Lab 1: 3-D by 3-D Array Multiplication

Knowledge Dzumba (813137), Sanele Ndlovu (716411)

February 24, 2019

Algorithm 1 computes the addition of two matrices A and B. This is done by adding corresponding elements in both these matrices. The result is returned in matrix C

Algorithm 1: rank2TensorAdd

```
Input: n x n matrices A and B
Output: n x n matrix C
________________________________
n = A.rows
Let C be a newly created n x n matrix
for i = 0 to n - 1 do
    for j = 0 to n - 1 do
        C[i][j] = A[i][j] + B[i][j]
    end
end
return C
```

Algorithm 2 computes the multiplication of two matrices A and B. The algorithm uses rank 2 tensor contraction method to compute this multiplication and the result is returned in C

Algorithm 2: rank2TensorMult

```
Input: n x n matrices A and B
Output: n x n matrix C
________________________________
n = A.rows
Let C be a newly created n x n matrix
for i = 0 to n - 1 do
    for j = 0 to n - 1 do
        C[i][j] = 0
        for k = 0 to n - 1 do
            C[i][j] = C[i][j] + A[i][k]*B[k][j]
        end
    end
end
return C
```

Algorithm 3 performs 3D matrix addition of two 3D matrices A and B. This operation is an element wise addition, so corresponding elements in matrix A and matrix B are simply added together and the result of the addition is returned in matrix C

Algorithm 3: rank3TensorAdd

```
Input: n x n x n matrices A and B
Output: n x n x n matrix C

n = A.rows
Let C be a newly created n x n x n matrix
for i = 0 to n - 1 do
    for j = 0 to n - 1 do
        for k = 0 to n - 1 do
            C[i][j][k] = A[i][j][k] + B[i][j][k]
        end
    end
end
return C
```

Algorithm 4 performs 3D matrix multiplication of two 3D matrices *A* and *B*. This is done using a similar strategy as that used for 2D matrix multiplication, modified to fit the case of a 3D matrix. In this algorithm, a row is considered to be a 2D horizontal matrix that is part of the 3D matrix (cube) *A*, and a column is considered to be a left-facing 2D vertical matrix that is also part of a 3D matrix *B*. In the case of 2D matrix multiplication, a row of some matrix *D* is multiplied by a column of some second matrix *D* to produce a single element(scalar) of the resulting matrix *F*, while in the case of 3D matrix multiplication, a row of matrix *A* is multiplied by a column of matrix *B* to produce a vector whose elements form part of the resulting *C* matrix. An element at index *i,j,k* of the resulting *C* matrix is found using the formula:

$$C_{ijk} = \sum_{l} A_{ilk} \times B_{ljk}$$

It is assumed in this algorithm that the dimensions of both *A* and *B* matrices are the same, that is they are both $n \times n \times n$.

Algorithm 4: rank3TensorMult

Input: *n* x *n n* matrices *A* and *B*

Output: *n* x *n* x *n* matrix *C*

```
n = A.rows
Let C be a newly created n x n x n matrix
for i = 0 to n - 1 do
    for j = 0 to n - 1 do
        for k = 0 to n - 1 do
            C[i][j][k] = 0
            for l = 0 to n - 1 do
                C[i][j][k] = C[i][j][k] + A[i][l][k]*B[l][k][k]
            end
        end
    end
end
return C
```