CompSci-230: Homework 3

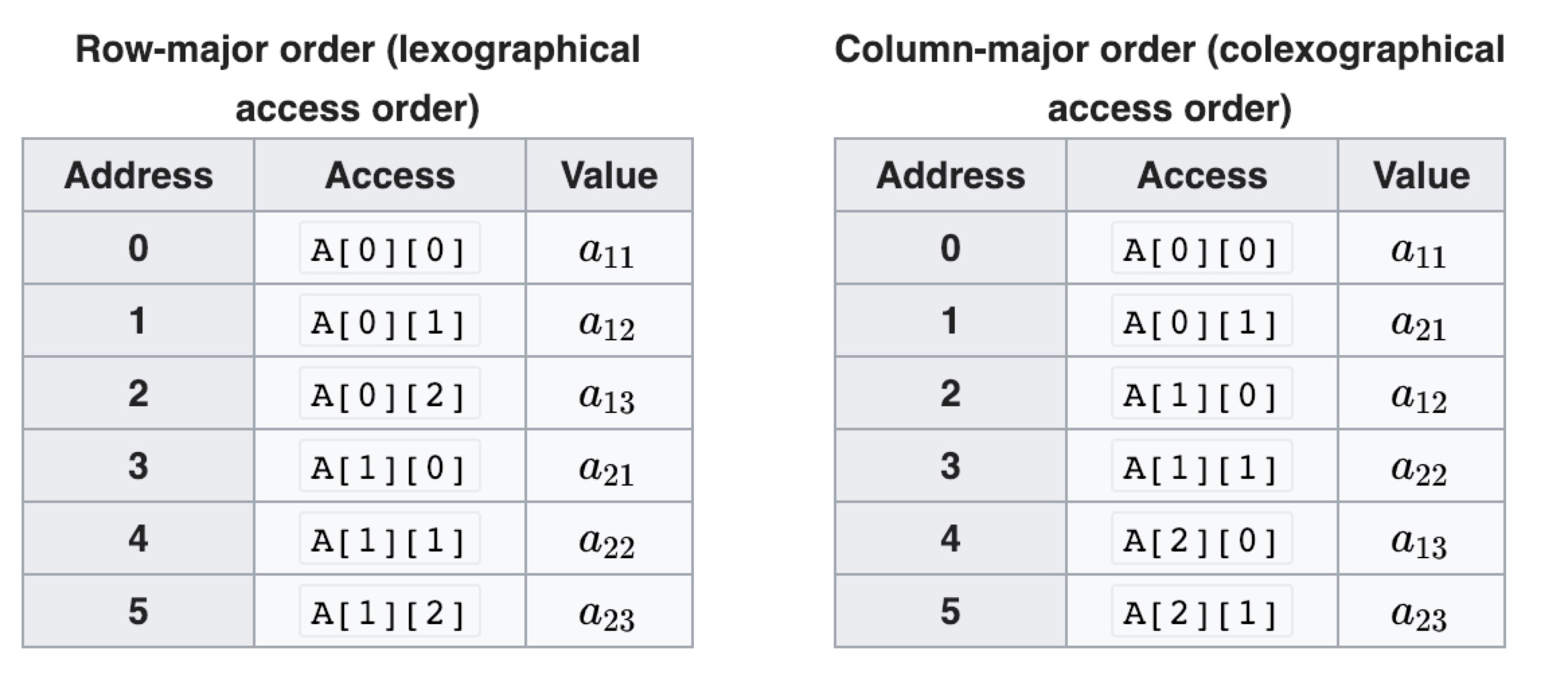
Jitao Zhang

1. **Introduction**

Main Calculation: Ci,j = Ci,j + Ai,k × Bk,j

Explanation: i means the row index of matrix A and j means the column index of B. k means the kth element in each row or column.

Two ways of mapping 2d matrix data to memory:



**In order to make more detailed analyze, I give the big O notation a constant prefix.**

1. **Single Processor Computer Analyze**

All the cases below can be row major or column major because it is running in a single processor computer.

Calculation Time Complexity is O(n3)

Communication Time Complexity is 0

Space Complexity is O(3n2)

1. i, j, k

Every time the innermost k moves 1, Ci,j gets updated. So after each iteration of innermost k, a new element Ci,j can be generated.

1. i, k, j

Every time the innermost j moves 1, a different Ci,j gets updated. So after the whole inner k and inner j loop once for a specific i, a new set of elements Ci,j for a certain i can be generated. So it is generated by row.

1. j, i, k

Nearly the same as i, j, k. But the difference is that the outermost is j, so the generating sequence of new element of C is different. In (i, j, k), the new element in C is generated by each row, but in (j, i, k), the new element in C is generated by each column.

1. j, k, i

Nearly the same as i, k, j. But the difference is that the outermost is j. So after the whole inner k and inner i loop once for a specific j, a new set of elements Ci,j for a certain j can be generated. So it is generated by column.

1. k, i, j

Every time the innermost j moves 1, a different Ci,j gets updated. After the whole 3 loops finish, all the new elements are generated simultaneously.

1. k, j, i

The same as (k, i, j), because all the new elements are generated after the whole 3 loops. The difference is the update sequence. For (k, i, j), the Ci,j gets updated by row. For (k, j, i), the Ci,j gets updated by column.

1. **N-Processor Rings Analyze**
2. i, j, k

We store a certain row of A and a certain column of B in each processor. So for each processor, it needs to run a two for loop (j and k), but when the inner k loop once, the processor needs to get a column of B from a certain processor. So it needs time to communicate. But the space is saved a lot(otherwise we need to store whole B for each processor). And we also need to store n new elements for ith row of C. So the memory should be row major for convenience.

Calculation Time Complexity is O(n2)

Communication Time Complexity is O(n2)

Space Complexity is O(3n)

1. i, k, j

We store a certain row of A and all B in each processor. This case is different from with previous one because the inner loop is for j so we need all the B to calculate for each innermost for loop. And we also need to store n new elements for ith row of C. So the memory should be row major for convenience.

Calculation Time Complexity is O(n2)

Space Complexity is O(n2)

1. j, i, k

Nearly the same as (i, j, k). We store a certain column of B and a certain row of A in each processor. So for each processor, it needs to run a two for loop (i and k), but when the inner k loop once, the processor needs to get a row of A from a certain processor. And we also need to store n new elements for jth column of C. So the memory should be column major for convenience.

Calculation Time Complexity is O(n2)

Communication Time Complexity is O(n2)

Space Complexity is O(3n)

1. j, k, i

Nearly the same as (i, k, j). We store a certain column of B and all A in each processor. And we also need to store n new elements for jth column of C. So the memory should be column major for convenience.

Calculation Time Complexity is O(n2)

Space Complexity is O(n2)

1. k, i, j

We need to store all the information of A or B in each processor, no matter in row major or column major. After O(n2) time calculation, each processor only gets partial result for each Ci,j. So in order to get the final result, we need to choose one of n processors to be the “result” processor to store final result. But the case is worst because all the n processors contain only a “partial” result, we still need to recalculate all the partial matrix in another processor, which needs another total O(n3) calculation time and O(n2) communication time.

Calculation Time Complexity is O(n3)

Space Complexity is O(3n2)

1. k, j, i

Totally the same as k, i, j.

Calculation Time Complexity is O(n2)

Space Complexity is O(3n2)

1. **Summary**

To sum up, the i, j, k (the same as j, i, k) in n processors and i, k, j (the same as j, k, i) in n processors are both good. And the single processor case is next. The worst one is k, i, j (the same as k, j, i) in n processors, it is even worse than single processor.