分析Sorting時間

計算機演算法 作業1

姓名：巫逸哲

班級：資訊三乙

學號：D0641604

目錄

[壹、 規則建立說明 1](#_Toc38358008)

[貳、 程式碼 1](#_Toc38358009)

[參、 執行結果 12](#_Toc38358010)

[肆、 討論 13](#_Toc38358011)

[伍、 心得 13](#_Toc38358012)

**計算機演算法 作業1 分析Sorting時間**

# 規則建立說明

分別建立五種排序法的程式，執行時會計算執行時間，並分別輸出執行結果，並建立一個批次檔輪流執行五種排序法。

# 程式碼

## 氣泡排序 ( Bubble sort )

# include <stdio.h>

# include <stdlib.h>

# include <unistd.h>

# include <windows.h>

# include <time.h>

# define TIMES 25

# define MAX 50000

int \*generate\_number (int \*data) {

    for (int i = 0; i < MAX; i++) {

        data[i] = rand() % 1000000 + 1;

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

    return (int \*) data;

}

void check\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

}

double bubble\_sort (int \*data) {

    srand(time(NULL));

    clock\_t t\_start = clock();

    for (int i = 0; i < MAX; i++) {

        for (int j = 0; j < MAX-i-1; j++) {

            if (data[j] > data[j+1]) {

                int temp = data[j];

                data[j] = data[j+1];

                data[j+1] = temp;

            }

        }

    }

    clock\_t t\_end = clock();

    return ((double)t\_end - t\_start) / CLOCKS\_PER\_SEC;

}

int main () {

    int data[MAX] = {0};

    double times[TIMES] = {0};

    FILE \*output = fopen("bubble\_sort\_output.txt", "w");

    for (int i = 0; i < TIMES; i++) {

        generate\_number(data);

        printf("\n\t%2d times Sorting Start...\n\n\n", i + 1);

        times[i] = bubble\_sort(data);

        sleep(1);

        printf("\n\t%2d times Sorting End...\n\n\n", i + 1);

        check\_number(data);

        printf("\n\t%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        sleep(2);

    }

    system("cls");

    for (int i = 0; i < TIMES; i++) {

        printf("%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        fprintf(output, "%2d times -> Spend : %lf sec\n", i + 1, times[i]);

    }

    fclose(output);

    return 0;

}

## 選擇排序 ( Selection sort )

# include <stdio.h>

# include <stdlib.h>

# include <unistd.h>

# include <windows.h>

# include <time.h>

# define TIMES 25

# define MAX 50000

int \*generate\_number (int \*data) {

    for (int i = 0; i < MAX; i++) {

        data[i] = rand() % 1000000 + 1;

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

    return (int \*) data;

}

void check\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

}

double selection\_sort (int \*data) {

    srand(time(NULL));

    clock\_t t\_start = clock();

    for (int i = 0; i < MAX; i++) {

        for (int j = i + 1; j < MAX; j++) {

            if (data[i] > data[j]) {

                int temp = data[i];

                data[i] = data[j];

                data[j] = temp;

            }

        }

    }

    clock\_t t\_end = clock();

    return ((double)t\_end - t\_start) / CLOCKS\_PER\_SEC;

}

int main () {

    int data[MAX] = {0};

    double times[TIMES] = {0};

    FILE \*output = fopen("selection\_sort\_output.txt", "w");

    for (int i = 0; i < TIMES; i++) {

        generate\_number(data);

        printf("\n\t%2d times Sorting Start...\n\n\n", i + 1);

        times[i] = selection\_sort(data);

        sleep(1);

        printf("\n\t%2d times Sorting End...\n\n\n", i + 1);

        check\_number(data);

        printf("\n\t%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        sleep(2);

    }

    system("cls");

    for (int i = 0; i < TIMES; i++) {

        printf("%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        fprintf(output, "%2d times -> Spend : %lf sec\n", i + 1, times[i]);

    }

    fclose(output);

    return 0;

}

## 插入排序 ( Insertion sort )

# include <stdio.h>

# include <stdlib.h>

# include <unistd.h>

# include <windows.h>

# include <time.h>

# define TIMES 25

# define MAX 50000

int \*generate\_number (int \*data) {

    for (int i = 0; i < MAX; i++) {

        data[i] = rand() % 1000000 + 1;

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

    return (int \*) data;

}

void check\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

}

double insertion\_sort (int \*data) {

    srand(time(NULL));

    clock\_t t\_start = clock();

    for (int i = 1; i < MAX; i++) {

        int insertion\_index = i;

        while (insertion\_index > 0 && (data[insertion\_index-1] > data[insertion\_index])) {

            int temp = data[insertion\_index];

            data[insertion\_index] = data[insertion\_index-1];

            data[insertion\_index-1] = temp;

            insertion\_index -= 1;

        }

    }

    clock\_t t\_end = clock();

    return ((double)t\_end - t\_start) / CLOCKS\_PER\_SEC;

}

int main () {

    int data[MAX] = {0};

    double times[TIMES] = {0};

    FILE \*output = fopen("insertion\_sort\_output.txt", "w");

    for (int i = 0; i < TIMES; i++) {

        generate\_number(data);

        printf("\n\t%2d times Sorting Start...\n\n\n", i + 1);

        times[i] = insertion\_sort(data);

        sleep(1);

        printf("\n\t%2d times Sorting End...\n\n\n", i + 1);

        check\_number(data);

        printf("\n\t%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        sleep(2);

    }

    system("cls");

    for (int i = 0; i < TIMES; i++) {

        printf("%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        fprintf(output, "%2d times -> Spend : %lf sec\n", i + 1, times[i]);

    }

        fclose(output);

    return 0;

}

## 快速排序 ( Quick sort )

# include <stdio.h>

# include <stdlib.h>

# include <unistd.h>

# include <windows.h>

# include <time.h>

# define TIMES 25

# define MAX 50000

int \*generate\_number (int \*data) {

    for (int i = 0; i < MAX; i++) {

        data[i] = rand() % 1000000 + 1;

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

    return (int \*) data;

}

void check\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

}

int Partition(int \*data, int front, int end){

    int i = front - 1, temp = 0;

    for (int j = front; j < end; j++) {

        if (data[j] < data[end]) {

            i++;

            temp = data[i];

            data[i] = data[j];

            data[j] = temp;

        }

    }

    i++;

    temp = data[i];

    data[i] = data[end];

    data[end] = temp;

    return i;

}

double quick\_sort (int \*data, int front, int end) {

    srand(time(NULL));

    clock\_t t\_start = clock();

    if (front < end) {

        int seed = Partition(data, front, end);

        quick\_sort(data, front, seed - 1);

        quick\_sort(data, seed + 1, end);

    }

    clock\_t t\_end = clock();

    return ((double)t\_end - t\_start) / CLOCKS\_PER\_SEC;

}

int main () {

    int data[MAX] = {0};

    double times[TIMES] = {0};

    FILE \*output = fopen("quick\_sort\_output.txt", "w");

    for (int i = 0; i < TIMES; i++) {

        generate\_number(data);

        printf("\n\t%2d times Sorting Start...\n\n\n", i + 1);

        times[i] = quick\_sort(data, 0, MAX-1);

        sleep(1);

        printf("\n\t%2d times Sorting End...\n\n\n", i + 1);

        check\_number(data);

        printf("\n\t%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        sleep(2);

    }

    system("cls");

    for (int i = 0; i < TIMES; i++) {

        printf("%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        fprintf(output, "%2d times -> Spend : %lf sec\n", i + 1, times[i]);

    }

    fclose(output);

    return 0;

}

## 堆積排序 ( Heap sort )

# include <stdio.h>

# include <stdlib.h>

# include <unistd.h>

# include <windows.h>

# include <time.h>

# define TIMES 25

# define MAX 50000

int \*generate\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        data[i] = rand() % 1000000 + 1;

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

    return (int \*) data;

}

void check\_number (int \*data) {

    for (int i = 1; i < MAX; i++) {

        ((i+1)%10 == 0) ? printf("%10d\n", data[i]) : printf("%10d ", data[i]);

    }

}

void MaxHeapify(int \*data, int root, int length) {

    int left\_child = 2 \* root;

    int right\_child = 2 \* root + 1;

    int largest = 0;

    if (left\_child <= length && data[left\_child] > data[root]) {

        largest = left\_child;

    } else {

        largest = root;

    }

    if (right\_child <= length && data[right\_child] > data[largest]) {

        largest = right\_child;

    }

    if (largest != root) {

        int temp = data[largest];

        data[largest] = data[root];

        data[root] = temp;

        MaxHeapify(data, largest, length);

    }

}

void BuildMaxHeap (int \*data) {

    for (int i = MAX/2; i >= 1;i--) {

        MaxHeapify(data, i, MAX-1);

    }

}

double heap\_sort (int \*data) {

    srand(time(NULL));

    clock\_t t\_start = clock();

    BuildMaxHeap(data);

    int size = MAX-1;

    for (int i = size; i>=2 ; i--) {

        int temp = data[1];

        data[1] = data[i];

        data[i] = temp;

        size--;

        MaxHeapify(data, 1, size);

    }

    clock\_t t\_end = clock();

    return ((double)t\_end - t\_start) / CLOCKS\_PER\_SEC;

}

int main () {

    int data[MAX] = {0};

    double times[TIMES] = {0};

    FILE \*output = fopen("heap\_sort\_output.txt", "w");

    for (int i = 0; i < TIMES; i++) {

        generate\_number(data);

        printf("\n\t%2d times Sorting Start...\n\n\n", i + 1);

        times[i] = heap\_sort(data);

        sleep(1);

        printf("\n\t%2d times Sorting End...\n\n\n", i + 1);

        check\_number(data);

        printf("\n\t%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        sleep(2);

    }

    system("cls");

    for (int i = 0; i < TIMES; i++) {

        printf("%2d times -> Spend : %lf sec\n", i + 1, times[i]);

        fprintf(output, "%2d times -> Spend : %lf sec\n", i + 1, times[i]);

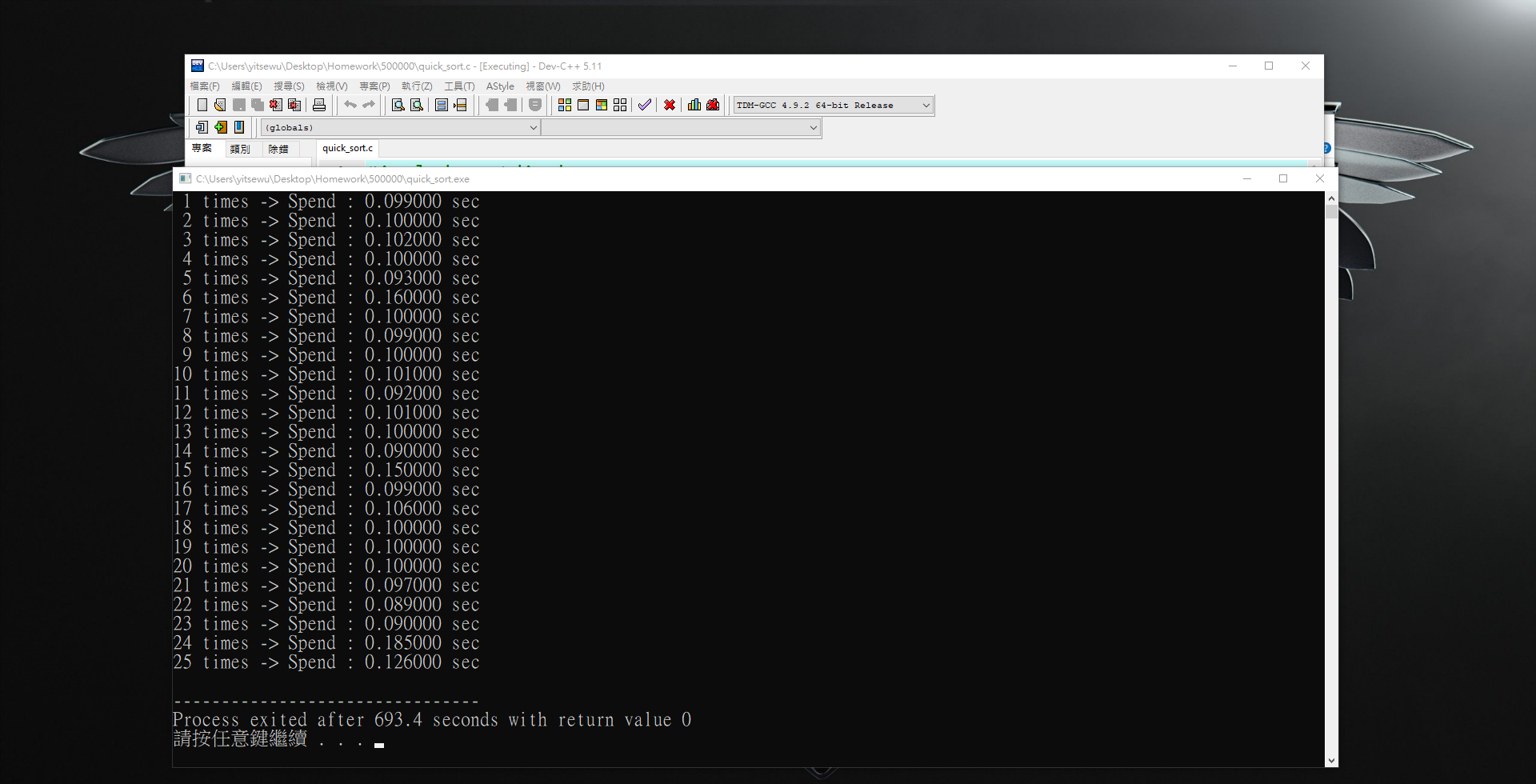
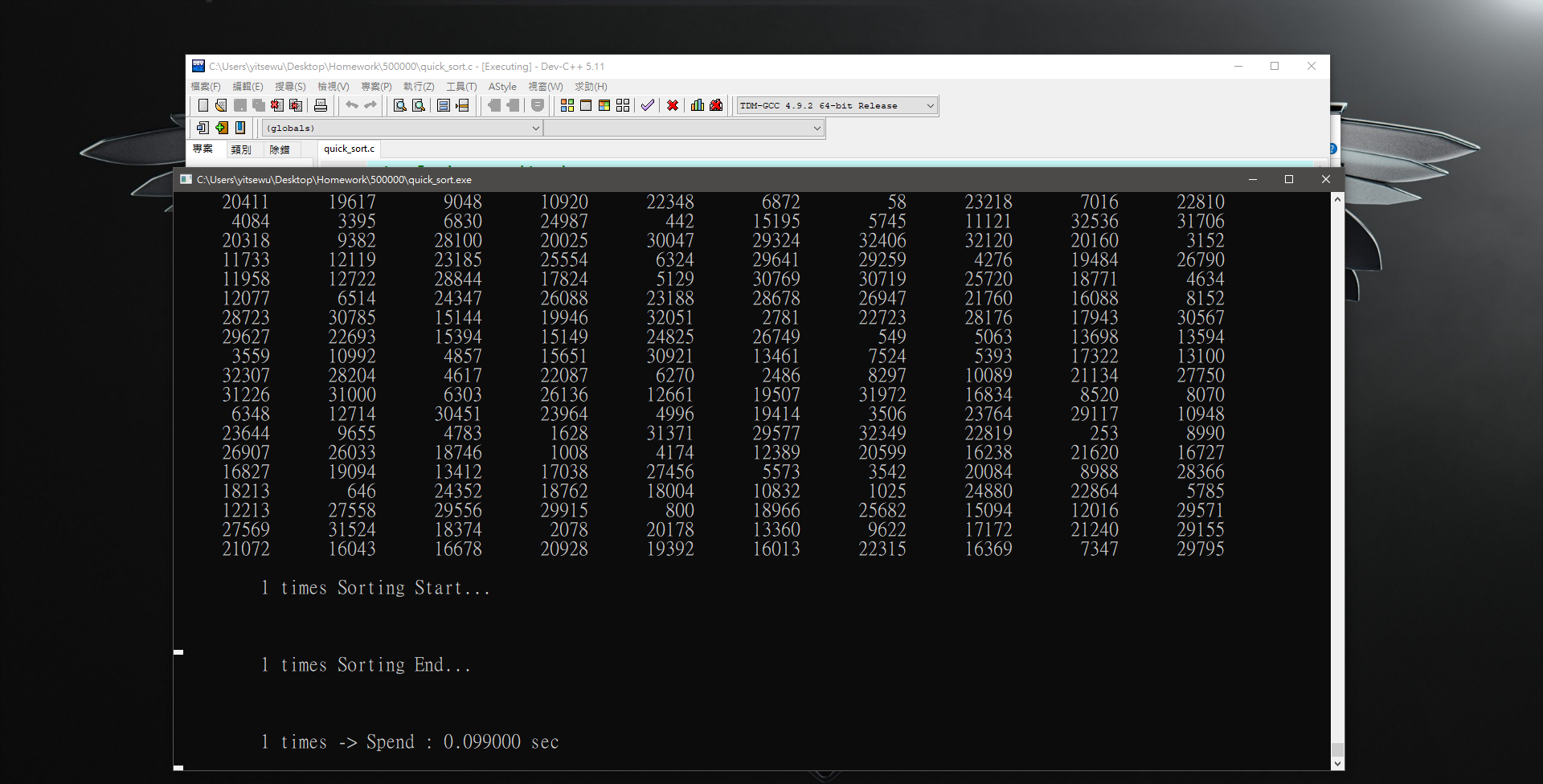
    }

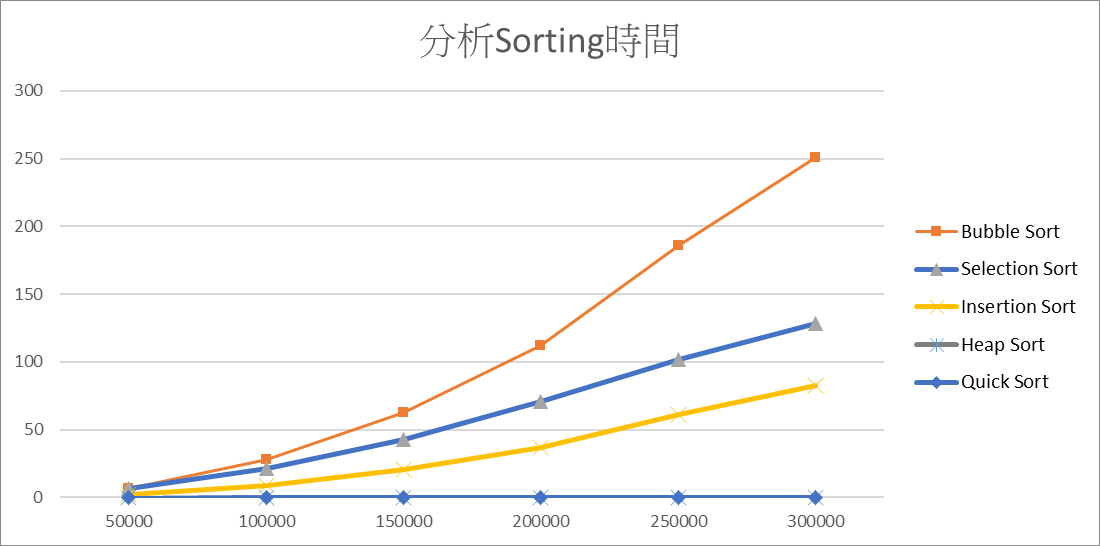
    fclose(output);

    return 0;

}

# 執行結果





# 討論

時間上來說，Bubble sort真的最花時間，然後Quick sort跟Heap sort議整個快到不行，在測資500000的時候最有感覺。以下全部花費時間Bubble Sort > Selection Sort > Insertion Sort > Quick Sort ~= Heap Sort。

# 心得

我寫好以後發現他真的要跑好久，然後一開始我沒有想好，沒有寫任何可以寫檔記錄的功能，然後分開寫的五隻程式，每隻跑六次，我花很多時間在等待要執行下一次，然後還會跑完沒記錄到，然後後來搞了寫檔，又寫了一隻批次檔，可以幫我跑全部，結果我寫檔的部分因為都是複製貼上寫好的同個模板，居然把Selection Sort 跟Bubble Sort的輸出寫到同檔案，後來要寫報告的時候才發現大事不妙，只好再跑一次，然後再重新等那500000的跑完，整個氣到不行。