Time complexity for Bubble Sort.

$\Omega(n)$ - the array is already sorted

$O(n^2)$ - sorted in reverse order

$$T(n) = n + (n-1) + (n-2) + \dots + 2 + 1$$

$$= \frac{n(n+1)}{2}$$ (sum of natural numbers)

$$= O(n^2)$$ ✓

Time complexity for Counting Sort

1. Loop of lines 2-3 takes $O(k)$ time
2. Loop of lines 4-5 takes $O(n)$ time
3. Loop of lines 7-8 takes $O(k)$ time
4. Loop of lines 10-12 takes $O(n)$ time

$$T(n) = O(k) + O(n) + O(k) + O(n)$$

$$= O(k+n)$$ ✓

Time complexity for Radix Sort

$$T(n) = O(d(n+k))$$

where d is the maximum lenght of integers

eg. d for $10 = 2$, $213 = 3$, $10213 = 5$

and $(n+k)$ is the time to sort the d digits using counting sort.

Time complexity for Bucket Sort

Bucket sort performs at $O(n)$ on average, $\Theta(n)$

provided:

1. number of buckets n must be equal to the array being sorted
2. input array must be uniformly distributed across the length of bucket value.

If not, performance will be depending on the time running insertion sort (lines 7-8), which is $O(n^2)$

Time complexity for Shell Sort

Time complexity of Shell Sort depends heavily on gap sequence therefore the average performance is between $O(n)$ and $O(n^2)$

$$T(n) = O(n \log n)$$

Several types of increment sequence:

1. Shell : $n/2$, $n/4$, ... 1 (repeatedly divide by 2)
2. Hibbard : 1, 3, 7 .. $2^k - 1$
3. Knuth : 1, 4, 13, .. $(3^k - 1)/2$
4. Sedgewick : 1, 5, 19, 41, 109 ...