|  |  |  |  |
| --- | --- | --- | --- |
| Address | Type | Name | Value |
| 0x000A | char[3][2] | new2DArray | 0x000B |
| 0x000B | int | len | 3 |
| 0x000C | pointer |  | 0x000G |
| 0x000D | pointer |  | 0x000K |
| 0x000E | pointer |  | 0x000O |
| 0x000F |  |  |  |
| 0x000G | int | length | 2 |
| 0x000H | int |  | \*a value\* |
| 0x000I | int |  | \*a value\* |
| 0x000J |  |  |  |
| 0x000K | int | length | 2 |
| 0x000L | int |  | \*a value\* |
| 0x000M | int |  | \*a value\* |
| 0x000N |  |  |  |
| 0x000O | int | length | 2 |
| 0x000P | int |  | \*a value\* |
| 0x000Q | int |  | \*a value\* |
| 0x000R |  |  |  |
| 0x000S |  |  |  |
| 0x000T |  |  |  |

Above is an example showing how much storage space a 2D array of a particular size takes in our computers Main memory. For the 3X2 array above, 14 memory slots are filled. By drawing out other diagrams like this I saw:

|  |  |
| --- | --- |
| Array Dimensions | Number of Main Memory Slots Filled |
| 1X1 | 5 |
| 2X2 | 10 |
| 3X2 | 14 |
| 3X3 | 17 |
| 4X4 | 25 |

The amount of memory needed is not directly proportional to the increase in array dimensions. Rather, it grows at a slightly greater rate than the array dimensions grow.

**This leads me to choose a linear space complexity function my lower bound, Ω(n), and the polynomial space complexity function as my upper bound, O(n^2).**