1.00 STAR_00025
star 916.35 537.62 -1.00 STAR_00026
star 706.76 67.74 -1.00 STAR_00027
star 337.89 341.30 -1.00 STAR_00028

```

Your program will also take as input a library of one or more known constellations; an example is shown below right. In addition to the star data, the file also includes a label for the constellation name, and the line segments between stars that make the classic constellation diagram. Note that the star names include extended ASCII characters from the Greek alphabet. The STL string library should handle this for you.

Ursa_Minor.txt

```

constellation Ursa_Minor
star 497.59 273.15 1.97 Polaris
star 528.93 558.78 2.07 β_UMi
star 492.26 600.57 3.04 γ_UMi
star 446.47 412.32 4.21 ε_UMi
star 473.30 495.15 4.29 ζ_UMi
star 466.45 333.89 4.35 δ_UMi
star 433.75 523.53 4.95 η_UMi
line Polaris δ_UMi
line β_UMi γ_UMi
line β_UMi ζ_UMi
line γ_UMi η_UMi
line δ_UMi ε_UMi
line ε_UMi ζ_UMi
line ζ_UMi η_UMi

```

Your program should search for potential matches between the input file and library of constellations. If you cannot find a match, your program should print “No matching constellation found” to `std::cout`. But when you do identify a matching constellation, you should print basic data on the match to `std::cout` as shown below.

```

example_stdout.txt
FOUND CONSTELLATION Ursa_Minor
assigning STAR_00008 ̢_UMi
assigning STAR_00012 ̣_UMi
assigning STAR_00016 ̤_UMi
assigning STAR_00018 ̥_UMi
assigning STAR_00020 ̦_UMi
assigning STAR_00024 ̧_UMi
assigning STAR_00028 Polaris

```

Your program will also revise the input file to label each identified star with their official name, update the apparent magnitudes, and add the line segment information. The updated file is saved to the specified output file name, which is shown on the right.

We provide a simple stand-alone program that will read the star positions and line segments from a file (either the input or output version) and produce a Scalable Vector Graphics (SVG) visualization stored as an `.html` file. You can open this file locally in a standard web browser (e.g., Chrome, Firefox, etc.)

Here are some sample command lines:

```

./stars.out -i example_input.txt -o example_output.txt -l Ursa_Minor.txt > example_stdout.txt
./plot.out example_output.txt example.html
./stars.out -i unknown_A.txt -o out.txt -l Ursa_Minor.txt -l Ursa_Major.txt -l Orion.txt

```

example_output.txt

```

constellation Ursa_Minor
star 784.74 66.13 -1.00 STAR_00001
star 34.89 685.95 -1.00 STAR_00002
star 813.38 217.91 -1.00 STAR_00003
star 108.37 578.21 -1.00 STAR_00004
star 957.61 896.12 -1.00 STAR_00005
star 489.87 971.57 -1.00 STAR_00006
star 895.22 195.74 -1.00 STAR_00007
star 562.02 521.11 2.07 ̢_UMi
star 142.30 387.11 -1.00 STAR_00009
star 423.50 994.14 -1.00 STAR_00010
star 961.07 316.81 -1.00 STAR_00011
star 565.64 576.59 3.04 ̣_UMi
star 992.62 995.02 -1.00 STAR_00013
star 262.11 724.67 -1.00 STAR_00014
star 910.54 367.30 -1.00 STAR_00015
star 358.82 406.26 4.35 ̤_UMi
star 918.46 461.54 -1.00 STAR_00017
star 400.15 475.86 4.21 ̥_UMi
star 473.01 749.98 -1.00 STAR_00019
star 477.69 515.45 4.29 ̦_UMi
star 77.51 739.39 -1.00 STAR_00021
star 581.32 781.21 -1.00 STAR_00022
star 974.54 631.06 -1.00 STAR_00023
star 469.79 563.48 4.95 ̧_UMi
star 710.01 741.47 -1.00 STAR_00025
star 916.35 537.62 -1.00 STAR_00026
star 706.76 67.74 -1.00 STAR_00027
star 337.89 341.30 1.97 Polaris
line Polaris ̤_UMi
line ̢_UMi ̣_UMi
line ̢_UMi ̦_UMi
line ̣_UMi ̧_UMi
line ̤_UMi ̥_UMi
line ̥_UMi ̦_UMi
line ̦_UMi ̧_UMi

```

Getting Started

We suggest you start this homework by prototyping a brute-force search solution for some of the smaller example files. You may find that recursion is helpful! The idea is to start with a pair of stars in the input file, compute the distance between those points, and then see if you can find a matching pair of stars in one of the constellation library files that has the same distance. Note that when comparing floating point numbers (either distances or angles) you don’t want to use exact equality. You’ll want to compare if the difference between the numbers is less than a small *epsilon* threshold value. Can you then expand that initial match to the whole constellation?

Once you have a basic version of the program working, evaluate what part of the search is most expensive. What data structures or algorithms can you use to improve this search? Can you use a hash table? What is the information you will be hashing? Should you consider using a spatial data structure to accelerate nearest neighbor queries? A grid is probably simpler than an octree or bounding volume hierarchy.

In addition to comparing distances, we can compare the angles formed between three neighboring stars. For extra credit extend your program to handle input images that might be scaled (a.k.a. zoomed), where the matched constellation is either smaller or larger than the example in the library.

Homework Submission

Use good coding style and detailed comments when you design and implement your program. You must do this assignment on your own, as described in the “[Collaboration Policy & Academic Integrity](#)” handout. If you did discuss this assignment, problem solving techniques, or error messages, etc. with anyone, please list their names in your `README.txt` file.