Joaquín Mateos Barroso

	1	2	3	4	5	6	7	8	9	10	11	12
	Comparación Complejidad Arrays / Lists											
	front()	back()	get(i) (a. aleat.)	pushFront()	pushBack()	Insert()	popFront()	popBack()	remove()	Siguiente en orden	Recorrido en orden	Localizar
A Array	NA	NA	O(1)	na	na	na	na	na	na	O(n)	O(n log n)	O(n)
B DArray	NA	NA	O(1)	na	O(n)/(A)(1)	O(n)	na	O(1)	O(n)	O(n)	O(n log n)	O(n)
C List (DArray)	O(1)	NA		O(n)/(A)(1)	na	O(n)	O(1)	na	O(n)	O(n)	O(n log n)	O(n)
D List (LNode)*	O(1)	O(n)	O(n)	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)/(A)(1)	O(n)	O(n log n)	O(n)
E List (DLNode)	O(1)	O(n)	O(n)	O(1)	O(n)	O(1)	O(1)	O(n)	O(1)	O(n)	O(n log n)	O(n)
F List (circular)	O(1)	O(1)	O(n)	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	O(n)	O(n log n)	O(n)
G OrderedList**	O(1)	O(n)	O(n)	NA	NA	O(n)	O(1)	O(n)	O(1)	O(1)	O(n)	O(log n)

* Con cursor usando <prev_,cur_>

** Usando CList.

porque hay que avanzar todos los previos

Se ordena con qSort o mergeSort, y se recorre.
Así, O(n log n)+O(n) = O(n log n)