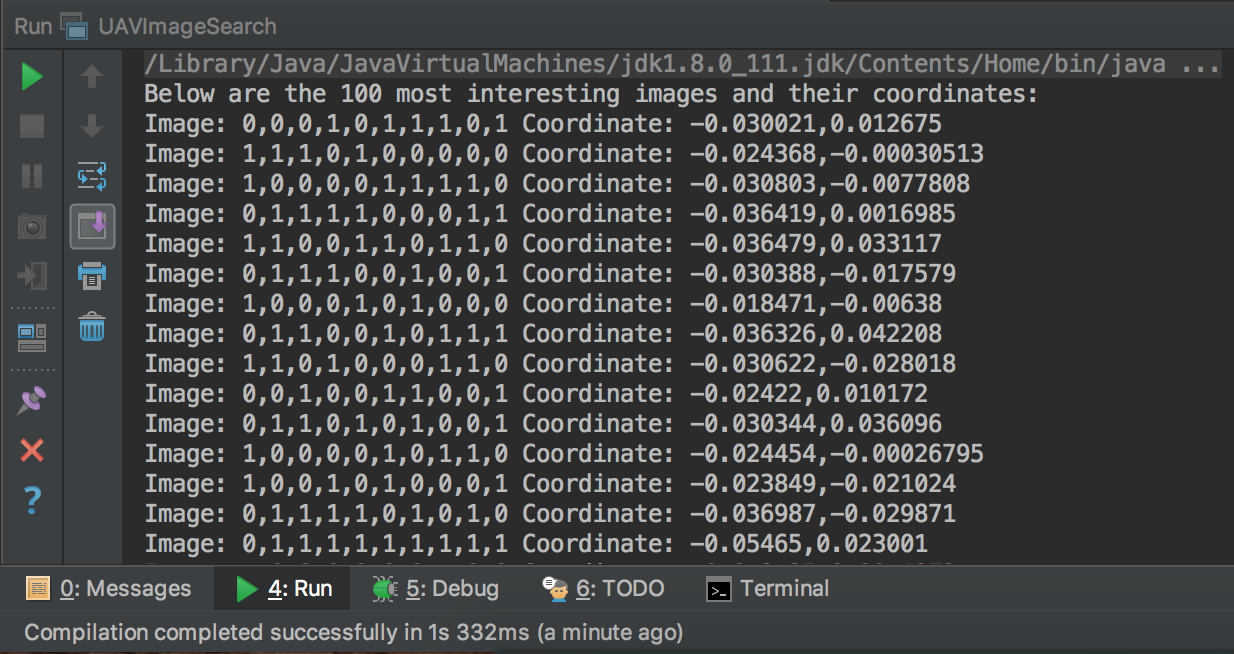
CMPE 365 Algorithms Lab 3

Robert Saunders

10194030

Please evaluate and run the code for the lab, feel free to choose a number of interesting images you would like the program to return. The program will run an algorithm to find the most interesting images from a set of images, where an image is a 10-bit string where each bit indicates some sort of information. Before diving into how the algorithm works in my implementation the output you should see when running the program is:



Upon graphing the 2-dimensional coordinates of the interesting images we get a distribution such as below:

As you can see the images plotted appear to be reasonably spread out, but there is definitely room for improve as bunching occurs in bands on the y-axis. Discrepancies of the interesting set is further explained below. The graph was obtained by taking the interesting image coordinate output and graphing it in Excel.

The algorithm first divides the original image set into groups, specifically the number of images in the original dataset divided by the number of images required also divided by 2, so if 1000 images are provided and we wish to have 100 interesting images there would be 20 groups of images each with 50 elements in them. The algorithm then compares all elements in every group to find the two images with the greatest hamming distance, those two images are considered to be the most interesting from that group and saved to the list of interesting images, this process repeats until all groups have found their two interesting images. This explains why we divide the number of images required by two at the beginning because each group has two interesting images, not one. In the process, we save the indexes of those interesting images from the original data so we can retrieve their corresponding coordinates later. The idea of segmenting the images into smaller sets and finding the most interesting from that set will provide more coverage of the dataset, this is why it is a successful approach, that being said there is quite a lot of improvements to be made, see below:

Improvements to be made:

* Make group creation less linear, this will significantly improve group comparison discrepancies, could take elements from front and back original data set.
* If the hamming distance is at its max when comparing elements in a group then bail because there is no need to continue as algorithm take first two elements with
* Do the process multiple times on the resultant output, every iteration would increase the quality of the output, that being said the implementation does not have a good time complexity, need to look into improving this.

Overall, this lab was a good challenge and look forward to hearing what a very good solution to the problem is.