計算機演算法

作業一

班級：資訊三丙

學號：D0683497

姓名：柯利韋

1. 作法

依照 Bubble Sort、Selection Sort、Insertion Sort、Quick Sort、Heap Sort 的順序去做，每個排序法分別有一個 class，class 中的 Sort 方法為排序法的主程式、StartTest\_排序法 會產生亂數、排序、計時，完成後會呼叫寫 excel 檔的 function。

1. 程式碼

using System;

namespace HW01

{

    class Program

    {

        static void Main(string[] args)

        {

            foreach (var name in Config.SortList)

            {

                switch (name)

                {

                    case "Bubble Sort":

                        Console.WriteLine($"開始 {name}");

                        BubbleSort.StartTestBubbleSort();

                        break;

                    case "Selection Sort":

                        Console.WriteLine($"\n\n開始 {name}");

                        SelectionSort.StartTestSelectionSort();

                        break;

                    case "Insertion Sort":

                        Console.WriteLine($"\n\n開始 {name}");

                        InsertionSort.StartTestInsertionSort();

                        break;

                    case "Quick Sort":

                        Console.WriteLine($"\n\n開始 {name}");

                        QuickSort.StartTestQuickSort();

                        break;

                    case "Heap Sort":

                        Console.WriteLine($"\n\n開始 {name}");

                        HeapSort.StartTestHeapSort();

                        break;

                    default:

                        break;

                }

            }

            Console.WriteLine("請按任意鍵退出");

            Console.ReadLine();

        }

    }

}

using System.Collections.Generic;

using System.IO;

namespace HW01

{

    public class Config

    {

        public static List<string> SortList = new List<string> { "Bubble Sort", "Selection Sort", "Insertion Sort", "Quick Sort", "Heap Sort" };

        public static List<int> SeedList = new List<int> { 50000, 100000, 150000, 200000, 250000, 300000 };

        public static int Round = 25;

        public static string FileName = Path.Combine(Directory.GetCurrentDirectory(), "D0683497\_HW01.xlsx");

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Spreadsheet;

namespace HW01

{

    public class Heaplers

    {

        public static List<int> GenderateRadomNumbers(int seed)

        {

            Random rand = new Random(Guid.NewGuid().GetHashCode());

            List<int> randList = new List<int>(Enumerable.Range(1, seed));

            return randList.OrderBy(o => rand.Next()).ToList<int>();

        }

        public static void Swap<T>(IList<T> list, int indexA, int indexB)

        {

            T tmp = list[indexA];

            list[indexA] = list[indexB];

            list[indexB] = tmp;

        }

        public class Record

        {

            public string SortName { get; set; }

            public string Seed { get; set; }

            public string Round { get; set; }

            public string Time { get; set; }

        }

        public static void WriteExcel(List<Record> records)

        {

            string sortName = records[0].SortName;

            if (sortName == "Bubble Sort")

            {

                CreateExcel(records);

            }

            else

            {

                EditExcel(records);

            }

        }

        private static void CreateExcel(List<Record> records)

        {

            using (SpreadsheetDocument document = SpreadsheetDocument.Create(Config.FileName, SpreadsheetDocumentType.Workbook))

            {

                WorkbookPart workbookPart = document.AddWorkbookPart();

                workbookPart.Workbook = new Workbook();

                WorksheetPart worksheetPart = workbookPart.AddNewPart<WorksheetPart>();

                worksheetPart.Worksheet = new Worksheet();

                Sheets sheets = workbookPart.Workbook.AppendChild(new Sheets());

                Sheet sheet = new Sheet() { Id = workbookPart.GetIdOfPart(worksheetPart), SheetId = 1, Name = records[0].SortName };

                sheets.Append(sheet);

                workbookPart.Workbook.Save();

                SheetData sheetData = worksheetPart.Worksheet.AppendChild(new SheetData());

                #region Header

                Row row = new Row();

                row.Append(

                    new Cell() { CellValue = new CellValue("演算法名稱"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("亂數總數"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("回合數"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("時間(秒)"), DataType = CellValues.String }

                );

                sheetData.AppendChild(row);

                #endregion

                #region Content

                foreach (var record in records)

                {

                    row = new Row();

                    row.Append(new Cell() { CellValue = new CellValue(record.SortName), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Seed), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Round), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Time), DataType = CellValues.String });

                    sheetData.AppendChild(row);

                }

                #endregion

                worksheetPart.Worksheet.Save();

            }

        }

        private static void EditExcel(List<Record> records)

        {

            using (SpreadsheetDocument document = SpreadsheetDocument.Open(Config.FileName, true))

            {

                WorksheetPart newWorksheetPart = document.WorkbookPart.AddNewPart<WorksheetPart>();

                newWorksheetPart.Worksheet = new Worksheet();

                Sheets sheets = document.WorkbookPart.Workbook.GetFirstChild<Sheets>();

                string relationshipId = document.WorkbookPart.GetIdOfPart(newWorksheetPart);

                uint sheetId = 1;

                if (sheets.Elements<Sheet>().Count() > 0)

                {

                    sheetId = sheets.Elements<Sheet>().Select(s => s.SheetId.Value).Max() + 1;

                }

                Sheet sheet = new Sheet() { Id = relationshipId, SheetId = sheetId, Name = records[0].SortName };

                sheets.Append(sheet);

                newWorksheetPart.Worksheet.Save();

                SheetData sheetData = newWorksheetPart.Worksheet.AppendChild(new SheetData());

                #region Header

                Row row = new Row();

                row.Append(

                    new Cell() { CellValue = new CellValue("演算法名稱"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("亂數總數"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("回合數"), DataType = CellValues.String },

                    new Cell() { CellValue = new CellValue("時間(秒)"), DataType = CellValues.Number }

                );

                sheetData.AppendChild(row);

                #endregion

                #region Content

                foreach (var record in records)

                {

                    row = new Row();

                    row.Append(new Cell() { CellValue = new CellValue(record.SortName), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Seed), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Round), DataType = CellValues.String });

                    row.Append(new Cell() { CellValue = new CellValue(record.Time), DataType = CellValues.String });

                    sheetData.AppendChild(row);

                }

                #endregion

                newWorksheetPart.Worksheet.Save();

            }

        }

    }

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace HW01

{

    public class BubbleSort

    {

        #region 排序法

        public static void Sort(List<int> randList)

        {

            int lenght = randList.Count;

            for (int i = 1; i <= lenght - 1; i++)

            {

                for (int j = 1; j <= lenght - i; j++)

                {

                    if (randList[j] < randList[j - 1])

                    {

                        Heaplers.Swap(randList, j, j - 1);

                    }

                }

            }

        }

        #endregion

        public static void StartTestBubbleSort()

        {

            Stopwatch sw = new Stopwatch();

            List<Heaplers.Record> result = new List<Heaplers.Record>();

            foreach (int seed in Config.SeedList)

            {

                for (int i = 1; i <= Config.Round; i++)

                {

                    List<int> randList = Heaplers.GenderateRadomNumbers(seed);

                    sw.Reset();

                    sw.Start();

                    Sort(randList);

                    sw.Stop();

                    result.Add(new Heaplers.Record

                    {

                        SortName = "Bubble Sort",

                        Seed = seed.ToString(),

                        Round = i.ToString(),

                        Time = sw.Elapsed.TotalSeconds.ToString()

                    });

                    Console.WriteLine($"({seed}) 第 {i} 回合 花費時間 {sw.Elapsed.TotalSeconds.ToString()} 秒");

                }

            }

            Console.WriteLine("Bubble Sort 結束");

            Console.WriteLine("開始將 Bubble Sort 結果寫入檔案");

            Heaplers.WriteExcel(result);

            Console.WriteLine("寫檔完畢");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace HW01

{

    public class InsertionSort

    {

        #region 排序法

        public static void Sort(List<int> randList)

        {

            int lenght = randList.Count;

            for (int i = 0; i < lenght - 1; i++)

            {

                for (int j = i + 1; j > 0; j--)

                {

                    if (randList[j - 1] > randList[j])

                    {

                        Heaplers.Swap(randList, j - 1, j);

                    }

                }

            }

        }

        #endregion

        public static void StartTestInsertionSort()

        {

            Stopwatch sw = new Stopwatch();

            List<Heaplers.Record> result = new List<Heaplers.Record>();

            foreach (int seed in Config.SeedList)

            {

                for (int i = 1; i <= Config.Round; i++)

                {

                    List<int> randList = Heaplers.GenderateRadomNumbers(seed);

                    sw.Reset();

                    sw.Start();

                    Sort(randList);

                    sw.Stop();

                    result.Add(new Heaplers.Record

                    {

                        SortName = "Insertion Sort",

                        Seed = seed.ToString(),

                        Round = i.ToString(),

                        Time = sw.Elapsed.TotalSeconds.ToString()

                    });

                    Console.WriteLine($"({seed}) 第 {i} 回合 花費時間 {sw.Elapsed.TotalSeconds.ToString()} 秒");

                }

            }

            Console.WriteLine("Insertion Sort 結束");

            Console.WriteLine("開始將 Insertion Sort 結果寫入檔案");

            Heaplers.WriteExcel(result);

            Console.WriteLine("寫檔完畢");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace HW01

{

    public class SelectionSort

    {

        #region 排序法

        public static void Sort(List<int> randList)

        {

            int lenght = randList.Count;

            int smallest;

            for (int i = 0; i < lenght - 1; i++)

            {

                smallest = i;

                for (int j = i + 1; j < lenght; j++)

                {

                    if (randList[j] < randList[smallest])

                    {

                        smallest = j;

                    }

                }

                Heaplers.Swap(randList, smallest, i);

            }

        }

        #endregion

        public static void StartTestSelectionSort()

        {

            Stopwatch sw = new Stopwatch();

            List<Heaplers.Record> result = new List<Heaplers.Record>();

            foreach (int seed in Config.SeedList)

            {

                for (int i = 1; i <= Config.Round; i++)

                {

                    List<int> randList = Heaplers.GenderateRadomNumbers(seed);

                    sw.Reset();

                    sw.Start();

                    Sort(randList);

                    sw.Stop();

                    result.Add(new Heaplers.Record

                    {

                        SortName = "Selection Sort",

                        Seed = seed,

                        Round = i,

                        Time = sw.Elapsed.TotalSeconds

                    });

                    Console.WriteLine($"({seed}) 第 {i} 回合 花費時間 {sw.Elapsed.TotalSeconds.ToString()} 秒");

                }

            }

            Console.WriteLine("Selection Sort 結束");

            Console.WriteLine("開始將 Selection Sort 結果寫入檔案");

            Heaplers.WriteExcel(result);

            Console.WriteLine("寫檔完畢");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace HW01

{

    public class QuickSort

    {

        #region 排序法

        // https://dotblogs.com.tw/kennyshu/2009/10/24/11270

        public static void Sort(List<int> randList, int left, int right)

        {

            if (left < right)

            {

                int i = left;

                int j = right + 1;

                while (true)

                {

                    while (i + 1 < randList.Count && randList[++i] < randList[left]) ;

                    while (j - 1 > -1 && randList[--j] > randList[left]) ;

                    if (i >= j)

                    {

                        break;

                    }

                    Heaplers.Swap(randList, i, j);

                }

                Heaplers.Swap(randList, left, j);

                Sort(randList, left, j - 1);

                Sort(randList, j + 1, right);

            }

        }

        #endregion

        public static void StartTestQuickSort()

        {

            Stopwatch sw = new Stopwatch();

            List<Heaplers.Record> result = new List<Heaplers.Record>();

            foreach (int seed in Config.SeedList)

            {

                for (int i = 1; i <= Config.Round; i++)

                {

                    List<int> randList = Heaplers.GenderateRadomNumbers(seed);

                    sw.Reset();

                    sw.Start();

                    Sort(randList, 0, randList.Count - 1);

                    sw.Stop();

                    result.Add(new Heaplers.Record

                    {

                        SortName = "Quick Sort",

                        Seed = seed.ToString(),

                        Round = i.ToString(),

                        Time = sw.Elapsed.TotalSeconds.ToString()

                    });

                    Console.WriteLine($"({seed}) 第 {i} 回合 花費時間 {sw.Elapsed.TotalSeconds.ToString()} 秒");

                }

            }

            Console.WriteLine("Quick Sort 結束");

            Console.WriteLine("開始將 Quick Sort 結果寫入檔案");

            Heaplers.WriteExcel(result);

            Console.WriteLine("寫檔完畢");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace HW01

{

    public class HeapSort

    {

        #region 排序法

        public static void Sort(List<int> randList)

        {

            int length = randList.Count;

            for (int i = length / 2 - 1; i >= 0; i--)

            {

                Heapify(randList, length, i);

            }

            for (int i = length - 1; i >= 0; i--)

            {

                Heaplers.Swap(randList, 0, i);

                Heapify(randList, i, 0);

            }

        }

        private static void Heapify(List<int> list, int length, int i)

        {

            int largest = i;

            int left = 2 \* i + 1;

            int right = 2 \* i + 2;

            if (left < length && list[left] > list[largest])

            {

                largest = left;

            }

            if (right < length && list[right] > list[largest])

            {

                largest = right;

            }

            if (largest != i)

            {

                Heaplers.Swap(list, i, largest);

                Heapify(list, length, largest);

            }

        }

        #endregion

        public static void StartTestHeapSort()

        {

            Stopwatch sw = new Stopwatch();

            List<Heaplers.Record> result = new List<Heaplers.Record>();

            foreach (int seed in Config.SeedList)

            {

                for (int i = 1; i <= Config.Round; i++)

                {

                    List<int> randList = Heaplers.GenderateRadomNumbers(seed);

                    sw.Reset();

                    sw.Start();

                    Sort(randList);

                    sw.Stop();

                    result.Add(new Heaplers.Record

                    {

                        SortName = "Heap Sort",

                        Seed = seed.ToString(),

                        Round = i.ToString(),

                        Time = sw.Elapsed.TotalSeconds.ToString()

                    });

                    Console.WriteLine($"({seed}) 第 {i} 回合 花費時間 {sw.Elapsed.TotalSeconds.ToString()} 秒");

                }

            }

            Console.WriteLine("Heap Sort 結束");

            Console.WriteLine("開始將 Heap Sort 結果寫入檔案");

            Heaplers.WriteExcel(result);

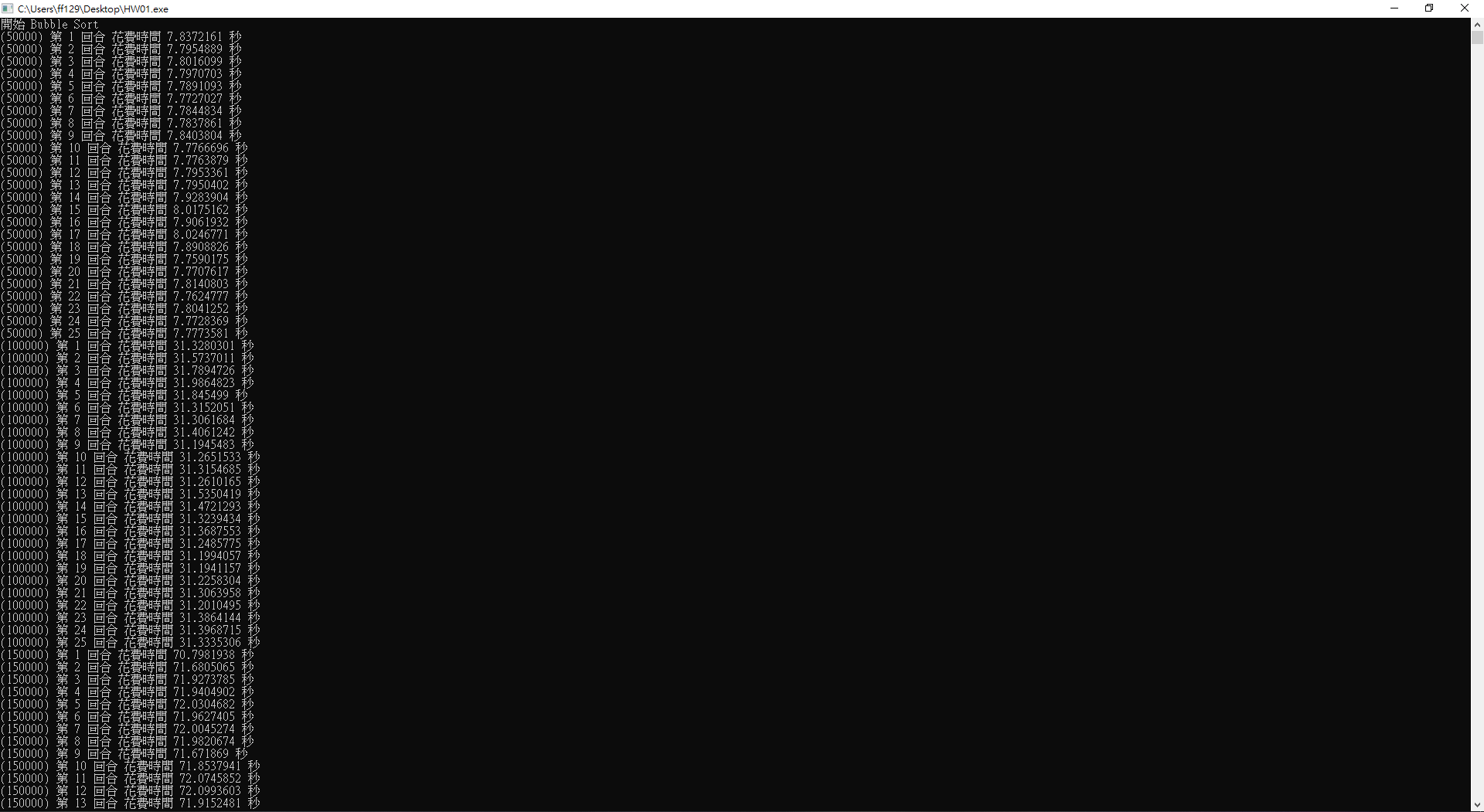
            Console.WriteLine("寫檔完畢");

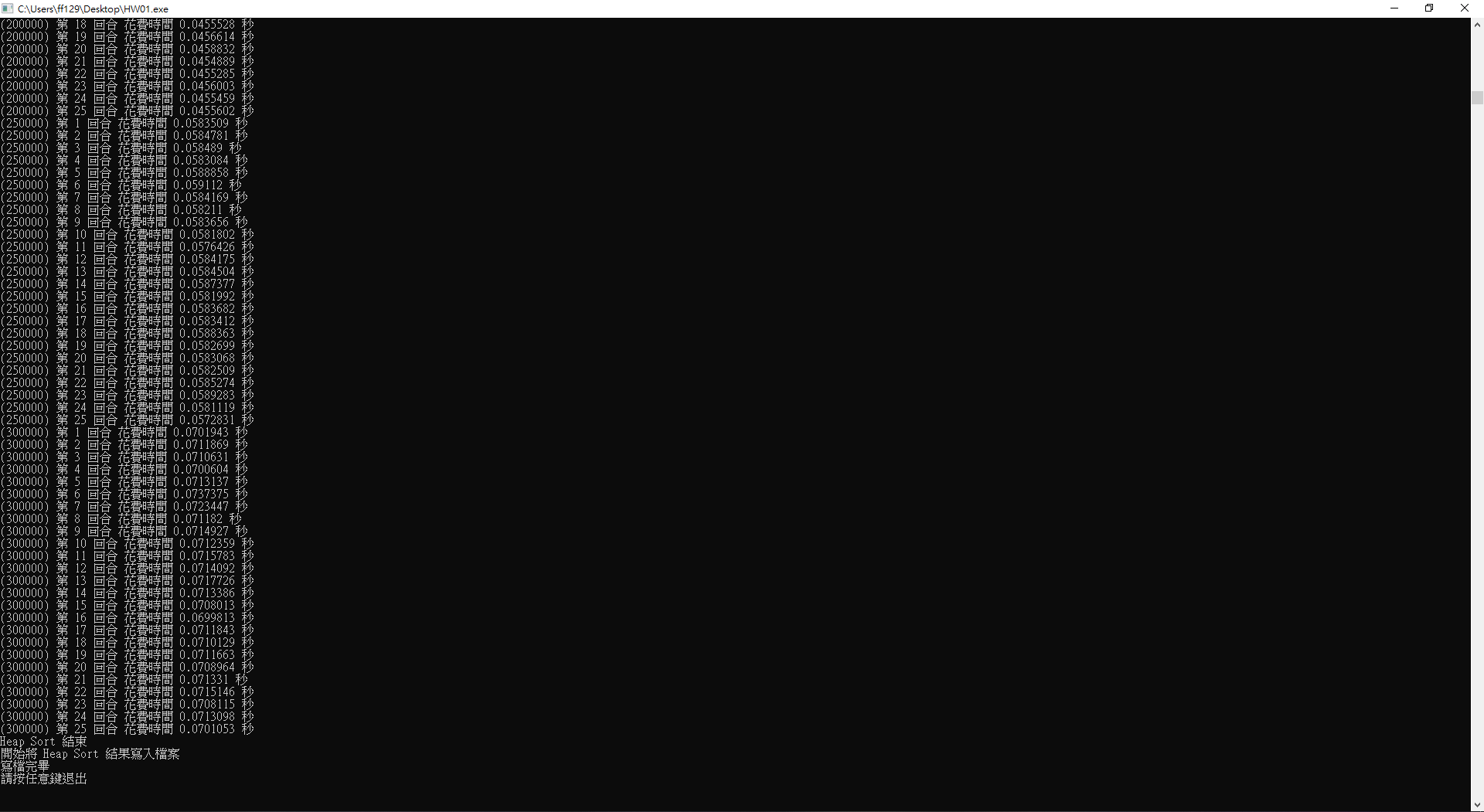
        }

    }

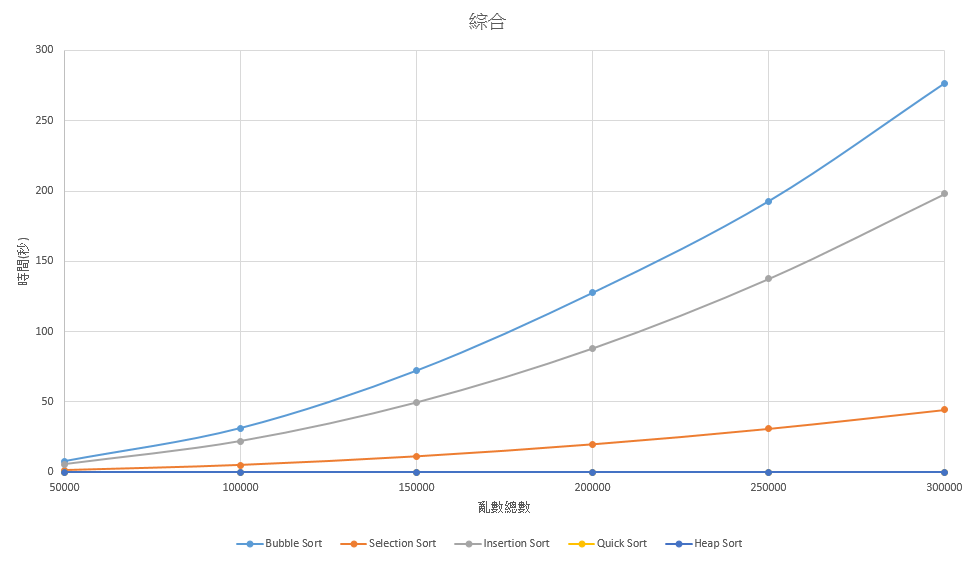
}

1. 執行結果
   1. 程式執行過程





* 1. 圖表結果



1. 討論

從結果來看 Quick Sort、Heap Sort 最快，全部 case 都在 0.1秒內就完成了。

1. 心得

基本上過程中沒有遇到太大的困難，只是因為我不想把結果一個一個貼到 Excel 中，所以我就直接把結果寫到 Excel，寫的過程中讀檔的部分，因為套件的官方文件沒有寫的很清楚所以研究了一下。