Order of Growth

Link: tinyurl
For inquiries, kindly contact the author via email.

Basic Concept

Time complexity is considered as the total number of steps operated. One command is one step.

Space complexity is considered as the total number of spaces occupied. One number/character is one space.

Time Complexity

`x = 1`	O(1)
`for i in range(n)`	O(n)

`for i in range(n):`
`    for j in range(n)`
	O(n^2)

`while i < n:`
`    i += 1`
	O(n)

`def f(x):`
`    return f(x - 1)`
	O(n)

`f(x) -> O(n)`
`for i in range(n):`
`    f(x)`
	O(n^2)

`str[1:]`	O(n)
`tpl[1:]`	O(n)
`list[1:]`	O(n)
`tpl1 + tpl2`	O(n)

`def f:`
`    return f + f`
	O(2^n)

Space Complexity

`x = 1`	O(1)
`str, tpl, list`	O(n)
`str[1:]`	O(n)
`tpl[1:]`	O(n)
`list[1:]`	O(n)
`((),)`	O(n^2)

`def f(x):`
`    return f(x - 1)`
	O(n)

`def f:`
`    return f + f`
	O(n)

Tree

Time: No. of leaves
Space: Depth of tree

Time Complexity of List Operation

len(lst)
O(1)

i in lst O(n), n = len(lst)

lst.append(x)
O(1)

lst.extend(lst1)
O(n), n = len(lst1)

lst.insert(i, x)
O(n), n = len(lst)

lst.remove(x)
O(n), n = len(lst)

lst.pop(i)/del lst[i] O(n), n = len(lst)

lst.pop()
O(1)

lst.clear()	O(1)
lst.index(x)	O(n), n = len(lst)
lst.count(x)	O(n), n = len(lst)
lst.sort()	O(nlogn), n = len(lst)
lst.copy()	O(n), n = len(lst)

Time Complexity of Dic Operation

k in dic	O(1)
dic[key]	O(1)
dic.get(key)	O(1)
dic.keys()/values()/items()	O(1)
dic.pop(key)/del dic[key]	O(1)
dic.update(dic1)	O(n), n = len(dic[1])
dic.clear()	O(1)
dic.copy()	O(n), n = len(dic)