Arrays

Angel R. Perez

Basics: Concepts

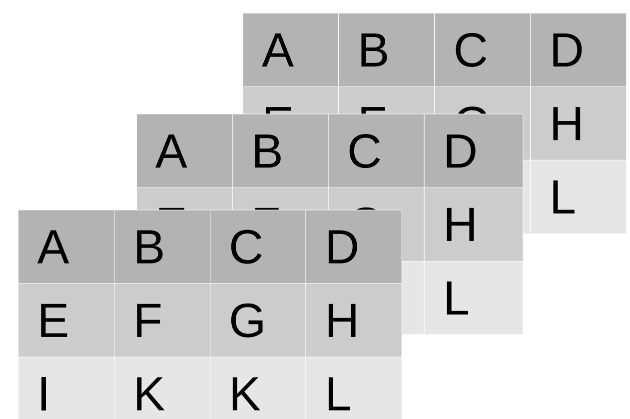
- **Array:** a collection of elements, each identified by an index or key. *In a computer is represented as consecutive memory addresses*

Zero-Index	0	1	2	...	N-1
One-Index	1	2	3	...	N
Values	A	B	C	...	Z

- **Vector:** one-dimensional array
- **Matrix:** multi-dimensional array

A	B	C	D	...	Z

A	B	C	D
E	F	G	H
I	K	K	L



Basics: Compact Layout

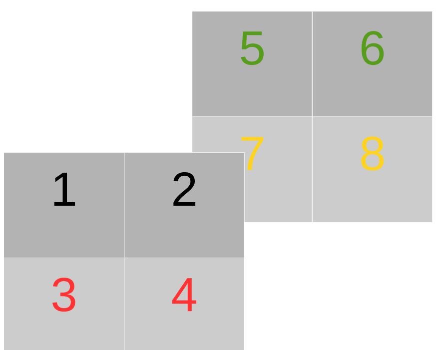
1	2	3
4	5	6
7	8	9

Row-Major order

1	2	3	4	5	6	7	8	9

Column-Major order

1	4	7	2	5	8	3	6	9



Row-Major order

1	5	2	6	3	7	4	8

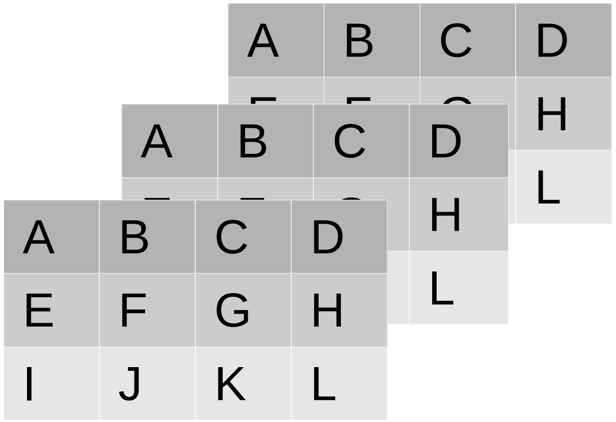
Basics: Addressing

A	B	C	D	...	Z

- **char A1[N]**
  - A1[i] | 0 < i < N
  - ADDR = B + (i * S)

A	B	C	D
E	F	G	H
I	J	K	L

- **char A2[N][M]**
  - A2[i][j] | 0 < i < N and 0 < j < M
  - ADDR = B + (i * R) + (j * C)



- **char A3[N][M][O]**
  - A3[i][j][k] | 0 < i < N and 0 < j < M and 0 < k < O
  - ADDR = B + (i * R) + (j * C) + (k * Z)

Row-Major	Column-Major
C = (M * S)	C = (1 * S)
R = (1 * S)	R = (N * S)

Problem Definition

- Population count or Hamming weight
  - Cryptography, Coding theory, Information theory

Given an integer, count the number of bits **ON** in its binary representation.

Examples:

Dec	Bin	Bits ON
0	00000000	0
10	00001010	2
105	01101001	4
1000	11111010	6

Approaches?

- Convert X into a string of 0s and 1s and use a loop to count the number of 1s contained in the string – **BAD!!!!**
- Check if the less significant bit is 1 and perform a right shifting – **Not good!**

```
While X != 0: do
    if LSB(X) == 1
        count = count + 1
    X = X >> 1
```

- There are some efficient algorithms to perform this task, but, **What about arrays?**

Solution

- Implement a look-up table with 256 slots(8 bits) that store the number of bits **ON** for each number from 0 to 255.

  uchar bset[256] = {0, 1, 1, 2, 1, 2, 2, 3, …};

- Given an integer **X**(32 bits integer) split it in 4 parts and calculate an index for each one.

H		L	
HH	**HL**	**LH**	**LL**
(X >> 24) & 0xFF	(X >> 16) & 0xFF	(X >> 8) & 0xFF	X & 0xFF

- Use these indexes to look-up into bset and a sum these 4 results.

  Ones = bset[HH] + bset[HL] + bset[LH] + bset[LL];

Problem Definition

- Where is the Marble?
  - https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1415

First Approach

- Sort the list of Marbles.

  For each element in the list of Queries, perform a search in Marbles, looking for the first occurrence of Qi.

Solution

- Create a vector of M elements and initialize each element with 0.
  - **T[M] = {0, 0, 0, …};**
- Read the list of N numbers and use each of them(Ni) as an index of T increasing the slot value in one.
  - **Loop 1:N do**

    T[Ni] = T[Ni] + 1;
- Calculate the accumulated sum:
  - **Loop 1:M do**

    T[i] = T[i] + T[i – 1];
- Read the list of Q queries and use each of them(Qi) as a index to get the first occurrence on T, If Qi is equal to Qi-1 then there is not occurrence.
  - **Loop 1:Q do**

    If T[Qi] == T[Qi-1] then

    Not found

    else

    Qi found in T[Qi] + 1

More Challenges

- <title1> <link1>
- <title2> <link2>
- ...

Use Cases

- Memory pool
- Linear/Binary search
- Trivial hash functions: Index mapping
- Image processing: Color look-up table(CLUT)
- Compute sines
- Jump table
- State Machines and DFAs
- Graphs

References

- http://eli.thegreenplace.net/2015/memory-layout-of-multi-dimensional-arrays/
- https://en.wikipedia.org/wiki/Array_data_structure
- https://en.wikipedia.org/wiki/Hamming_weight
- https://en.wikipedia.org/wiki/Hamming_distance