Week 6 — October 11, 2018

Homework

- Read chapter 6 pp. 219-255 and chapter 12 pp. 489-519 in "Beginning C"

Exercises

1. Take this quiz to test your understanding of strings.
2. A sparse matrix is a matrix in which many entries are zero. Storing such a matrix as a full two-dimensional array is often inefficient in terms of both *space* (i.e., memory) and *time* (i.e., computational cost). For example, a diagonal matrix of order n has zeros in all off-diagonal positions, and hence it can be stored as a one-dimensional array of length n corresponding to the n diagonal elements. Furthermore, multiplication with a diagonal matrix is much cheaper computationally than multiplication with a general dense matrix. More generally, a sparse matrix C of size $m \times n$ can be represented as a set of *triplets* of the form (i, j, C_{ij}). For example, the matrix

$$C = \begin{bmatrix} 1.0 & 0 & 0 & 4.0 \\ 0 & 2.0 & 3.0 & 0 \\ 5.0 & 0 & 6.0 & 7.0 \end{bmatrix}$$

has seven nonzeros and can be represented in triplet form as

$$\{(1, 1, 1.0), (3, 1, 5.0), (2, 2, 2.0), (2, 3, 3.0), (3, 3, 6.0), (1, 4, 4.0), (3, 4, 7.0)\}.$$

When working with sparse matrices in triplet form, it is convenient to define a structure that stores both the dimensions of the matrix as well as three arrays corresponding to the row indices, the column indices, and the nonzero values. Such a structure may look like this:

```c
/* Structure representing a sparse matrix in triplet form */
struct sparse_triplet {
    size_t m;   /* number of rows     */
    size_t n;   /* number of columns  */
    size_t nnz; /* number of nonzeros */
    size_t * I; /* pointer to array with row indices    */
    size_t * J; /* pointer to array with column indices */
    double * V; /* pointer to array with values         */
  };
```

   - Write a function that can read a sparse matrix in triplet form from a text file in which each line corresponds to a nonzero element. Your function should allocate and return a `sparse_triplet` structure. You may assume that the first line of the text file contains the dimensions m and n as well as the number of nonzeros, i.e., the text file that corresponds to the above example should look like this:

```
3 4 7
1 1 1.0
3 1 5.0
2 2 2.0
2 3 3.0
3 3 6.0
1 4 4.0
3 4 7.0
```

The function should convert the indices to 0-based indices so that row indices are between 0 and $m-1$ (instead of 1 to m) and column indices are between 0 and $n-1$ (instead of 1 and n).

Hint: Your function prototype could look like this:

```
struct sparse_triplet * read_sparse(const char * filename);
```

Start by allocating memory for a `sparse_triplet` structure, and read the first line of the file to determine `m`, `n`, and `nnz`. Then allocate memory for the row/columns indices (`I` and `J`) and the nonzero values (`V`), i.e., two `size_t` arrays of length `nnz` and a `double` array of length `nnz`. Finally, read the remaining `nnz` lines of the file and fill the arrays. Do not forget to close files that you open.

- Write a short program to test your function. .

3. Write a function that takes a `sparse_triplet` structure and a file name as input and creates a text file with the matrix in triplet form. The text file should follow the format described in the previous exercise (i.e., the first line should contain the dimensions `m n nnz` and the remaining `nnz` lines should contain the triplets). Write a short program to test your function.

   Hint: Your function prototype could look like this:

   ```
   int write_sparse(const char *filename, struct sparse_triplet *A);
   ```

4. Go to CodeJudge to do the "Week 06" exercises.