CS 111: Homework 2: Due by 11:59 pm Sunday, January 29, 2023

Submit your paper as one PDF file, and tell GradeScope which page(s) each problem is on. If you worked with a partner, you must each turn in your own homework paper, and report the name and perm number of your partner. No groups of more than two allowed.

1. Consider the permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

1.1. Find a 4-permutation `v = [ something ]` such that `A[v,:]  == P @ A` holds for *every* 4-by-4 matrix A. Test your answer by running a few lines of Python, and include the code and output from your testing in your LaTeX writeup.

1.2. For the same P, find a 4-permutation `w = [ something ]` such that `A[:,w] == A @ P` holds for *every* 4-by-4 matrix A. Turn in your testing for your answer.

2. The routine `Lsolve()` is in the file `cs111/LU.py`. It is called as `y = Lsolve(L, b)`, where L must be unit lower triangular, that is, a square, lower triangular matrix whose diagonal is all ones. Modify `Lsolve()` to take an optional third keyword argument `unit_diag` that defaults to `True`. If `unit_diag` is `False`, your modified routine should not require (or `assert`) that the diagonal is all ones, but instead it should do the necessary arithmetic to get the right answer to $Ly = b$ for any nonsingular lower triangular matrix L. Test your answer, both by itself and with `LUsolve()`, and include a screenshot of your testing along with your code as part of your LaTeX writeup.

3. Write `Usolve()`, analogous to `Lsolve()` in the file `cs111/LU.py`, to solve an upper triangular system $Ux = y$. You should again include an optional argument `unit_diag`, as in problem (2), but this time its default should be `False`. Test your answer, both by itself and with `LUsolve()`, and include a screenshot of your testing along with your code as part of your LaTeX writeup. Hint: Loops can be run backward in Python, say from $n-1$ down to 0, by writing

```
for i in reversed(range(n)):
```

4. Suppose that A is a nonsymmetric invertible matrix, b is a vector, and that you have called

```
L, U, p = cs111.LUfactor(A).
```

Now suppose you want to solve the system $A^T x = b$ (not $Ax = b$) for x. Show how to do this using only calls to `Lsolve()` and `Usolve()` (as modified in problems (2) and (3)).

You may not call `LUsolve()` or any of `numpy`'s built-in solvers (like `npla.solve()`), and you may not call `LUfactor()` again. You are allowed to transpose any matrices you wish; recall that `M.T` means the transposed matrix M^T in `numpy`. Test your method in `numpy` on a randomly generated 6-by-6 matrix and show the code and output in Jupyter.