Homework 7:
Compressive sensing and convex optimization

For this homework, you should use the `cvxpy` package for convex programming. There are installation instructions linked on the course webpage. If you are using colab, it should also be installed by default.

Problem 1: Compressive Sensing

Consider the image `wonderland-tree.png`. If it's easier to use, it's also represented as a 2D binary array in the file `wonderland-tree.txt`.

(a) [4 points] Let n be the total number of pixels and let k be the total number of 1's. What is k/n? Recall that compressive sensing only works when the image is sparse, so we are hoping that $k/n \ll 1$.

(b) [9 points] Fix a 1200×1200 matrix A of independently chosen $N(0,1)$ entries. Let A_r denote the first r rows of A. Let x be the image represented as a vector of 1200 pixels. Our compressed image will be $b_r = A_r x$.

Based on Lecture 12, write a linear program that recovers an approximation x_r to x from b_r and verify using `numpy.allclose()` that $x_{700} = x$ up to numerical precision (in other words, that your recovery is "exact.")

(c) [7 points] Let r^* be the smallest r such that $|x - x_r|_1 < 0.001$. Find r^*. **Hint:** Use binary search.

(d) [5 points] Plot $\|x_i - x\|_1$ for $i \in \{r-10, r-9, \ldots, r1, r, r+1, r+2\}$. Is there a sharp drop-off?

(e) [7 points] Now let A be a 1200×1200 matrix whose entries are chosen independently from the distribution:

$$A_{ij} = \begin{cases} -1 & \text{with probability } \frac{p}{2} \\ +1 & \text{with probability } \frac{p}{2} \\ 0 & \text{with probability } 1-p. \end{cases}$$

For each value of $p \in [0,1]$, let $\overline{r}(p)$ denote the average of r^* over 5 trials, where r^* is the smallest value r such that $|x - x_r|_1 < 0.001$. Report the value of $\overline{r}(0.5)$.

(f) [5 points] Plot the value $\overline{r}(p)$ for $p \in 0.2, 0.4, 0.6, 0.8, 1.0$.

(g) [6 points] How does the average number of measurements $\overline{r}(p)$ behave as a function of p? Can you offer an explanation? Why would smaller values of p be better for implementing the measurements?

(h) [3 points] In class, we (vaguely) justified why a Gaussian matrix should satisfy the restricted isometry condition (up to scaling) with high probability. How does the justification change for this new ensemble of random matrices? You don't need to give a formal proof; an informal justification should suffice. How would you expect the number of measurements necessary for this ensemble to satisfy RIP to scale with p?

Problem 2: Image Inpainting

Oh no! Some evil professor has defaced our poor pengu with lies. See `pengu_corrupted_1.png`. You will explore some methods for reconstructing the original image, which is a cropped version of the image from Homework 5. The following Python code will read in the image:

```
from PIL import Image
from numpy import array
img = array(Image.open("pengu_corrupted_1.png"), dtype=int)[:,:]
Known = (img > 0).astype(int)
```

Using matplotlib, you should display `Known` and make sure it matches what you expect:

```
import matplotlib.pyplot as plt
plt.gray()
plt.imshow(Known)
plt.show()
```

(a) [4 points] First explore the naive solution for image reconstruction. For every pixel in `img` that is 0 (unknown), replace it with the average of its (up to 4) known neighbors' pixels. If there are no known neighbors, then keep the pixel value 0. Submit your recovered image, as well as the code, and a two-to-three sentence explanation for why the naive solution seems to perform poorly.

The following code should reconstruct our pengu pretty well:

```
from cvxpy import Variable, Minimize, Problem, multiply, tv
U = Variable(img.shape)
obj = Minimize(tv(U))
constraints = [multiply(Known, U) == multiply(Known, img)]
prob = Problem(obj, constraints)
prob.solve(verbose=True)

#the recovered image is now in U.value
```

Warning: This code will take a while (it took around 5-10 minutes for me on colab)!

(b) [4 points] Submit your recovered image and a 1-2 sentence comparison between this image and the one you reconstructed naively.

(c) [4 points] The documentation for `tv` is linked from the course website. Write down the form of this function, and explain why this function is convex.

(d) [8 points] Now that you have a formula for `tv`, write the entire convex program that this code is solving, and then interpret what the convex program is doing.

(e) [8 points] Our poor beleaguered pengu can't catch a break—the same professor has inscribed an even more insidious lie on the little guy! See `pengu_corrupted_2.png`. You should find that if you try the same method as above, you will *not* be able to get rid of the message. Explain the difference between these two images, and why this lie is harder to erase. Then, explain how you would change the objective function from `tv` to something else to overcome this difficulty. You do not need to run this new algorithm.