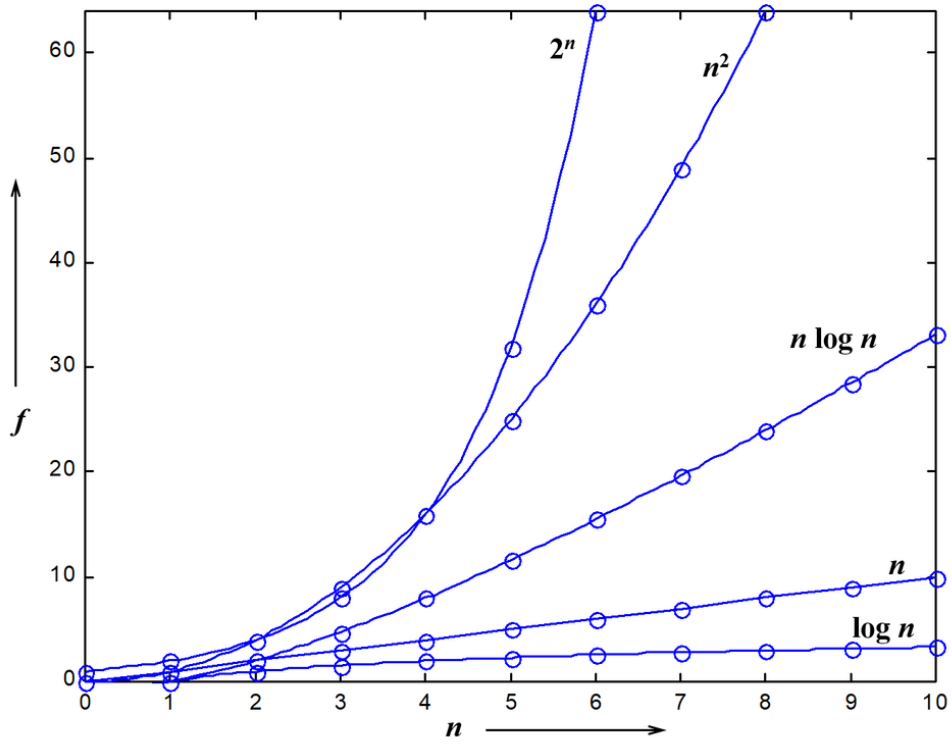
資料結構與C++進階班
排序

講師：黃銀鵬

E-mail: yinpenghuang@gmail.com

時間、空間複雜度

時間複雜度	表示式	n=100	n=10000	例子
常數複雜度	O(1)	1	1	隨機存取
對數複雜度	O(log(n))	2	4	二元數搜尋
線性複雜度	O(n)	100	10000	循序存取
對數線性複雜度	O(n•log(n))	200	40000	合併排序 快速排序
平方複雜度	O(n^2)	10000	10^8	氣泡排序
立方複雜度	O(n^3)	1000000	10^12	.....
指數複雜度	O(x^n) 以x=2為例	1.27*10^30	.....	費氏數列(遞迴)


60
50
40
30
20
10
0
f
0
1
2
3
4
5
6
7
8
9
10
n
2^n
n^2
n log n
n
log n

穩定排序法

- 不穩定排序演算法可能會在相等的鍵值中改變紀錄的相對次序，但是穩定排序演算法從來不會如此。(維基百科)
- 原始序列：(4,A);(3,B);(3,C);(5,D)
- 穩定排序：(3,B);(3,C);(4,A);(5,D)
- 非穩定排序：(3,C);(3,B);(4,A);(5,D)

內部、外部排序法

- 內部排序(Internal Sort) 資料筆數少，可以全部放到記憶體中排序。
- 外部排序(External Sort) 資料量大，無法放到記憶體中排序，需透過其它儲存裝置輔助。外部排序通常會分次載入部份的資料到記憶體，用內部排序演算法排序後再回存或合併結果

氣泡排序法

Round 1

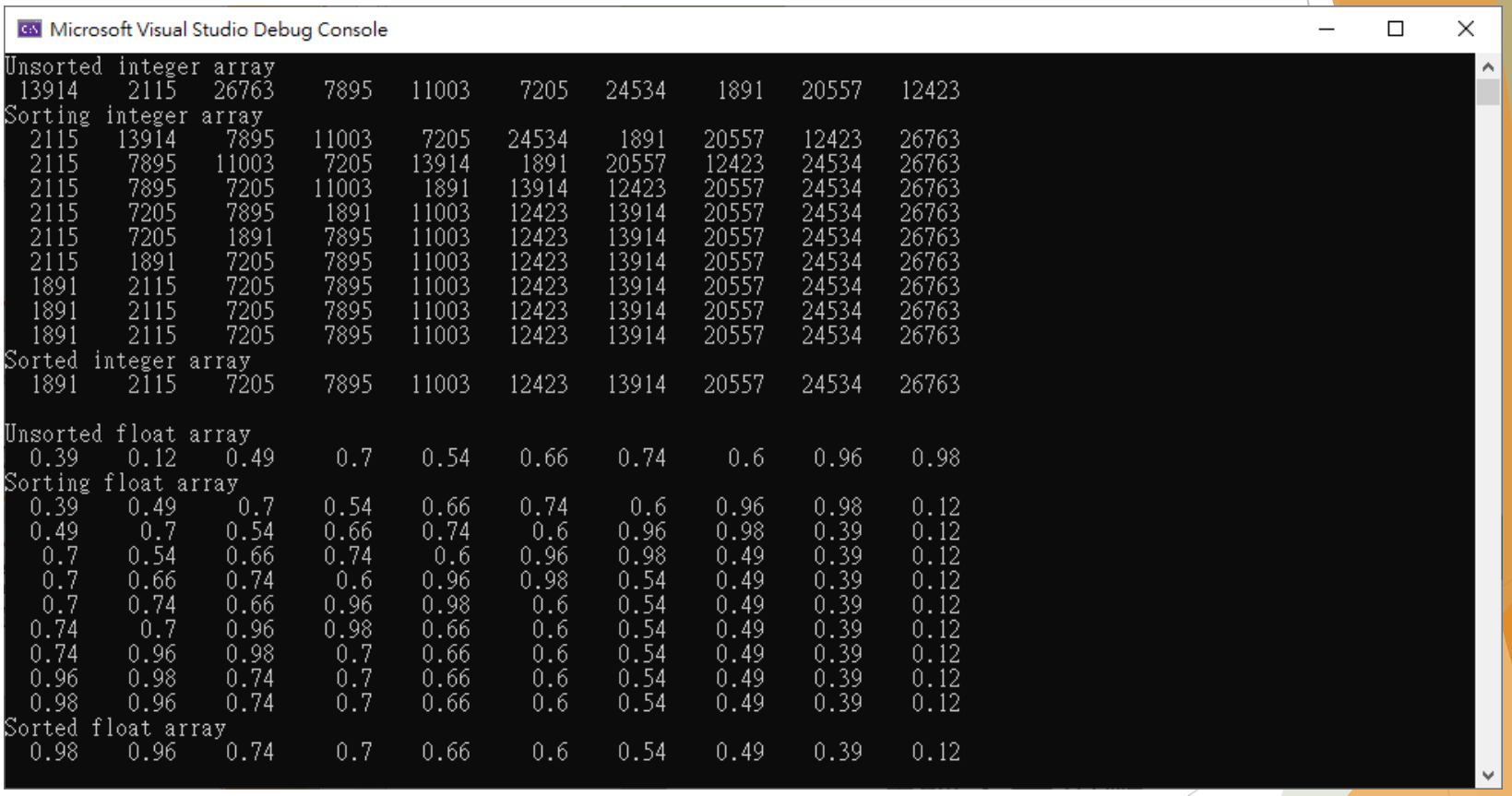
7	4	6	2
4	7	6	2
4	6	7	2
4	6	2	7

Round 2

4	6	2	7
4	6	2	7
4	2	6	7
4	2	6	7

Round 3

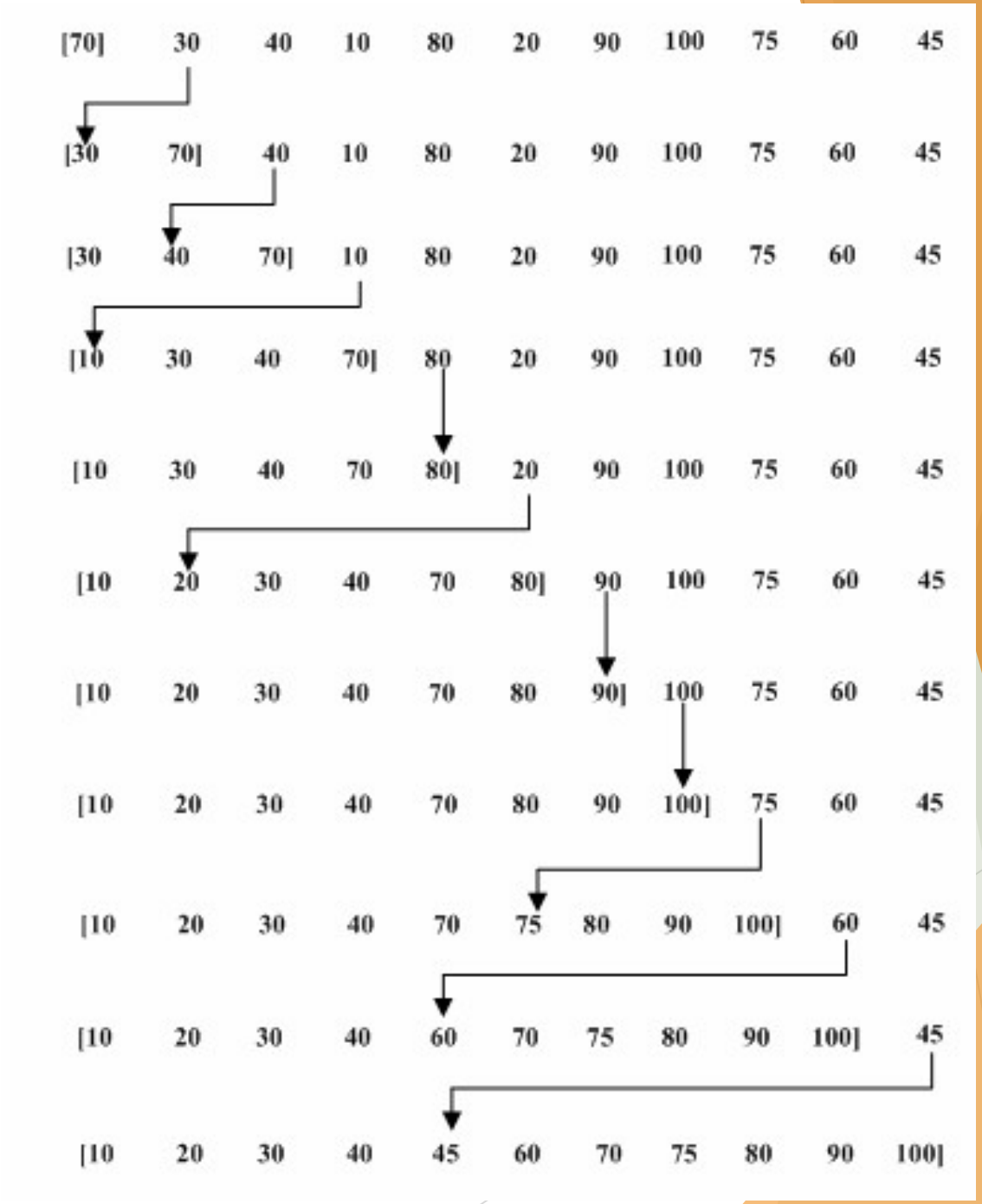
4	2	6	7
2	4	6	7
2	4	6	7
2	4	6	7

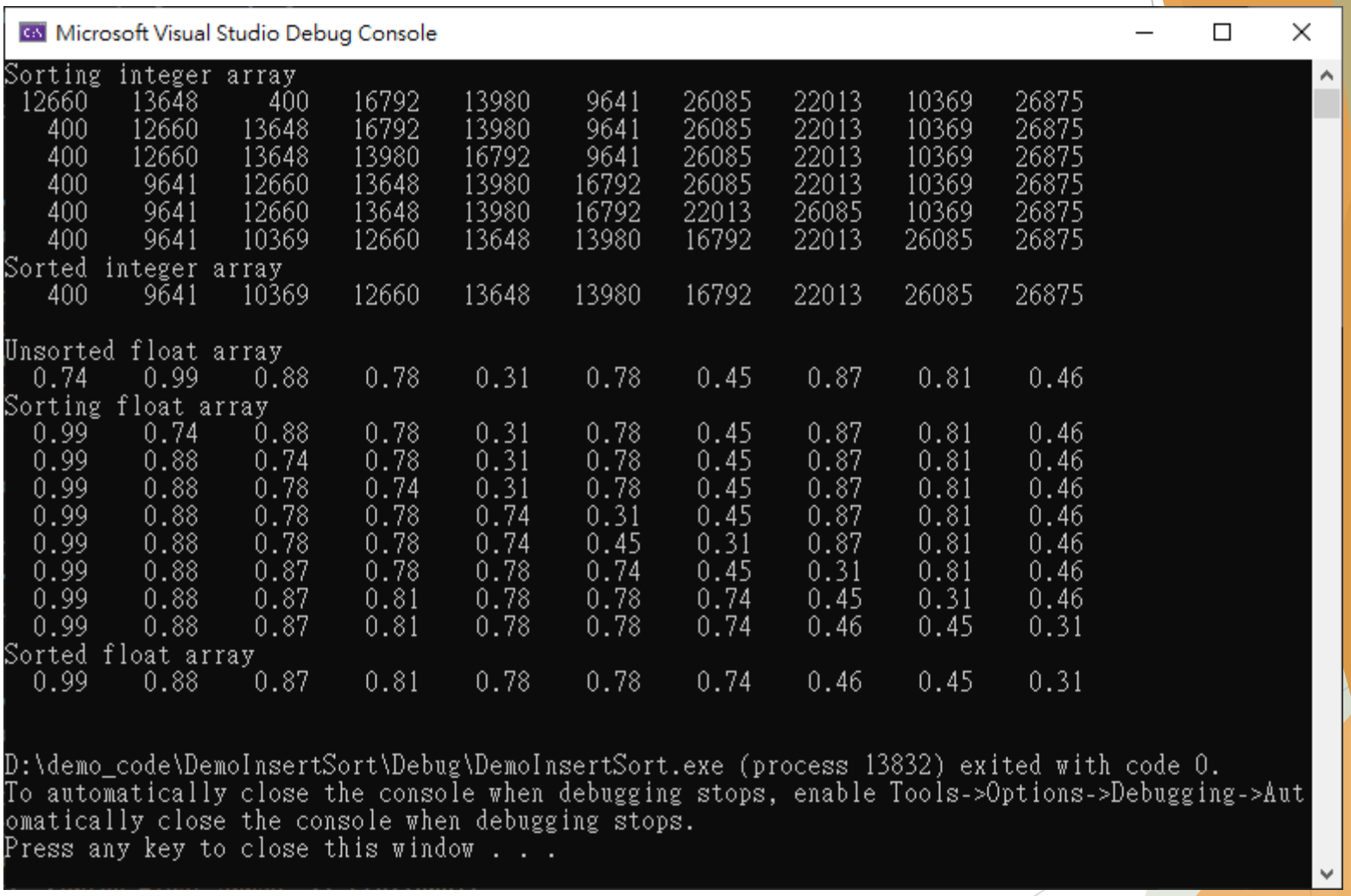

Microsoft Visual Studio Debug Console
Unsorted integer array
13914 2115 26763 7895 11003 7205 24534 1891 20557 12423
Sorting integer array
2115 13914 7895 11003 7205 24534 1891 20557 12423 26763
2115 7895 11003 7205 13914 1891 20557 12423 24534 26763
2115 7895 7205 11003 1891 13914 12423 20557 24534 26763
2115 7205 7895 1891 11003 12423 13914 20557 24534 26763
2115 7205 1891 7895 11003 12423 13914 20557 24534 26763
2115 1891 7205 7895 11003 12423 13914 20557 24534 26763
1891 2115 7205 7895 11003 12423 13914 20557 24534 26763
1891 2115 7205 7895 11003 12423 13914 20557 24534 26763
1891 2115 7205 7895 11003 12423 13914 20557 24534 26763
Sorted integer array
1891 2115 7205 7895 11003 12423 13914 20557 24534 26763
Unsorted float array
0.39 0.12 0.49 0.7 0.54 0.66 0.74 0.6 0.96 0.98
Sorting float array
0.39 0.49 0.7 0.54 0.66 0.74 0.6 0.96 0.98 0.12
0.49 0.7 0.54 0.66 0.74 0.6 0.96 0.98 0.39 0.12
0.7 0.54 0.66 0.74 0.6 0.96 0.98 0.49 0.39 0.12
0.7 0.66 0.74 0.6 0.96 0.98 0.54 0.49 0.39 0.12
0.7 0.74 0.66 0.96 0.98 0.6 0.54 0.49 0.39 0.12
0.74 0.7 0.96 0.98 0.66 0.6 0.54 0.49 0.39 0.12
0.74 0.96 0.98 0.7 0.66 0.6 0.54 0.49 0.39 0.12
0.96 0.98 0.74 0.7 0.66 0.6 0.54 0.49 0.39 0.12
0.98 0.96 0.74 0.7 0.66 0.6 0.54 0.49 0.39 0.12
Sorted float array
0.98 0.96 0.74 0.7 0.66 0.6 0.54 0.49 0.39 0.12

```cpp
#include <iostream>
#include <iomanip>
#include <time.h>
template<class T> void BubbleSort(T* array, unsigned int size, bool (*compareFun)(T a, T b));
template<class T> bool CompareBigger(T a, T b);
template<class T> bool CompareSmaller(T a, T b);
int main()
{
    int arraySize = 10;
    int* piArray = new int[arraySize];
    int i;
    srand((unsigned int)time(NULL));
    std::cout << "Unsorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        piArray[i] = rand();
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl;
    std::cout << "Sorting integer array" << std::endl;
    BubbleSort<int>(piArray, arraySize, CompareBigger<int>);
    std::cout << "Sorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
    float* pfArray = new float[arraySize];
    std::cout << "Unsorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        pfArray[i] = (float)rand() / RAND_MAX;
        std::cout << std::setprecision(2) << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl;
    std::cout << "Sorting float array" << std::endl;
    BubbleSort<float>(pfArray, arraySize, CompareSmaller<float>);
    std::cout << "Sorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
    delete[] piArray;
    delete[] pfArray;
    return 0;
}
```

```cpp
template<class T>
void BubbleSort(T* array, unsigned int size, bool(*compareFun)(T a, T b))
{
    unsigned int i, j, k;
    for (i = 0; i < size - 1; ++i)
    {
        for (j = 0; j < size - 1; ++j)
        {
            if (compareFun(array[j], array[j + 1]))
            {
                T reg = array[j];
                array[j] = array[j + 1];
                array[j + 1] = reg;
            }
        }
        for (k = 0; k < size; ++k)
        {
            std::cout << std::setw(6) << array[k] << "\t";
        }
        std::cout << std::endl;
    }
}
template<class T>
bool CompareBigger(T a, T b)
{
    return a > b;
}
template<class T>
bool CompareSmaller(T a, T b)
{
    return a < b;
}
```

插入排序法


[70] 30 40 10 80 20 90 100 75 60 45
[30 70] 40 10 80 20 90 100 75 60 45
[30 40 70] 10 80 20 90 100 75 60 45
[10 30 40 70] 80 20 90 100 75 60 45
[10 30 40 70 80] 20 90 100 75 60 45
[10 20 30 40 70 80] 90 100 75 60 45
[10 20 30 40 70 80 90] 100 75 60 45
[10 20 30 40 70 80 90 100] 75 60 45
[10 20 30 40 70 75 80 90 100] 60 45
[10 20 30 40 60 70 75 80 90 100] 45
[10 20 30 40 45 60 70 75 80 90 100]

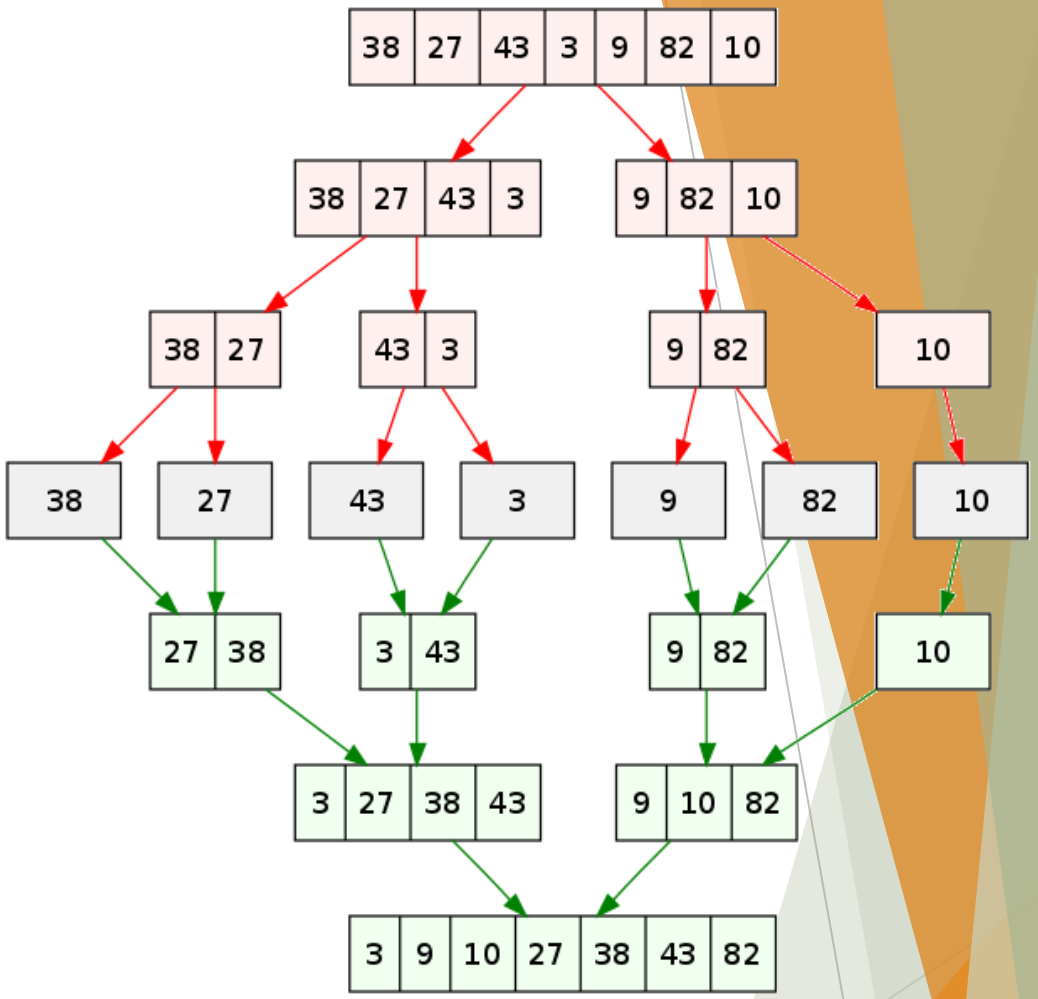

Microsoft Visual Studio Debug Console
Sorting integer array
12660 13648 400 16792 13980 9641 26085 22013 10369 26875
400 12660 13648 16792 13980 9641 26085 22013 10369 26875
400 12660 13648 13980 16792 9641 26085 22013 10369 26875
400 9641 12660 13648 13980 16792 26085 22013 10369 26875
400 9641 12660 13648 13980 16792 22013 26085 10369 26875
400 9641 10369 12660 13648 13980 16792 22013 26085 26875
Sorted integer array
400 9641 10369 12660 13648 13980 16792 22013 26085 26875
Unsorted float array
0.74 0.99 0.88 0.78 0.31 0.78 0.45 0.87 0.81 0.46
Sorting float array
0.99 0.74 0.88 0.78 0.31 0.78 0.45 0.87 0.81 0.46
0.99 0.88 0.74 0.78 0.31 0.78 0.45 0.87 0.81 0.46
0.99 0.88 0.78 0.74 0.31 0.78 0.45 0.87 0.81 0.46
0.99 0.88 0.78 0.78 0.74 0.31 0.45 0.87 0.81 0.46
0.99 0.88 0.78 0.78 0.74 0.45 0.31 0.87 0.81 0.46
0.99 0.88 0.87 0.78 0.78 0.74 0.45 0.31 0.81 0.46
0.99 0.88 0.87 0.81 0.78 0.78 0.74 0.45 0.31 0.46
0.99 0.88 0.87 0.81 0.78 0.78 0.74 0.46 0.45 0.31
Sorted float array
0.99 0.88 0.87 0.81 0.78 0.78 0.74 0.46 0.45 0.31
D:\demo_code\DemoInsertSort\Debug\DemoInsertSort.exe (process 13832) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Aut
omatically close the console when debugging stops.
Press any key to close this window . . .

```cpp
#include <iostream>
#include <iomanip>
#include <time.h>
template<class T> void InsertionSort(T* array, unsigned int size, bool (*compareFun)(T a, T b));
template<class T> bool CompareBigger(T a, T b);
template<class T> bool CompareSmaller(T a, T b);
int main()
{
    int arraySize = 10;
    int* piArray = new int[arraySize];
    int i;
    srand((unsigned int)time(NULL));
    std::cout << "Unsorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        piArray[i] = rand();
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl;
    std::cout << "Sorting integer array" << std::endl;
    InsertionSort<int>(piArray, arraySize, CompareBigger<int>);
    std::cout << "Sorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
```

```
    float* pfArray = new float[arraySize];
    std::cout << "Unsorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        pfArray[i] = (float)rand() / RAND_MAX;
        std::cout << std::setprecision(2) << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl;
    std::cout << "Sorting float array" << std::endl;
    InsertionSort<float>(pfArray, arraySize, CompareSmaller<float>);
    std::cout << "Sorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
    delete[] piArray;
    delete[] pfArray;
    return 0;
}
```

```cpp
template<class T>
void InsertionSort(T* array, unsigned int size, bool(*compareFun)(T a, T b))
{
    unsigned int i, j, k;
    unsigned int bound = 1;
    for (i = bound; i < size; ++i)
    {
        for (j = 0; j < bound; ++j)
        {
            if (compareFun(array[j], array[i]))
            {
                T reg = array[i];
                for (k = bound; k >= j + 1; --k)
                {
                    array[k] = array[k - 1];
                }
                array[j] = reg;
                for (k = 0; k < size; ++k)
                {
                    std::cout << std::setw(6) << array[k] << "\t";
                }
                std::cout << std::endl;
                break;
            }
        }
        bound++;
    }
}
template<class T>
bool CompareBigger(T a, T b)
{
    return a > b;
}
template<class T>
bool CompareSmaller(T a, T b)
{
    return a < b;
}
```

合併排序法



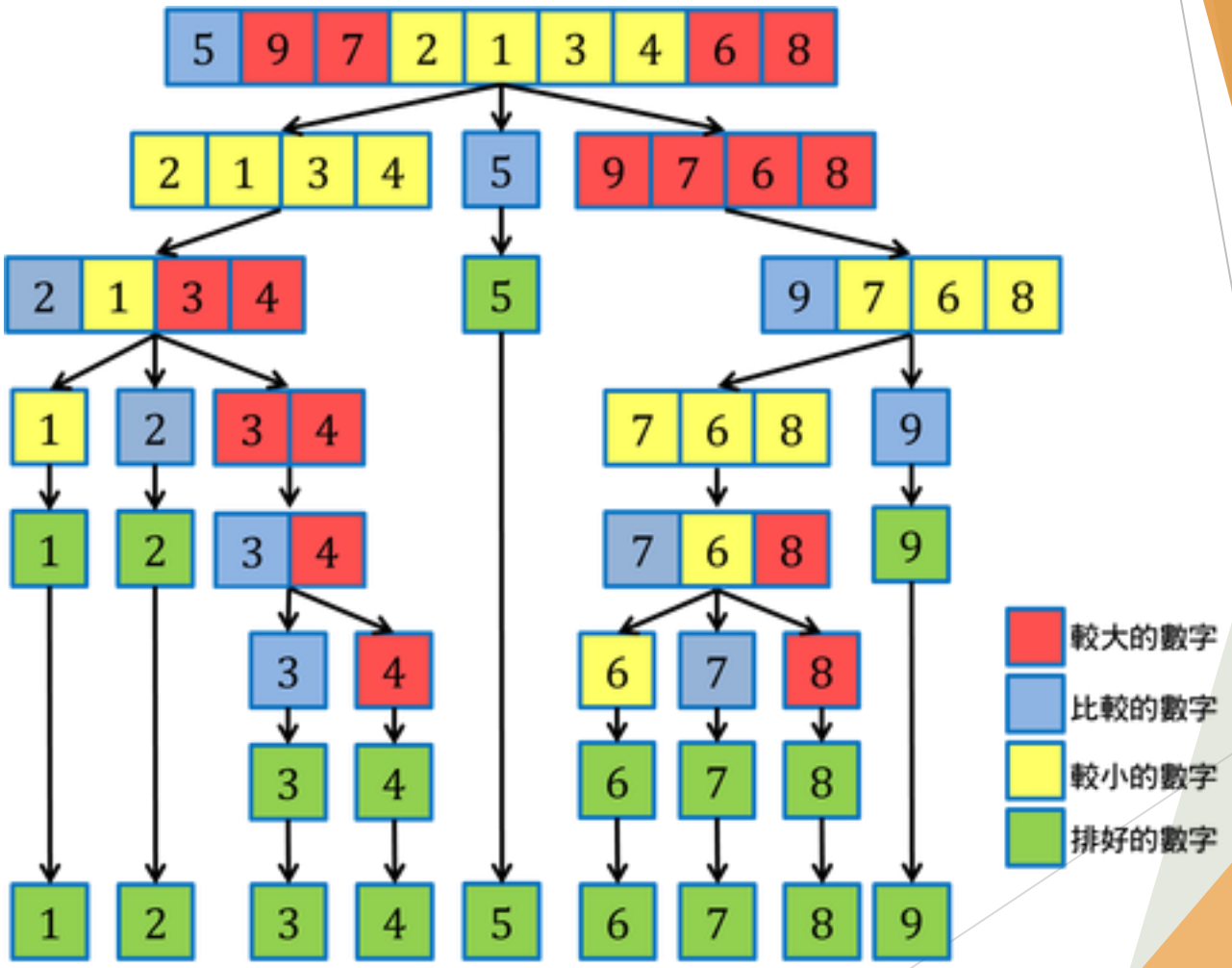

```
Unsorted integer array
 15846    4609   29899   24010    8873   30890   23612   18037   14318   17994
Sorted integer array
  4609    8873   14318   15846   17994   18037   23612   24010   29899   30890
Unsorted float array
   0.7     0.9    0.57    0.61   0.099    0.88    0.25    0.29    0.41    0.61
Sorted float array
   0.9    0.88     0.7    0.61    0.61    0.57    0.41    0.29    0.25   0.099
```

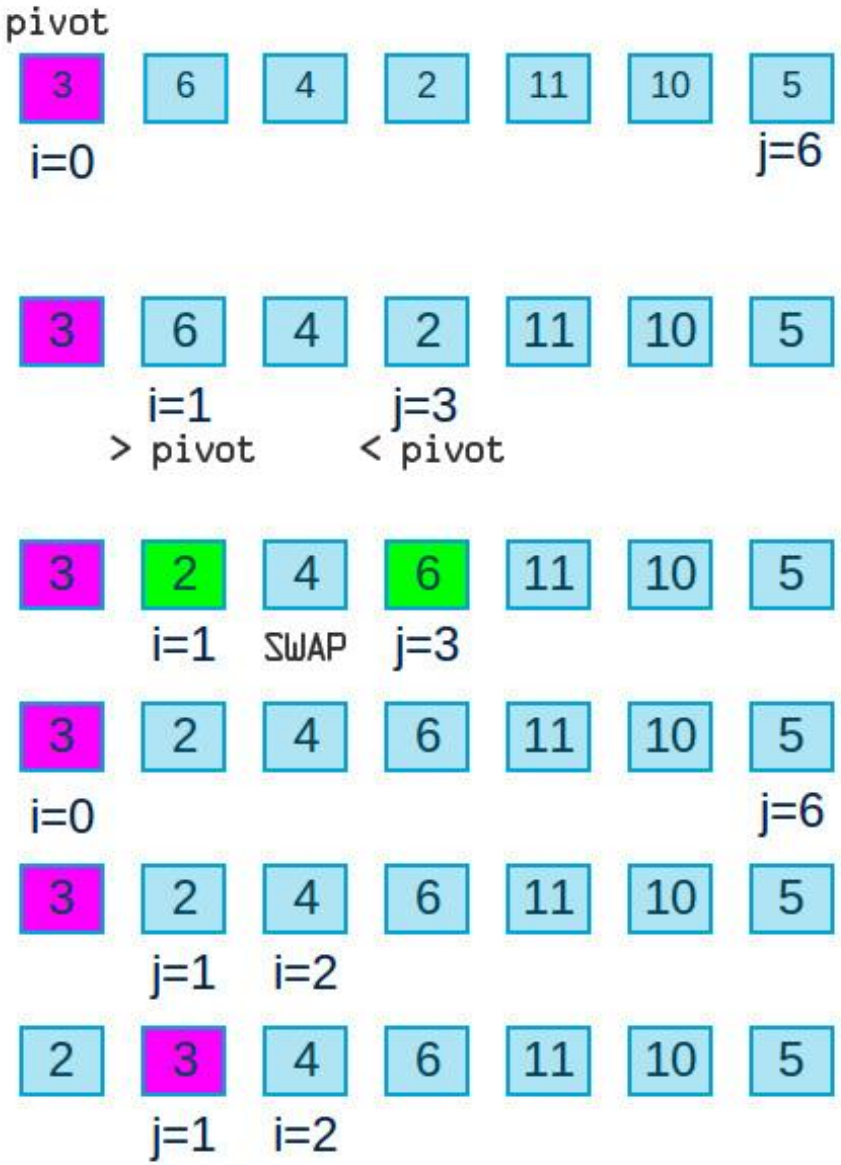
```cpp
#include <iostream>
#include <iomanip>
#include <time.h>
template<class T> void MargeSort(T* array, unsigned int size, bool(*compareFun)(T a, T b));
template<class T> void Marge(T* arrayA, unsigned int sizeA, T* arrayB, unsigned int sizeB, bool(*compareFun)(T a, T b));
template<class T> bool CompareBigger(T a, T b);
template<class T> bool CompareSmaller(T a, T b);
int main()
{
    int arraySize = 10;
    int* piArray = new int[arraySize];
    int i;
    srand((unsigned int)time(NULL));
    std::cout << "Unsorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        piArray[i] = rand();
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl;
    MargeSort<int>(piArray, arraySize, CompareBigger);
    std::cout << "Sorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
        std::cout << std::setw(6) << piArray[i] << "\t";
    std::cout << std::endl;
    float* pfArray = new float[arraySize];
    std::cout << "Unsorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        pfArray[i] = (float)rand() / RAND_MAX;
        std::cout << std::setprecision(2) << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl;
    MargeSort<float>(pfArray, arraySize, CompareSmaller<float>);
    std::cout << "Sorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
        std::cout << std::setw(6) << pfArray[i] << "\t";
    delete[] piArray;
    delete[] pfArray;
    return 0;
}
```

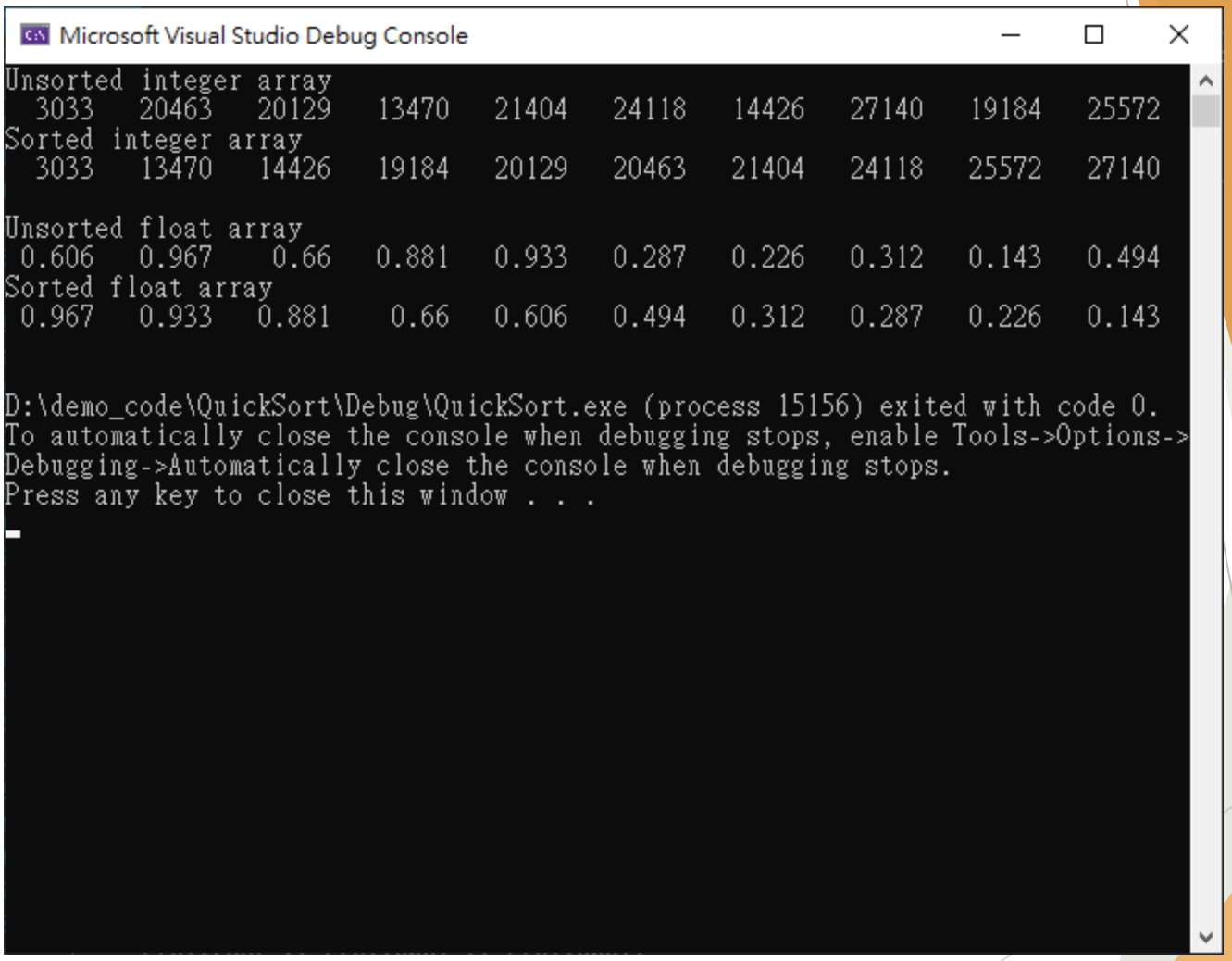
```cpp
template<class T>
void MargeSort(T* array, unsigned int size, bool(*compareFun)(T a, T b))
{
    if (size == 1) return;
    MargeSort<T>(array, size / 2, compareFun);
    MargeSort<T>(array + (size / 2), size - size / 2, compareFun);
    Marge<T>(array, size / 2, array + (size / 2), size - size / 2, compareFun);
}
template<class T>
void Marge(T* arrayA, unsigned int sizeA, T* arrayB, unsigned int sizeB, bool(*compareFun)(T a, T b))
{
    unsigned int totalSize = sizeA + sizeB;
    T* destArray = new T[totalSize];
    unsigned int x, i = 0, j = 0;
    for (x = 0; x < totalSize; ++x)
    {
        if (!compareFun(i < sizeA ? arrayA[i] : arrayB[j], j < sizeB ? arrayB[j] : arrayA[i]))
        {
            destArray[x] = i < sizeA ? arrayA[i] : arrayB[j];
            i < sizeA ? ++i : ++j;
        }
        else
        {
            destArray[x] = j < sizeB ? arrayB[j] : arrayA[i];
            j < sizeB ? ++j : ++i;
        }
    }
    memcpy(arrayA, destArray, sizeof(T) * totalSize);

    delete[] destArray;
}
template<class T>
bool CompareBigger(T a, T b)
{
    return a > b;
}
template<class T>
bool CompareSmaller(T a, T b)
{
    return a < b;
}
```

快速排序法


5 9 7 2 1 3 4 6 8
2 1 3 4
5
9 7 6 8
2 1 3 4
5
9 7 6 8
1
2
3 4
7 6 8
9
1
2
3 4
7 6 8
9
3
4
6
7
8
3
4
6
7
8
1
2
3
4
5
6
7
8
9
較大的數字
比較的數字
較小的數字
排好的數字


pivot
3 6 4 2 11 10 5
i=0
j=6
3 6 4 2 11 10 5
i=1
> pivot
j=3
< pivot
3 2 4 6 11 10 5
i=1
SWAP
j=3
3 2 4 6 11 10 5
i=0
j=6
3 2 4 6 11 10 5
j=1
i=2
2 3 4 6 11 10 5
j=1
i=2


Microsoft Visual Studio Debug Console
Unsorted integer array
3033 20463 20129 13470 21404 24118 14426 27140 19184 25572
Sorted integer array
3033 13470 14426 19184 20129 20463 21404 24118 25572 27140
Unsorted float array
0.606 0.967 0.66 0.881 0.933 0.287 0.226 0.312 0.143 0.494
Sorted float array
0.967 0.933 0.881 0.66 0.606 0.494 0.312 0.287 0.226 0.143
D:\demo_code\QuickSort\Debug\QuickSort.exe (process 15156) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->
Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```cpp
#include <iostream>
#include <iomanip>
#include <time.h>
template<class T> void QuickSort(T* array, unsigned int size, bool (*compareFun)(T a, T b));
template<class T> bool CompareBigger(T a, T b);
template<class T> bool CompareSmaller(T a, T b);
template<class T> void Swap(T& a, T& b);
int main()
{
    int arraySize = 10;
    int* piArray = new int[arraySize];
    int i;
    srand((unsigned int)time(NULL));
    std::cout << "Unsorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        piArray[i] = rand();
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl;
    QuickSort<int>(piArray, arraySize, CompareBigger<int>);
    std::cout << "Sorted integer array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << piArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
    float* pfArray = new float[arraySize];
    std::cout << "Unsorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        pfArray[i] = (float)rand() / RAND_MAX;
        std::cout << std::setprecision(3) << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl;
    QuickSort<float>(pfArray, arraySize, CompareSmaller<float>);
    std::cout << "Sorted float array" << std::endl;
    for (i = 0; i < arraySize; ++i)
    {
        std::cout << std::setw(6) << pfArray[i] << "\t";
    }
    std::cout << std::endl << std::endl;
    delete[] piArray;
    delete[] pfArray;
    return 0;
}
```

```cpp
template<class T>
void QuickSort(T* array, unsigned int size, bool(*compareFun)(T a, T b))
{
    if (size <= 1) return;
    else if (size == 2 && compareFun(array[0], array[1]))
    {
        Swap(array[0], array[1]);
        return;
    }
    unsigned int i = 0, j = size - 1;
    while (i < j)
    {
        while (compareFun(array[0], array[i]) && i < size - 1) ++i;
        while (!compareFun(array[0], array[j]) && j > 0) --j;
        if (i <= j) Swap(array[i], array[j]);
    }
    Swap(array[0], array[j]);
    QuickSort<T>(&array[0], j, compareFun);
    QuickSort<T>(&array[j + 1], size - j - 1, compareFun);
}
template<class T>
bool CompareBigger(T a, T b)
{
    return a >= b;
}
template<class T>
bool CompareSmaller(T a, T b)
{
    return a <= b;
}

template<class T>
void Swap(T& a, T& b)
{
    T c = a;
    a = b;
    b = c;
}
```

想想看

- 兩個演算法具有相同的時間複雜度，代表兩種演算法執行所花費的時間就相同嗎？