COMP 550.001 - Fall 2017
Assignment 7

Part A due: Monday, December 4, 2017 (start of class)
Part B due: Wednesday, December 6, 2017 (4:00 p.m.)

For Part A, you should submit a physical copy of your written homework at the start of class.
For Part B, you should submit a .tar.gz or .zip file with your solutions on Sakai.

Part A: Due Monday, December 4, 2017

Be sure to include a collaboration statement with your assignment, even if you worked alone.

[10 points] Problem 1: CLRS Exercise 29.3-5

Solve the following linear program using the Simplex algorithm:

$$
\begin{array}{lrcrcr}
\text{maximize} & 18x_1 & + & 12.5x_2 & & \\
\text{subject to} & & & & & \\
& x_1 & + & x_2 & \leq & 20 \\
& x_1 & & & \leq & 12 \\
& & & x_2 & \leq & 16 \\
& & x_1, x_2 & & \geq & 0 \; .
\end{array}
$$

Make sure to show the state of the linear program after each iteration, list x_e and x_ℓ for each iteration, and state the z value that results from each basic solution.

[12 points] Problem 2: CLRS Exercise 24.4-1

Draw the constraint graph, and find a feasible solution or determine that no feasible solution exists for the following system of difference constraints:

$$
\begin{aligned}
x_1 - x_2 &\leq 1 \; , \\
x_1 - x_4 &\leq -4 \; , \\
x_2 - x_3 &\leq 2 \; , \\
x_2 - x_5 &\leq 7 \; , \\
x_2 - x_6 &\leq 5 \; , \\
x_3 - x_6 &\leq 10 \; , \\
x_4 - x_2 &\leq 2 \; , \\
x_5 - x_1 &\leq -1 \; , \\
x_5 - x_4 &\leq 3 \; , \\
x_6 - x_3 &\leq -8 \; .
\end{aligned}
$$

[22 points] Problem 3: extension of CLRS Exercise 29.2-2

a) Write out explicitly the linear program corresponding to finding the shortest path from node s to node y in Figure 24.2(a).

b) Put the resulting linear program into slack form. For clarity, it might help to use d_k to represent the variable for a node k's shortest distance, and x_j to represent slack variable j.

c) Solve the resulting linear program. (Hint: your result should match that of the shortest path from s to y from Figure 24.2.)

[8 points] Problem 4: CLRS Exercise 34.2-1

Two graphs $G = (V, E)$ and $G' = (V', E')$ are ***isomorphic*** if there exists a bijection $f : V \to V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$. In other words, we can relabel the vertices of G to be vertices of G', maintaining the corresponding edges in G and G'. You can find an example of two isomorphic graphs in Figure B.3(a) of appendix B.

Consider the language

$$\text{GRAPH-ISOMORPHISM} = \{\langle G_1, G2\rangle \ : \ G_1 \text{ and } G_2 \text{ are isomorphic graphs}\}.$$

Prove that GRAPH-ISOMORPHISM $\in$ NP by describing a polynomial-time algorithm to verify the language.

[8 points] Problem 5: Subset-Sum

In the ***subset-sum problem***, we are given a finite set S of positive integers and an integer target $t > 0$. We ask whether there exists a subset $S' \subseteq S$ whose elements sum to t. For example, if $S = \{1, 2, 7, 14, 49, 50\}$ and $t = 58$, then the subset $S' = \{2, 7, 49\}$ is a solution.

We can define the problem as a language:

$$\text{SUBSET-SUM} \ = \ \{\langle S, t\rangle \ : \ \text{there exists a subset } S' \subseteq S \text{ such that } t = \sum_{s \in S'} s\}.$$

If integers are coded in binary, this problem is NP-complete. Show that SUBSET-SUM $\in NP$ by describing a polynomial-time algorithm to verify the language.

Part B: Due Wednesday, December 6, 2017

[40 points] K-Means Clustering

As shown in lecture, K-Means Clustering is an algorithm to group data points into clusters. The input to the algorithm is a set of n data points, $\{x_1, \cdots, x_n\}$, and an integer k, representing the number of clusters to form. Each data point is a *feature vector* of m features. For example, a data point with $m = 2$ might represent the sleep and stress levels of a person or the height and weight of a dog, and a data point with $m = 3$ might correspond to the red, green, and blue values of a pixel in an image.

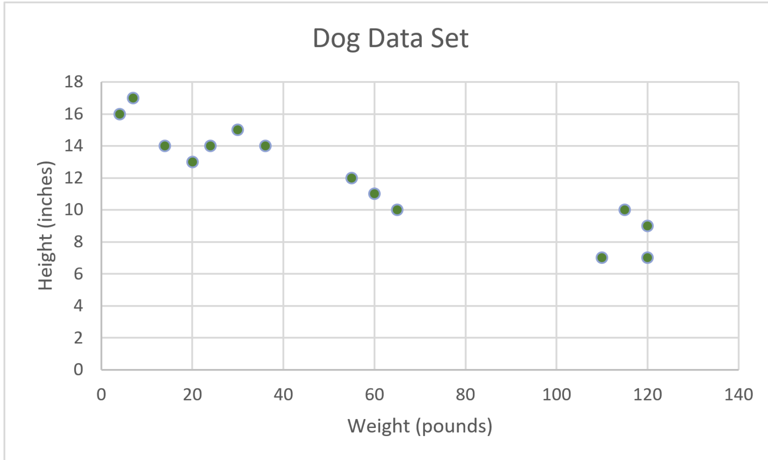
For this problem, you will implement the K-Means Clustering algorithm presented in lecture, and use the algorithm to cluster a data set of dogs. Recall from lecture that the K-Means Clustering algorithm consists of three steps:

K-MEANS-CLUSTERING(X, k):
normalize the data to have each feature in the range $[0, 1]$
$C = k$ initial random clusters from X
do
    assign each data point $x_i \in X$ to the nearest cluster $c_j \in C$
    update centroids
while (assignments changed)
return $map : centroid\ c_j \rightarrow \{x_i : x_i\ in\ cluster\ with\ centroid\ c_j\}$

In the dataset, each of the $n = 15$ data points represents the height and weight of a dog. For example, the Yorkshire Terrier (a "tiny" dog), could be represented as $(7, 6)$, indicating that it weighs 7 pounds and is 6 inches tall. The dog dataset is represented in the table below, and you'll find it in `Main.java`. As discussed in class, classifiers are typically built using *training data*, and verified using *testing data*. For this assignment, you'll focus on the training.

Dog Breed	**Weight (pounds)**	**Height (inches)**	**Label**
Chihuahua	4	5	Tiny
Yorkshire Terrier	7	6	Tiny
Miniature Poodle	14	12	Small
Beagle	20	13	Small
Pembroke Welsh Corgi	24	11	Small
Border Collie	30	20	Medium
Siberian Husky	36	22	Medium
Poodle	55	22	Large
Golden Cocker Retriever	60	20	Large
Labrador Retriever	65	30	Large
Bernese Mountain Dog	110	27	Huge
Great Pyrenese	115	28	Huge
Saint Bernard	120	35	Huge
Great Dane	120	40	Huge

This data set is visualized in the 2-dimensional plot below:



a) Normalizing the Dataset

For the first part of this assignment, you will normalize the dataset. Fill in the implementation of `normalizeDataset` in `KMeans.java`. You can calculate the expected results from the table above, but to give you an idea, you should expect that the "Chihuahua" data point normalizes to $(0, 0)$, the "Great Dane" data point normalizes to $(1, 1)$, and "Border Collie" normalizes to $(0.224, 0.429)$.

Your implementation should also set the class members `mins` and `maxes` for future use. These store the minimum and maximum value for each feature (e.g. weight and height), respectively.

b) Clustering the Data Points

Provided for you is the implementation to choose the initial centroids. For this part, fill in the implementation of `clusterDataPoints`. This method takes in the set of data points, as well as the current (empty) and prior iteration's clusters. You should always return `true` if `isFirstAssignment` is set to `true`. In addition, this method should return `true` if any of the points ends up in a different cluster than it was the prior iteration.

c) Updating the Centroids

You're on the last stretch! The last piece of the algorithm is to update the centroids at the end of the iteration. In the method `updateCentroids`, you should update the centroid of each cluster to be the mean of the data points in the cluster. You will likely find the `getNormalized` method of the `FeatureVector` class useful. This method gives the normalized feature value for the given dimension (0 to $m - 1$). In addition, you should update the label of the centroid to be the mode of the labels in the cluster. Don't update any of the fields directly; use the `updatecentroid` method of `FeatureVector` for this.

d) Classification

You should now have a working implementation of a K-Means classifier. Play around with the clusters for different values of k. Find one that seems to give reasonable results. Using your chosen value of k, run your classifier 5 times. For each run, record the number of iterations to convergence, and the final label assigned to the "Cocker Spaniel" data point. Make sure to include these values in your `readme` file.