冒泡排序

最差时间复杂度
$O(n^2)$
最优时间复杂度
$O(n)$
平均时间复杂度
$O(n^2)$
最差空间复杂度
总共$O(n)$，需要辅助空间$O(1)$

冒泡排序（英语：**Bubble Sort**，台湾另外一种译名为：**泡沫排序**）是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。
冒泡排序对n个项目需要O(n^2)的比较次数，且可以原地排序。尽管这个算法是最简单了解和实现的排序算法之一，但它对于少数元素之外的数列排序是很没有效率的。
冒泡排序是与插入排序拥有相等的运行时间，但是两种算法在需要的交换次数却很大地不同。在最好的情况，冒泡排序需要$O(n^2)$次交换，而插入排序只要最多$O(n)$交换。冒泡排序的实现（类似下面）通常会对已经排序好的数列拙劣地运行（$O(n^2)$），而插入排序在这个例子只需要$O(n)$个运算。因此很多现代的算法教科书避免使用冒泡排序，而用插入排序替换之。冒泡排序如果能在内部循环第一次运行时，使用一个旗标来表示有无需要交换的可能，也可以把最好的复杂度降低到$O(n)$。在这个情况，已经排序好的数列就无交换的需要。若在每次走访数列时，把走访顺序反过来，也可以稍微地改进效率。有时候称为鸡尾酒排序，因为算法会从数列的一端到另一端之间穿梭往返。
冒泡排序算法的运作如下：

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。

4 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

由于它的简洁，冒泡排序通常被用来对于程序设计入门的学生介绍算法的概念。

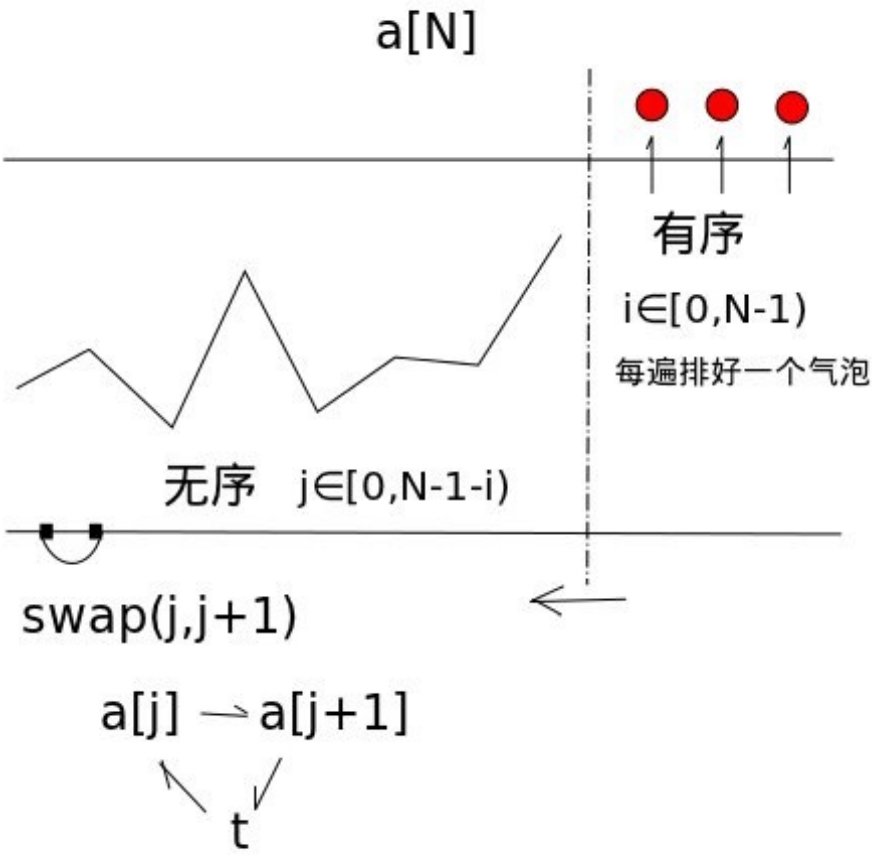
伪代码

```
function bubble_sort (array, length) {
    var i, j;
    for(i from 0 to length-1){
        for(j from 0 to length-1-i){
            if (array[j] > array[j+1])
                swap(array[j], array[j+1])
        }
    }
}
```

```
函數 冒泡排序 輸入 一個陣列名稱為array 其長度為length
    i 從 0 到 (length - 1)
        j 從 0 到 (length - 1 - i)
            如果 array[j] > array[j + 1]
                交換 array[j] 和 array[j + 1] 的值
            如果結束
        j迴圈結束
    i迴圈結束
函數結束
```

助记码

```
 i∈[0,N-1)                //循环N-1遍
   j∈[0,N-1-i)             //每遍循环要处理的无序部分
     swap(j,j+1)          //两两排序（升序/降序）
```



实现示例

C语言

```c
#include <stdio.h>
void bubble_sort(int arr[], int len) {
	int i, j, temp;
	for (i = 0; i < len - 1; i++)
		for (j = 0; j < len - 1 - i; j++)
			if (arr[j] > arr[j + 1]) {
				temp = arr[j];
				arr[j] = arr[j + 1];
				arr[j + 1] = temp;
			}
}
int main() {
	int arr[] = { 22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70 };
	int len = (int) sizeof(arr) / sizeof(*arr);
	bubble_sort(arr, len);
```

```
        int i;
        for (i = 0; i < len; i++)
                printf("%d ", arr[i]);
        return 0;
}
```

C++

```
#include <iostream>
#include <algorithm>
using namespace std;
template<typename T> //整數或浮點數皆可使用,若要使用物件(class)時必須設定大於(>)的運算子功能
void bubble_sort(T arr[], int len) {
        int i, j;
        for (i = 0; i < len - 1; i++)
                for (j = 0; j < len - 1 - i; j++)
                        if (arr[j] > arr[j + 1])
                                swap(arr[j], arr[j + 1]);
}
int main() {
        int arr[] = { 61, 17, 29, 22, 34, 60, 72, 21, 50, 1, 62 };
        int len = (int) sizeof(arr) / sizeof(*arr);
        bubble_sort(arr, len);
        for (int i = 0; i < len; i++)
                cout << arr[i] << ' ';
        cout << endl;
        float arrf[] = { 17.5, 19.1, 0.6, 1.9, 10.5, 12.4, 3.8, 19.7, 1.5, 25.4, 28.6, 4.4, 23.8, 5.4 };
        len = (int) sizeof(arrf) / sizeof(*arrf);
        bubble_sort(arrf, len);
        for (int i = 0; i < len; i++)
                cout << arrf[i] << ' ';
        return 0;
}
```

C#

```
static void BubbleSort(int[] intArray) {
    int temp = 0;//存储临时变量
    for (int i = 0; i < intArray.Length; i++)
        for (int j = i - 1; j >= 0; j--)
            if (intArray[j + 1] < intArray[j]) {
                temp = intArray[j + 1];
                intArray[j + 1] = intArray[j];
                intArray[j] = temp;
            }
}
```

JAVA

```
public class BubbleSort {
        public static void bubble_sort(int[] arr) {
                int i, j, temp, len = arr.length;
                for (i = 0; i < len - 1; i++)
                        for (j = 0; j < len - 1 - i; j++)
                                if (arr[j] > arr[j + 1]) {
                                        temp = arr[j];
                                        arr[j] = arr[j + 1];
                                        arr[j + 1] = temp;
                                }
        }
        public static void main(String[] args) {
                int[] number = {95,45,15,78,84,51,24,12};
                int temp = 0;
                bubble_sort(number);
                for(int i = 0; i < number.length; i++)
                        System.out.print(number[i] + " ");
                System.out.println();
        }
}
```

Ruby

```
class Array
  def bubble_sort!
```

```
    for i in 0...(size - 1)
      for j in 0...(size - i - 1)
        self[j], self[j + 1] = self[j + 1], self[j] if self[j] > self[j + 1]
      end
    end
    self
  end
end

puts [22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70].bubble_sort!
```

JavaScript

```
Array.prototype.bubble_sort = function() {
        var i, j, temp;
        for (i = 0; i < this.length - 1; i++)
                for (j = 0; j < this.length - 1 - i; j++)
                        if (this[j] > this[j + 1]) {
                                temp = this[j];
                                this[j] = this[j + 1];
                                this[j + 1] = temp;
                        }
        return this;
};
var num = [22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70];
num.bubble_sort();
for (var i = 0; i < num.length; i++)
        document.body.innerHTML += num[i] + " “;
```

Pascal

输入：（在程序同目录下的文本文件：input.txt）
一行：等待排序的数（用空格隔开）；
实例：194 638 124 482 469 245 852 294 484 243 623
输出：（在程序同目录下的文本文件：output.txt）
一行：已经排好的数（从小到大）；
实例：124 194 243 245 294 469 482 484 623 638 852

```
procedure swap(j:longint);  //交换過程
```

```
begin
  a[j]:=a[j] xor a[j+1];
  a[j+1]:=a[j] xor a[j+1];
  a[j]:=a[j] xor a[j+1];
end;

procedure bubble_sort;  //排序過程
var
  i,j:longint;
  flag:boolean;  //flag標誌：若一次排序未發現數據交換，則說明數據已經有序，可以結束排序過程
begin
  for i:=n-1 downto 1 do begin
    flag:=true;
    for j:=1 to i do begin
      if a[j]>a[j+1] then begin
        swap(j);
        flag:=false;
      end;
    end;
    if flag then exit;
  end;
end;
```

Python

```
def bubble(List):
    for j in range(len(List)-1,0,-1):
        for i in range(0,j):
            if List[i]>List[i+1]:List[i],List[i+1]=List[i+1],List[i]
    return List
```

示例：

```
testlist = [27, 33, 28, 4, 2, 26, 13, 35, 8, 14]
print('final:', bubble(testlist))
```

输出： final: ([2, 4, 8, 13, 14, 26, 27, 28, 33, 35])

VB.NET

```
'泡沫排序由大到小的程式，預先產生一儲存亂數內容的陣列B，使用中斷點check，
 switch 為自定兩數交換的sub

Dim i, j, count As Integer

 For i = 0 To UBound(b) – 1
    Dim check As Boolean = False    '進入排序後設定一布林變數令其初值為false

     For j = 0 To UBound(b) – 1 – i
        If b(j) < b(j + 1) Then switch(b(j), b(j + 1))

          check = True     '進行檢查程序，若符合交換條件即進行兩數值交換(呼叫sub
程序) 並於交換後
                           '將check的值變更為true(表示有進行交換動作，則此數列尚
未呈現最終排列序),
                            離開本層for迴圈後再度將check值重設成false
          count += 1
      Next

      If check = False Then Exit For '檢查進入迴圈後是否進行過數值交換，若
check值為false,
                                     '則表示排序進行到此時所有數列的值已呈現期望
中的順序,
                                      因此尚未進行完的排序檢查動作可提早結束以提
升效率。

       Next

 MsgBox("共經過了" & count & "次排序")
 '泡沫排序由小到大的程式

  Dim i, j, count As Integer
 Dim check As Boolean

  For i = 0 To UBound(b) – 1
       check=false
       For j = 0 To UBound(b) – 1 – i
          If b(j) > b(j + 1) Then switch(b(j), b(j + 1))
          count += 1
          check = True
```

```
        Next
        If chk = False Then Exit For
    Next

    MsgBox("共經過了" & count & "次的排序")

    '兩數值交換程式
    Private Sub switch(ByRef a as integer, ByRef b as integer)
        Dim c As Integer
        c = a
        a = b
        b = c
    End Sub
```

PHP

```
function swap(&$x, &$y) {
        $t = $x;
        $x = $y;
        $y = $t;
}

function bubble_sort(&$arr) {//php的陣列視為基本型別，所以必須用傳參考才能修改原陣列
        for ($i = 0; $i < count($arr) - 1; $i++)
                for ($j = 0; $j < count($arr) - 1 - $i; $j++)
                        if ($arr[$j] > $arr[$j + 1])
                                swap($arr[$j], $arr[$j + 1]);
}

$arr = array(21, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70);
bubble_sort($arr);
for ($i = 0; $i < count($arr); $i++)
        echo $arr[$i] . ' ';
```