

## Homework 1. Quadratic Sorts

106061151 劉安得

### 1. Introduction

This homework is to implement four kind of sorting algorithms, selection sort, insertion sort, bubble sort, and odd-even sort. Then record the CPU time for every input file and compare their performance.

### 2. Approach

#### A. Selection Sort

##### Algorithm 1.1.3. Selection Sort.

```
// Sort the array  $A[1 : n]$  into nondecreasing order.
// Input:  $A[1 : n]$ , int  $n$ 
// Output:  $A$ ,  $A[i] \leq A[j]$  if  $i < j$ .
1 Algorithm SelectionSort( $A, n$ )
2 {
3     for  $i := 1$  to  $n$  do { // for every  $A[i]$ 
4          $j := i$ ; // Initialize  $j$  to  $i$ 
5         for  $k := i + 1$  to  $n$  do // Search for the smallest in  $A[i + 1 : n]$ .
6             if ( $A[k] < A[j]$ ) then  $j := k$ ; // Found, remember it in  $j$ .
7          $t := A[i]$ ;  $A[i] := A[j]$ ;  $A[j] := t$ ; // Swap  $A[i]$  and  $A[j]$ .
8     }
9 }
```

For every index  $i$ , search for the smallest element in between  $i+1$  and  $n$ . And then swap it with index  $i$ .

Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$


#### B. Insertion sort

---

```

// Sort  $A[1 : n]$  into nondecreasing order.
// Input: array  $A$ ,  $\text{int } n$ 
// Output: array  $A$  sorted.
1 Algorithm InsertionSort( $A, n$ )
2 {
3   for  $j := 2$  to  $n$  do { // Assume  $A[1 : j - 1]$  already sorted.
4      $\text{item} := A[j]$ ; // Move  $A[j]$  to its proper place.
5      $i := j - 1$ ; // Init  $i$  to be  $j - 1$ .
6     while  $i \geq 1$  and  $\text{item} < A[i]$  do { // Find  $i$  such that  $A[i] \leq A[j]$ .
7        $A[i + 1] := A[i]$ ; // Move  $A[i]$  up by one position.
8        $i := i - 1$ ;
9     }
10     $A[i + 1] = \text{item}$ ; // Move  $A[j]$  to  $A[i + 1]$ .
11  }
12 }

```



For every index  $i$ , assume  $A[0 : i-1]$  is already sorted. And find the proper place to insert  $A[i]$  between  $A[0 : i-1]$ .

Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$

### C. Bubble Sort

---

```

// Sort  $A[1 : n]$  into nondecreasing order.
// Input: array  $A$ ,  $\text{int } n$ 
// Output: array  $A$  sorted.
1 Algorithm BubbleSort( $A, n$ )
2 {
3   for  $i := 1$  to  $n - 1$  do { // Find the smallest item for  $A[i]$ .
4     for  $j := n$  to  $i + 1$  step  $-1$  do {
5       if  $A[j - 1] > A[j]$  then { // Swap  $A[j]$  and  $A[j - 1]$ .
6          $t = A[j]$ ;
7          $A[j] = A[j - 1]$ ;
8          $A[j - 1] = t$ ;
9       }
10    }
11  }
12 }

```

---

For every index  $i$ , find the smallest element in  $A[i : n]$ , and insert it at  $A[i]$ .

Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$

D. Odd-Even Sort

---

```
// Sort A[1 : n] into nondecreasing order.
// Input: array A, int n
// Output: array A sorted.
1 Algorithm OddEvenSort(A, n)
2 {
3     done := false ;
4     while not done do {
5         done := true ;
6         for i := 1 to n - 1 step 2 do {
7             if A[i] > A[i + 1] then {
8                 done := false ;
9                 t = A[i]; A[i] = A[i + 1]; A[i + 1] = t;
10            }
11        }
12        for i := 0 to n - 1 step 2 do {
13            if A[i] > A[i + 1] then {
14                done := false ;
15                t = A[i]; A[i] = A[i + 1]; A[i + 1] = t;
16            }
17        }
18    }
19 }
```

---



Compare the adjacent index, swap according to nondecreasing order. And keep checking whether the order is correct.

Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$

3. Result & Observations

A. Table of results obtained

(Units: s)	selection	insertion	bubble	odd-even
------------	-----------	-----------	--------	----------

s1.dat	0.000002	0.000002	0.000002	0.000002
s2.dat	0.000005	0.000004	0.000006	0.000006
s3.dat	0.000013	0.000011	0.000021	0.000020
s4.dat	0.000042	0.000029	0.000078	0.000073
s5.dat	0.000145	0.000092	0.000305	0.000282
s6.dat	0.000512	0.000338	0.001177	0.001073
s7.dat	0.001943	0.001256	0.023431	0.020098
s8.dat	0.026003	0.025066	0.095048	0.089975
s9.dat	0.119151	0.078157	0.332129	0.334710

#### B. Observations

According to the result, Insertion Sort's performance is the best; the 2<sup>nd</sup> place is Selection Sort; And Bubble Sort and Odd-Even is the worst.

#### C. Conclusions

The time complexities of these four algorithms are all  $O(n^2)$ , I think this is the reason why the title of this homework is called "Quadratic Sorts".