BIG O notation

Arrays

Arrays are the simplest data structure and we use them to store list of items like a list of strings, list of integers, list of objects etc. These items are stored in sequence in memory.

Example:

Memory address: 100 104 108 112 ……………….so on

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array: | 27 | 7 | 44 | 12 | 54 |

Here we have an array with list of integers like 27, 7, 44, 12, 54 and their corresponding memory address for example starts with 100 and in java each integer value takes 4 bytes of memory so the next address will be 104, 108 and so on. Now because of this, the run time complexity of array for searching any item is **O(1)**. Here the calculation of memory address doesn’t take much time since no loops are involved or complex logic. So if we want to store items and access them by their index then arrays are the optimum ones.

In java and others Arrays are static, which means we have to specify the size of array while defining them. If we don’t the size upfront, then we have to guess. If we guess either low or more size then we will have either shortage of memory or more unused memory.

So for example, if we have an array with less size defined but we have more items to add and if we create a new array with bigger size and copy the previous array items to this new array, then the run time complexity will be linear which is O(n) (as the input grows so does the number insertions or copies to the new array linearly).

Now coming to the delete of an item from array, For the best case scenario if we have to delete an item from the last index then it is easy for array to check the index and delete. So best case scenario has run time complexity of O(1). But there is worst case scenario where we may have to delete an item from the first index, then the item has to be removed and all the rest of items indices has to be readjusted. So the run time complexity for the worst case scenario of deletion of item in array is O(n).

Sample code for deletion of an item from index 3 which is 40. Here we loop from the index=3 and shift the next to index item and same for the rest of items. So here if the index that we want to delete is in the beginning then we have to shift all the elements.

*int[] array = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};*

*int index=3;*

*for (int i = index; i < array.length - 1; i++) {*

*array[i] = array[i + 1];*

*}*

*Output: 10, 20, 30, 50, 60, 70, 80, 90, 100, 100*

If we observe here item value is removed but since array size is static the last value is still there and we have now duplicates. So the only way we have is to copy these items to a new array using a loop.

Alternatively we can use Arrays.copyOf(original array, new length) to copy to a new array.

*int[] array = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};*

*int [] array2 = Arrays.copyOf(array,9) // this is to remove last item*

*int [] array3 = Arrays.copyOf(array, 1); // to remove from start*

*System.out.println(Arrays.toString(array));*

So for lookup we use Arrays and for insert and delete we have to use Linked List.