Module 3 Portfolio Milestone 1: Analysis of Algorithms and Data Structures

Donghwan Kim

Colorado State University Global

CSC 506: Design and Analysis of Algorithms

Dr. Lori Farr

May 14, 2022

Portfolio Milestone: Analysis of Algorithms and Data Structures

This Portfolio Milestone 1 of the project which is concerned with the data structure and operations of multi-dimensional arrays describes an outline of the planned analysis and and results. Multi-dimensional lists and blockchains are two data structures widely used in the emerging fields of data science and artificial intelligence. Fortran 90/95 provides multi-dimensional arrays but has some limitations. Python does not provide a multi-dimensional list. As will be shown, however, the Python list can be used to construct a multi-dimensional list since it allows multiple data types in a list. This Milestone 1 assignment discusses about the structure of multi-dimensional lists and the definition of core operations to the data structure.

Muti-dimensional list as data container

Multi-dimensional lists and blockchians are two of mostly widely-used data structures in the emerging fields of data science and artificial intelligence. Blockchains are gaining its popularity due to strength in security. Cryptocurrency is an example of it. Multi-dimensional list can contain various data types of numbers, strings, booleans, and so on. For example, it can contain both image data and text data in a multi-dimensional list, which is essential in many applications of artificial intelligence and machine learning.

Fortran 90/95 provides multi-dimensional array data structure up to 7 dimensions but has some limitations (Wagener 1998). The 1-dimensional array is considered as a sequence of elements like the Python list. The 2-dimensional array can be constructed in several ways. Fortran 90/95 has the format of columns and rows, and the elements of it can be accessed with [i, j] for the i'th row and j'th column. The index of 3-dimensional arrays has the form of [i, j, k]. However, Fortran 90/95 does not allow different data types in a multi-dimensional array. All the columns should be declared as same data type. Numbers can not be with strings, for example. One way to incorporate multiple data types is to use separate arrays by data type. It will add complexity in designing operations.

To serve as a data container, a multi-dimensional list has to contain various types of data

including numbers and strings. However, one restriction is that all the elements in a column should be in the same type. Both numbers and strings can not be in a same column. In the sense, Excel spreadsheets have drawbacks. Python does not provide multi-dimensional list (Van Rossum and Drake Jr 2009). However, as will be shown, the multi-dimensional list can be implemented in various ways with data structures that Python provides. This Portfolio project is about how to construct multi-dimensional list along with some of operations to the data structure.

Operations to multi-dimensional lists

Once the 1-dimensional array like the Python list is extended to higher-dimensional lists, there are lots of considerations in designing data structure and operations. The order of storage also matters. It starts with a 2-dimensional list, denoted X, whose elements are $m \times n$ where m is the number of rows and n is the number of columns. Fortran 90/95 declares the data structure with $DIMENSION(m, n)$. The elements can be accessed with the index [i, j] in which i goes to 1 to m and j goes to 1 to n. The update(X) operation is possible, for example, with X[i, j] for each elements or X[:,j] for all the elements in the j'th column. In this way, more high-dimensional lists are possible.

The $m \times n$ 2-dimensional list can be constructed in various ways using Python's basic data structures like list, nested list, dictionary, and so on. For example, a nested list can be used to contain the 2-dimensional list as follows:

$$[[X[:, 0]], [X[:, 1]], [X[:, 2]], \cdots, [X[:, n-1]]]$$

in which X[:,j] is m elements of the j'th column. The index j goes from 0 to n-1. Each nested list has m elements and it has n nested list. Therefore, there are $m \times n$ elements in total. The multi-dimensional list can be constructed with m Python lists with n elements or n Python lists with m elements. This is where efficiency matters: Which one is more efficient? It is also possible to contain all the elements in a simply list like $[X(0, 0), X(1, 0), \cdots, X(m - 1, 0), \cdots, X(m - 1, n - 1)]$.

in the column-major order. One of the considerations in designing data structures for a data container is column names which provide information what each of columns in data means.

$$
\begin{aligned}
&[\{'record0' : \{'column0' : X[0,0], 'column1' : X[0,1], \cdots, 'columnLast' : X[0,n-1]\},\\
&\{'record1' : \{'column0' : X[1,0], 'column1' : X[1,1], \cdots, 'columnLast' : X[1,n-1]\},\\
&\vdots\\
&\{'recordLast'\{'column0' : X[m-1,0], 'column1' : X[m-1,1], \cdots, 'columnLast' : X[m-1,n-1]\}
\end{aligned}
$$

As a data container, a multi-dimensional list has lots of operations including update(X) and delete(X). It has most common operations applied in the 1-dimensional list (Lysecky and Vahid 2019: Module 4). Most operations applied in the 1-dimensional list become complicated and extended as the dimension increases, which includes adding new records or columns and converting to other data type.

Conclusions

This Portfolio Milestone 1 discusses about topic selection and an outline of the project plan and expected results. The project is concerned with multi-dimensional lists, one of data structures widely used for data analysis and modern artificial intelligence. As the beginning of the project, it explains about multi-dimensional lists and operations to the data structure. Implementation and test are discussed in Milestone 2.

References

Lysecky, R. and Vahid, F. (2019). Design and analysis of algorithms. In Lysecky, R. and Vahid, F., editors, *Data structures essential: Pseudocode with python examples*. Zybooks.

Van Rossum, G. and Drake Jr, F. L. (2009). *Python tutorial*. Scotts Valley, CA: CreateSpace.

Wagener, J. (1998). *Fortran 90 concise reference*. Absoft Corporation.