EE231002 Introduction to Programming

Lab07. Matrix Determinants

Due: Nov. 21, 2020

In this lab, you will write a program to calculate the determinant of a square matrix using Leibniz formula. In mathematics, a $n \times n$ square matrix can be represented as following.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}.$$

Then the Leibniz formula for determinant of matrix $\mathbf{A}$ is:

$$\det(\mathbf{A}) = \sum_{\sigma \in S_N} \operatorname{sgn}(\sigma) \prod_{i=1}^{N} A_{i,\sigma_i}.$$

Here the summation is computed over all permutations, $\sigma \in S_n$, of the set $\{1, 2, \cdots, n\}$, and S_N denotes all possible such permutations. The i'th position of a permutation σ is denoted by σ_i. For example, $\sigma = 231$ is a permutation of $\{1, 2, 3\}$, and $\sigma_1 = 2$, $\sigma_2 = 3$ and $\sigma_3 = 1$. The function $\operatorname{sgn}(\sigma)$ is defined as following.

$$\operatorname{sgn}(\sigma) = \begin{cases} 1 & \text{if } \sigma \text{ can be obtained by an even number of swapping} \\ & \text{from the first permutation } \{1, 2, \cdots, n\}, \\ -1 & \text{otherwise.} \end{cases}.$$

Example of a 3×3 matrix $\mathbf{A}$ and its determinant are shown below:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{aligned} \det(\mathbf{A}) =& a_{11}a_{22}a_{33} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} \\ &+ a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} \end{aligned}$$

Note that the first subscript of each matrix element of each term is always arranged from 1 to n, then the second subscript needs to go through all permutations of the set $\{1, 2, 3\}$. The function **sgn** is observed to have the following values.

$$\text{sgn}\{123\} = 1 \qquad \text{sgn}\{132\} = -1$$
$$\text{sgn}\{213\} = -1 \qquad \text{sgn}\{231\} = 1$$
$$\text{sgn}\{312\} = 1 \qquad \text{sgn}\{321\} = -1$$

It is suggested that you write a function `Pandita`(**P**) that generates the next lexicographical permutation and return **sgn** value of that permutation. This **sgn** should be based on the cumulative number of swappings from the first permutation to the current one, rather than the swappings needed for this permutation only. Please also note that the mathematical description above, the set is from 1 to n while in `C` the array index should be from 0 to $n-1$. Using this function then the matrix determinant can be easily generated. The prototype of this function can have the following form.

```
// This function generate the next lexicographic permutation based on
// Pandita algorithm.
//   input: P contains the previous permutation
//   output: return sgn of the cumulative number of swappings
//               and return 0 if no more permutation
//           P contains the next permutation

int Pandita(int P[N]);
```

Several matrices of various sizes are provides for you to test your program. They are `mat1.in` to `mat11.in`. Each file contains $n \times n$ integers that can be read in using double loops. To avoid editing the file for different n, please add the following compiler directives at the beginning of your program.

```
#if !defined(N)
#define N 3
#endif
```

Now you can recompile your program for different n as following.

```
$ gcc -DN=11 lab07.c
$ ./a.out < mat11.in
```

Execution of `a.out` is followed by "`< mat11.in`" redirection. Using this mechanism, the content of the file `mat11.in` is used as the input, and the standard `scanf` function can perform normal read operations.

The example program output is shown at the end of this `pdf` file. Please make sure your output matches with the example.

Notes.

1. Create a directory **lab07** and use it as the working directory.
2. Name your program source file as **lab07.c**.
3. The first few lines of your program should be comments as the following.

   ```
   // EE231000 Lab07 Matrix Determinant
   // ID, Name
   // Date:
   ```

4. After you finish verifying your program, you can submit your source code by

   ```
   $ ~ee2310/bin/submit lab07 lab07.c
   ```

   If you see a "submitted successfully" message, then you are done. In case you want to check which file and at what time you submitted your labs, you can type in the following command:

   ```
   $ ~ee2310/bin/subrec
   ```

   It will show the last few submission records.

5. Example output:


```
$ gcc -DN=3 lab07.c
$ ./a.out < mat2.in
Matrix A is
   1 0 0
   2 4 5
   3 1 2
det(A) = 3
```