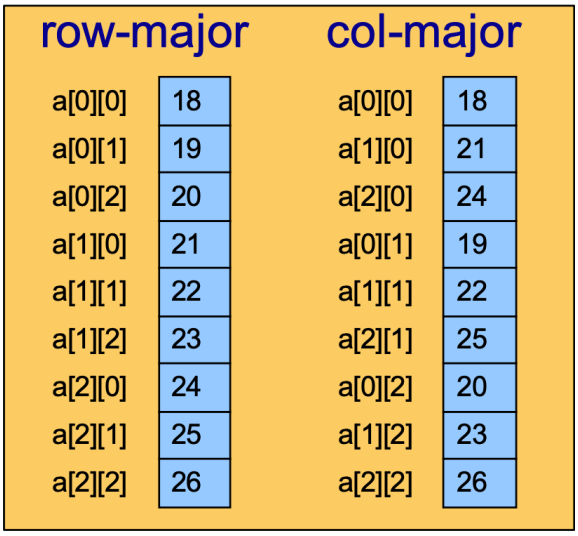
Difference:

Spatial Locality	Temporal Locality
The probability that a nearby address space will be accessed by the OS.	The probability that the same address space will be accessed by the OS.

C: Row major, Good spatial locality.



Characteristics:

- Access in row-major order ⇒ good spatial locality
- Access in column-major order ⇒ poor spatial locality

Significance:

Speeds up things (One of Os's major goals).

Locality example

```
sum = 0;
for (i = 0; i < n; i++)
   sum += a[i];
```

Typical code
(good locality)

- **Temporal locality**
  - *Data:* Whenever the CPU accesses **`sum`**, it accesses **`sum`** again shortly thereafter
  - *Instructions:* Whenever the CPU executes **`sum += a[i]`**, it executes **`sum += a[i]`** again shortly thereafter
- **Spatial locality**
  - *Data:* Whenever the CPU accesses **`a[i]`**, it accesses **`a[i+1]`** shortly thereafter
  - *Instructions:* Whenever the CPU executes **`sum += a[i]`**, it executes **`i++`** shortly thereafter

6

Addr	Content
0xFFAB	Data(sum)
…	
0xAB12	Instr(sum += a[i])

In this case, the OS can keep fetching data and instruction from the same location again and again.

Addr	Content
0xFFAB	Data(a[0])
0xFFAC	Data(a[1])
0xFFAD	Data(a[2])
…	
0xAB12	Instr(sum += a[i])
0xFF15	Instr (i++)

In this case, the OS has to fetch data and instructions from different memory locations each time.

How to ensure good program execution time?

In C, you can do so by Creating a program that is **row major**.

Suggested reading :

https://www.cs.princeton.edu/courses/archive/spr17/cos217/lectures/18_StorageMgmt.pdf