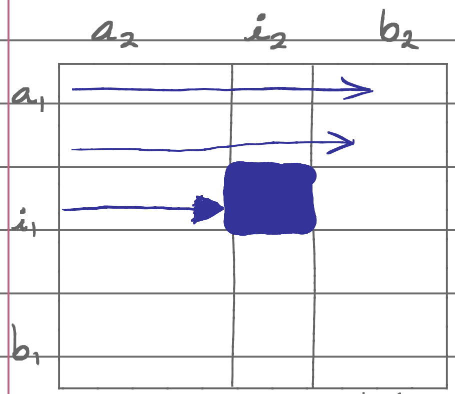
36 11-05-09

Note Title 11/5/2009

2D Array $[a_1 \ldots b_1, a_2 \ldots b_2]$

Row-Major

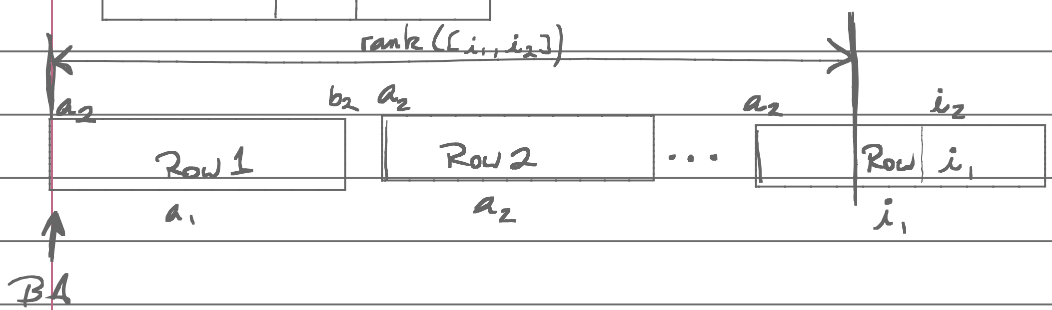


$$address([i_1, i_2]) = BA + rank([i_1, i_2]) \times ES$$

$$rank([i_1, i_2]) = (i_1 - a_1)(b_2 - a_2 + 1)$$

$$address([i_1, i_2]) = BA + \left((i_1 - a_1)(b_2 - a_2 + 1) + (i_2 - a_2)\right) \times ES$$

$$= \underbrace{BA - (a_1(b_2 - a_2 + 1) + a_2) \times ES}_{\text{Virtual BA}} + (i_1(b_2 - a_2 + 1) + i_2) \times ES$$

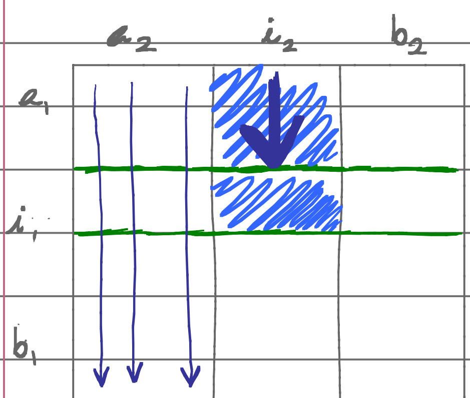


$x = A[i_1, i_2];$

$x = A[3, i_2];$

$x = A[3, 5];$

Column-Major



symmetry law for 2-dimensional case

$i_1 \leftrightarrow i_2$

$a_1 \leftrightarrow a_2$

$b_1 \leftrightarrow b_2$

$$rank([i_1, i_2]) = (i_2 - a_2)(b_1 - a_1 + 1) + (i_1 - a_1)$$

$$address([i_1, i_2]) = BA + \left((i_2 - a_2)(b_1 - a_1 + 1) + (i_1 - a_1)\right) \times ES$$

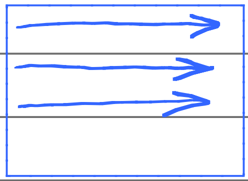
$$= \underbrace{BA - (a_2(b_1 - a_1 + 1) + a_1) \times ES}_{\text{Virtual BA}} + (i_2(b_1 - a_1 + 1) + i_1) \times ES$$

- Almost all languages use Row-Major order. (exception: FORTRAN = Column-Major)
- Knowledge of order is practically important when traversing a large portion of big arrays, element by element (ex: $1000 \times 1000 = 1{,}000{,}000$

  $10000 \times 10000 = 100{,}000{,}000$)

- Traverse in the order used in the implementation. Otherwise, tremendous slow-down occurs due to "thrashing" of virtual main memory
- Suppose main memory can contain one row at a time - the rest is stored in the hard disk

ex =

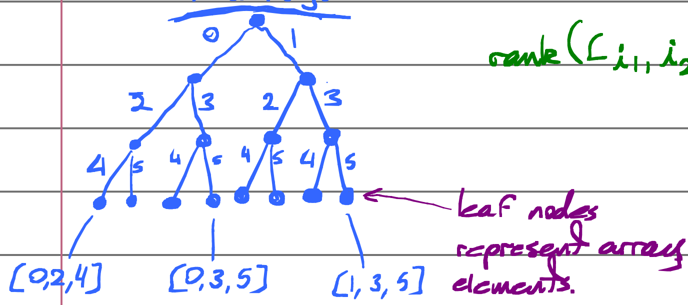


- What if your program traverses in column-major order?

N-Dimensional Array [$a_1 \ldots b_1, \ldots, a_n \ldots b_n$]

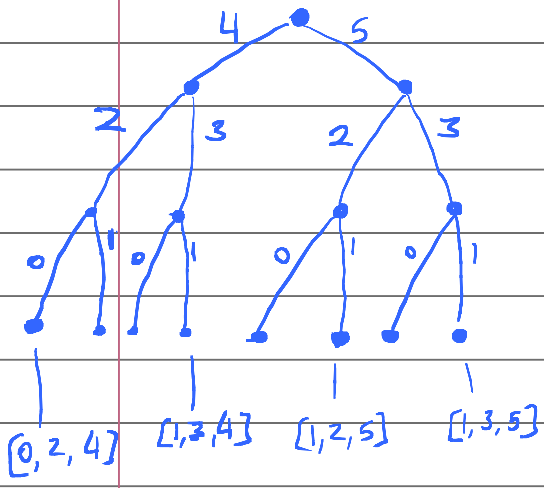
- Derive rank([$i_1, \ldots, i_n$]) by access tree
  - Example: [0...1, 2...3, 4...5]

Row-Major



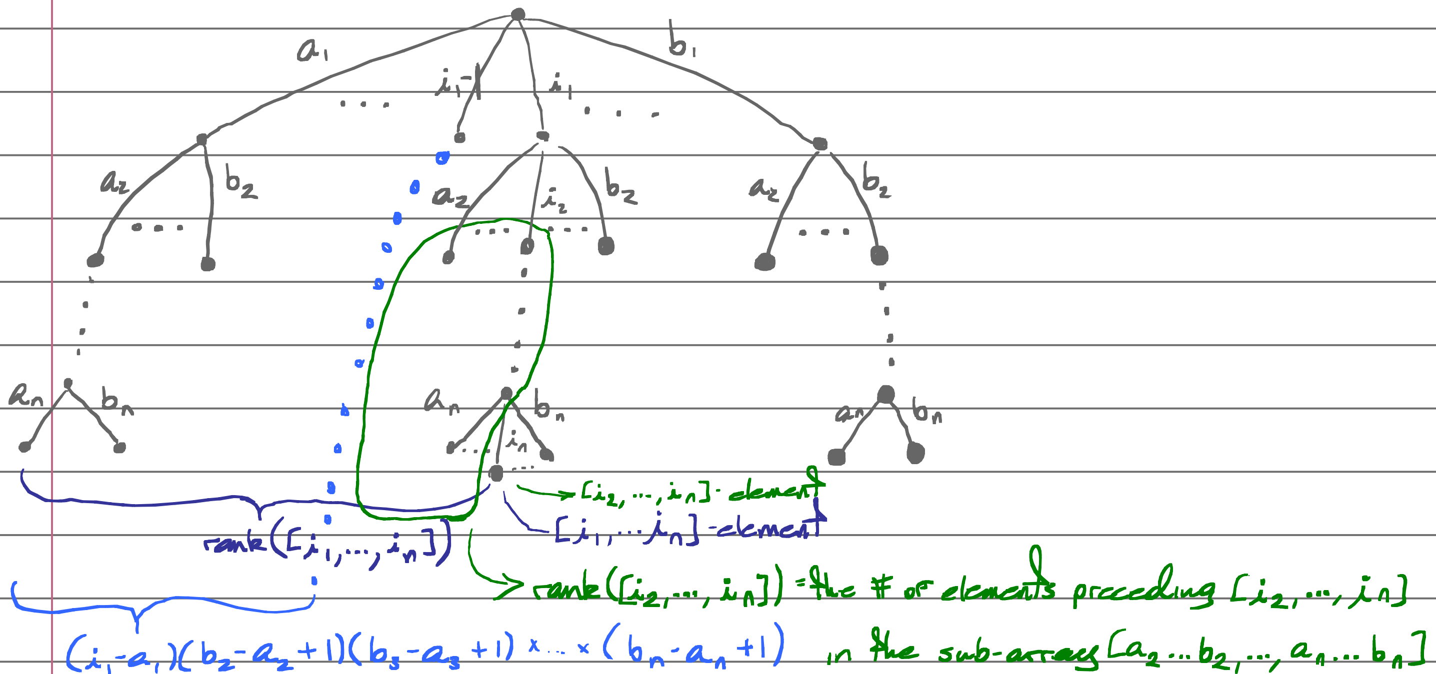
rank([i_1, i_2, i_3]) = the # of leaves to the left of [i_1, i_2, i_3]

Column-Major



rank([i_1, i_2, i_3]) = the # of leaves to the left of [i_1, i_2, i_3]

Access Tree for N-Dimensional, Row-Major



The formula for rank: $\text{rank}([i_1, \ldots, i_n]) = (i_1-a_1)(b_2-a_2+1) \times \ldots \times (b_n-a_n+1) + \text{rank}([i_2, \ldots, i_n])$

rank$([i_k, \ldots, i_n])$ = the rank of $[i_k, \ldots, i_n]$-element in the sub-array $[a_k \ldots b_k, \ldots, a_n \ldots b_n]$ for $1 \le k \le n$

$$\text{rank}([\,]) = 0$$

$$\text{rank}([i_1]) = (i_1 - a_1) + \text{rank}([\,]) = i - a_i + 0 = i_1 - a_1$$

$$\text{rank}([i_1, i_2]) = (i_1-a_1)(b_2-a_2+1) + \text{rank}([i_2])$$
$$= (i_1-a_1)(b_2-a_2+1) + (i_2-a_2)$$

$$\text{rank}([i_1, i_2, i_3]) = (i_1-a_1)(b_2-a_2+1)(b_3-a_3+1) + \text{rank}([i_2, i_3])$$
$$= (i_1-a_1)(b_2-a_2+1)(b_3-a_3+1) + (i_2-a_2)(b_3-a_3+1) + (i_3-a_3)$$