Part 2

Code:

```
bubbleSort(int arr[], int n)
{
 int temp;
 for(int i = 0; i < n; i++)
 {
  for(int j = 0; j < n-i-1; j++)
  {
   int swap = 0;
   if( arr[j] > arr[j+1])
   {
    temp = arr[j];
    arr[j] = arr[j+1];
    arr[j+1] = temp;
    swap = 1;
   }
  }
  if(swap==0)
  {
   break;
  }
 }
}
```

About Code:

We won't always need to swap elements, for certain iterations the elements being compared (within the array) may already be sorted correctly. Thus, for this case we can exit the nested loop rather than performing an unnecessary swap.

Time Complexity:

The first iteration will make n-1 comparisons, after which the next iteration will make n-1 comparisons and this process will continue.
We can represent this process as follows:

(n-1) + (n-2) + (n-3) + ... + 2 + 1
n(n-1)/2
n^2

We can see from our representation above that the time complexity is O(n^2).

Best/Worst/Average Case (Time Complexities):

Best Case: O(n) -> the array is already in order (no swaps required).
Worst Case: O(n^2) -> at least one swap is required in each iteration.
Average Case: O(n^2)