ANALIZA ZŁOŻONOŚCI OBLICZENIOWEJ
ALGORYTM SORTOWANIA BĄBELKOWEGO

1. for (indeks i=n-1; i >= 1; zmniejszaj **i** o 1) {	Dla n = 7 i: 6, 5, 4, 3, 2, 1		n-1	c_1

2. for (indeks j = 0; j < i; zwiększaj j o 1) {
if (T[j] > T[j+1]) {
zamień miejscami T[j] i T[j+1]
}
}
}
	i
	6
	5
	4
	3
	2
	1
		j
		0,1,2,3,4,5
		0,1,2,3,4
		0,1,2,3
		0,1,2
		0,1
		0
			suma
			6
			5
			4
			3
			2
			1
				
				n-1
				n-2
				...
				3
				2
				1
					c_2

Pierwsza pętla: $(n-1)$

Druga pętla: $S = 1 + 2 + \cdots + (n-1) = \frac{1+(n-1)}{2} \cdot (n-1) = \frac{n \cdot n - n}{2} = \frac{1}{2}(n^2 - n)$

Suma ciągu arytmetycznego: $S_n = \frac{a_1 + a_n}{2} \cdot n$

$$T(n) = c_1(n-1) + c_2\left(\frac{1}{2}(n^2 - n)\right) = c_1 n - c_1 + \frac{c_2}{2}n^2 - \frac{c_2}{2}n = \frac{c_2}{2}n^2 + \left(c_1 - \frac{c_2}{2}\right)n - c_1 = an^2 + bn - c$$

Gdy **szacujemy złożoność obliczeniową** nie interesuje nas zazwyczaj dokładna złożoność algorytmu, ale *rząd wielkości złożoności*. W związku z powyższym do szacowania od góry złożoności algorytmów używamy notacji O (czyt. „O duże"), która upraszcza szacowania pozwalając *pominąć w trakcie obliczeń stałe i wolniej rosnące składniki wzoru*.

$T(n) = an^2 + bn - c = O(n^2)$

Algorithm	Time Complexity (Best)	Time Complexity (Average)	Time Complexity (Worst)	Space Complexity
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	O(1)
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	O(1)
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)