

# CSE 351 Section 6 Solutions – Arrays and Structs

Welcome back to section, we're happy that you're here ☺

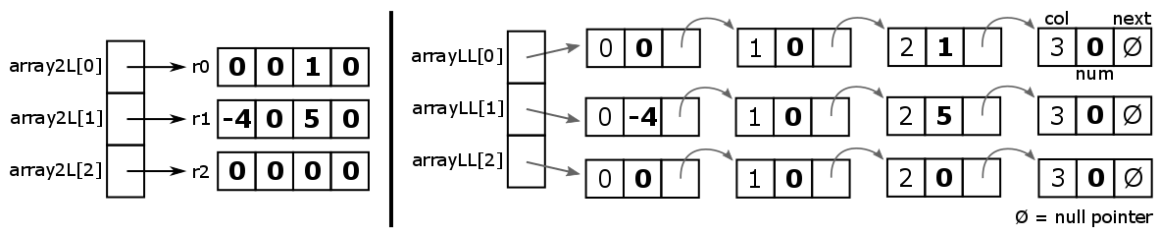
We have a two-dimensional matrix of integer data of size  $M$  rows and  $N$  columns. We are considering 3 different representation schemes:

- 1) 2-dimensional array `int array2D[][]`, //  $M*N$  array of ints
- 2) 2-level array `int* array2L[]`, and //  $M$  array of int arrays
- 3) array of linked lists `struct node* arrayLL[]`. //  $M$  array of linked lists (struct node)

Consider the case where  $M = 3$  and  $N = 4$ . The declarations are given below:

2-dimensional array:	2-level array:	Array of linked lists:
<code>int array2D[3][4];</code>	<code>int r0[4], r1[4], r2[4]; int* array2L[] = {r0,r1,r2};</code>	<code>struct node { int col, num; struct node* next; }; struct node* arrayLL[3]; // code to build out LLs</code>

For example, the diagrams below correspond to the matrix  $\begin{bmatrix} 0 & 0 & 1 & 0 \\ -4 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$  for `array2L` and `arrayLL`:



a) Fill in the following comparison chart:

	2-dim array	2-level array	Array of LLs:
Overall Memory Used	$M*N*\text{sizeof}(\text{int}) = 48 \text{ B}$	$M*N*\text{sizeof}(\text{int}) + M*\text{sizeof}(\text{int} *) = 72 \text{ B}$	$M*\text{sizeof}(\text{struct node} *) + M*N*\text{sizeof}(\text{struct node}) = 216 \text{ B}$
Largest <i>guaranteed</i> continuous chunk of memory	The whole array (48 B)	The array of pointers (24 B) > row array (16 B)	The array of pointers (24 B) > struct (16 B)
Smallest <i>guaranteed</i> continuous chunk of memory	The whole array (48 B)	Each row array (16 B)	Each struct node (16 B)
Data type returned by:	<code>array2D[1]</code> <code>int *</code>	<code>array2L[1]</code> <code>int *</code>	<code>arrayLL[1]</code> <code>struct node *</code>
Number of memory accesses to get <code>int</code> in the <i>BEST</i> case	1	2	First node in LL: 2
Number of memory accesses to get <code>int</code> in the <i>WORST</i> case	1	2	Last node in LL: 5 (we have to read next)

b) Sam Student claims that since our arrays are relatively small ( $N < 256$ ), we can save space by storing the `col` field as a `char` in `struct node`. Is this correct? If so, how much space do we save? If not, is this an example of internal or external fragmentation?

**No.** Alignment requirement of  $K = 4$  for `int num` leaves 3 bytes of internal fragmentation between `col` and `num`.