# Programming Questions

first()

* 1. O(1)
  2. O(1)

last()

1. O(1)
2. O(1)

prev(p)

1. O(1)
2. O(1)

next(p)

1. O(1)
2. O(1)

set(p, e)

1. O(1)
2. O(1)

addFirst(e)

1. O(1)
2. O(n)

addLast(e)

1. O(1)
2. O(1)

addBefore(p, e)

1. O(1)
2. O(n)

addAfter(p, e)

1. O(1)
2. O(n)

delete(p)

1. O(1)
2. O(n)

swap(p1, p2)

1. O(1)
2. O(1)

truncate( )

1. O(1)
2. O(n)

SetExpansionRule(c)

1. O(1)
2. O(1)

expandArray()

1. O(1)
2. O(n)

The truncate method has a time complexity of but it can reduce your space complexity by up to 60%

Programming is a game of tradeoffs and a particular application might have very limited space available and it could thus require a truncate method to free up memory despite its time complexity.

In sum, it is advantageous to have the truncate method but should be avoided unless absolutely necessary.

1. .
2. .
3. .

In my own personal opinion that is mine and no one else’s, the doubling expansion rule is likely to provide better performance. The reason is because as the array gets larger, doubling its size increases the space in the array faster than adding 10, reducing the likelihood of calling the expandArray() function again which has a complexity of .

No. Using a linked list would not change the complexity of the methods, not considering the manipulation of the underlying array. When we don’t consider the manipulation of the underlying array, all the methods have a complexity of which is the same complexity of the methods using a linked list.

Yes. Using a linked list would change all the methods to methods. The reason is because linked lists eliminate the necessary shifting of all the elements of the array when an element is added or deleted.