

Insertion Sort

20170411

used some pictures and codes from
<http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf>
Data Structures and Algorithm Analysis
by Clifford A. Schaffer

Copyright (c) 2015 - 2017 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the
GNU Free Documentation License, Version 1.2 or any later version published by the Free Software
Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of
the license is included in the section entitled "GNU Free Documentation License".

| | i=1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----|-----|----|----|----|----|----|----|
| 42 | 20 | 17 | 13 | 13 | 13 | 13 | 13 |
| 20 | 42 | 20 | 17 | 17 | 14 | 14 | 14 |
| 17 | 17 | 42 | 20 | 20 | 17 | 17 | 15 |
| 13 | 13 | 13 | 42 | 28 | 20 | 20 | 17 |
| 28 | 28 | 28 | 28 | 42 | 28 | 23 | 20 |
| 14 | 14 | 14 | 14 | 14 | 42 | 28 | 23 |
| 23 | 23 | 23 | 23 | 23 | 23 | 42 | 28 |
| 15 | 15 | 15 | 15 | 15 | 15 | 15 | 42 |

42

20

17

13

28

14

23

15

① - A

| | $i=1$ | $i=2$ | $i=3$ |
|---|----------|----------|----------|
| 0 | 42 20 | 20 42 | 17 20 |
| 1 | 20 42 | 42 17 | 17 20 |
| 2 | 17 17 | 17 42 | 42 13 |
| 3 | 13 13 | 13 13 | 13 42 |
| 4 | 28 28 | 28 28 | 28 28 |
| 5 | 14 14 | 14 14 | 14 14 |
| 6 | 23 23 | 23 23 | 23 23 |
| 7 | 15 15 | 15 15 | 15 15 |

| | $i=4$ | $i=5$ |
|---|----------|----------------------------|
| 0 | 13 13 | 13 13 13 13 13 |
| 1 | 17 17 | 17 17 17 17 14 |
| 2 | 20 20 | 20 20 20 14 17 |
| 3 | 42 28 | 28 28 14 20 20 |
| 4 | 28 42 | 42 14 28 28 28 |
| 5 | 14 14 | 14 42 42 42 42 |
| 6 | 23 23 | 23 23 23 23 23 |
| 7 | 15 15 | 15 15 15 15 15 |

2

- A

 $i=6$

| | | | |
|---|----|----|----|
| 0 | 13 | 13 | 13 |
| 1 | 14 | 14 | 14 |
| 2 | 17 | 17 | 17 |
| 3 | 20 | 20 | 20 |
| 4 | 28 | 28 | 23 |
| 5 | 42 | 23 | 28 |
| 6 | 23 | 42 | 42 |
| 7 | 15 | 15 | 15 |

 $i=7$

| | | | | | |
|----|----|----|----|----|----|
| 13 | 13 | 13 | 13 | 13 | 13 |
| 14 | 14 | 14 | 14 | 14 | 14 |
| 17 | 17 | 17 | 17 | 17 | 17 |
| 20 | 20 | 20 | 20 | 15 | 17 |
| 23 | 23 | 23 | 15 | 20 | 20 |
| 28 | 28 | 15 | 23 | 23 | 23 |
| 42 | 42 | 15 | 28 | 28 | 28 |
| 15 | 42 | 42 | 42 | 42 | 42 |

① - B

| | $i=1$ | $i=2$ | $i=3$ | |
|---|-------|-------|-------|----|
| 0 | 42 | 20 | 17 | 13 |
| 1 | 20 | 42 | 20 | 17 |
| 2 | 17 | 17 | 42 | 20 |
| 3 | 13 | 13 | 13 | 42 |
| 4 | 28 | 28 | 28 | 28 |
| 5 | 14 | 14 | 14 | 14 |
| 6 | 23 | 23 | 23 | 23 |
| 7 | 15 | 15 | 15 | 15 |

| | $i=4$ | $i=5$ | |
|---|-------|-------|----|
| 0 | 13 | 13 | 13 |
| 1 | 17 | 17 | 17 |
| 2 | 20 | 20 | 20 |
| 3 | 42 | 28 | 14 |
| 4 | 28 | 42 | 28 |
| 5 | 14 | 14 | 42 |
| 6 | 23 | 23 | 23 |
| 7 | 15 | 15 | 15 |

2 - B

| | $i=6$ | | | $i=7$ | | | | | | | |
|---|-------|----|----|-------|----|----|----|----|----|----|----|
| 0 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| 1 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| 2 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 15 |
| 3 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 15 | 17 |
| 4 | 28 | 28 | 23 | 23 | 23 | 23 | 23 | 23 | 15 | 20 | 20 |
| 5 | 42 | 23 | 28 | 28 | 28 | 15 | 23 | 23 | 23 | 23 | 23 |
| 6 | 23 | 42 | 42 | 42 | 15 | 28 | 28 | 28 | 28 | 28 | 42 |
| 7 | 15 | 15 | 15 | 15 | 42 | 42 | 42 | 42 | 42 | 42 | 42 |

C++ template

```
#include <stdio.h>
template <class T>
T square(T x) {
    return (x*x);
}
```

int
float
double

```
int main(void) {
    int i2, i=2;
    float f2, f=3.0;
    double d2, d=4.0;

    i2 = square<int>(i);
    f2 = square<float>(f);    <float>
    d2 = square<double>(d);   <double>

    printf("i= %d i2= %d \n", i, i2);
    printf("f= %f f2= %f \n", f, f2);
    printf("d= %f d2= %f \n", d, d2);

    return 0;
}
```

```
template <class T>
T square(T x) {
    return (x*x);
}
```

```
template <typename T>
T square(T x) {
    return (x*x);
}
```

square<float>(f);

```
int square(float x) {
    return (x*x);
}
```

square<int>(i);

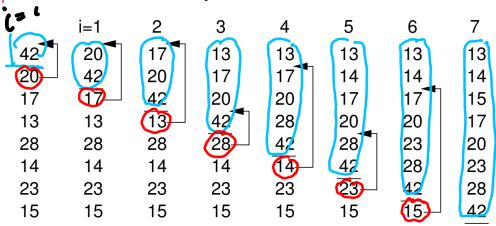
```
int square(int x) {
    return (x*x);
}
```

square<double>(d);

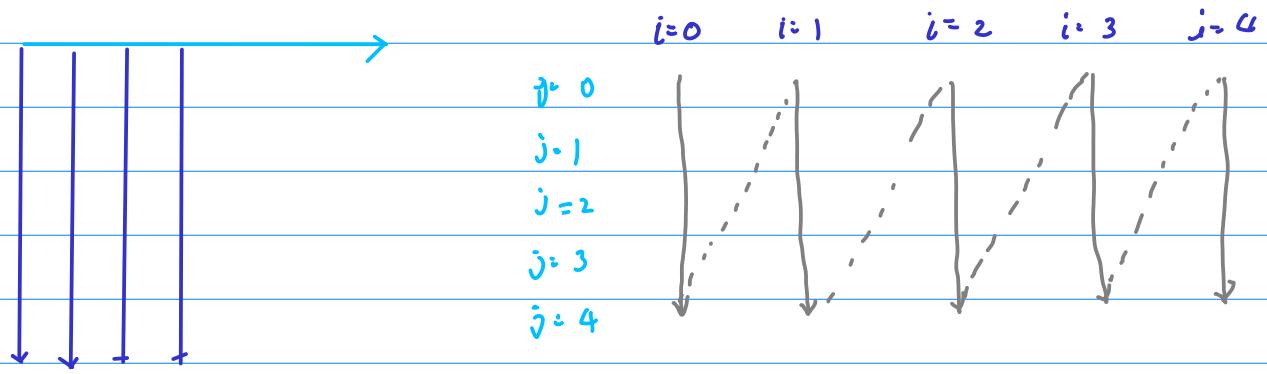
```
double square(double x) {
    return (x*x);
}
```

Swap

```
// Swap two elements in a generic array
template<typename E>
inline void swap(E A[], int i, int j) {
    E temp = A[i];
    A[i] = A[j];
    A[j] = temp;
}
// Random number generator functions
```

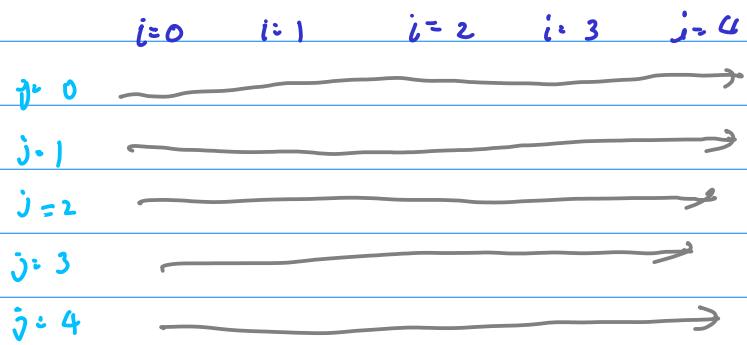


```
template <typename E, typename Comp>
void inssort(E A[], int n) { // Insertion Sort
    for (int i=1; i<n; i++) // Insert i'th record
        for (int j=i; (j>0) && (Comp::prior(A[j] < A[j-1])); j--)
            swap(A, j, j-1);
}
```



`for (i=0; i<5; ++i)`

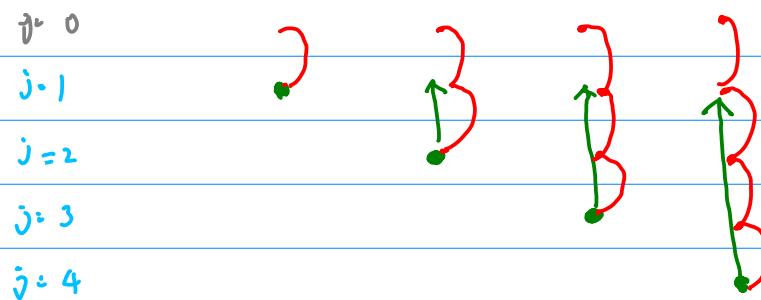
`for (j=0; j<5; --j)`



`for (j=0; j<5; --j)`

`for (i=0; i<5; ++i)`

$i=0$ $i=1$ $i=2$ $i=3$ $i=4$



`for($i=1$; $i<5$; $++i$)`

`for($j=i$; $j>0$; $-j$)`

$A[j] < A[j-1]$

i=1

| | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| 42 | 20 | 17 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 7 |
| 20 | 42 | 20 | 42 | 20 | 42 | 20 | 42 | 20 | 42 | 20 | 42 | 20 | 42 | |
| 17 | 17 | 17 | 17 | 28 | 14 | 14 | 23 | 23 | 23 | 28 | 28 | 23 | 23 | |
| 13 | 13 | 13 | 13 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 17 | 17 | |
| 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 23 | 23 | |
| 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 20 | 20 | |
| 23 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 28 | 28 | |
| 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 42 | 42 | |

i=1 j=1

i=2 j=2

i=2 j=1

i=3 j=3

i=3 j=2

i=3 j=1.

i=4 j=4.

i=5 j=5

i=5 j=4

i=5 j=3

i=5 j=2

i=6 j=6

i=6 j=5

i=7 j=7

i=7 j=6

i=7 j=5

i=7 j=4

i=7 j=3

for $i = 1$ to N

for $j = 2$ to $(N-1)$

① 2

do {

Read $A[i, j] :$

Write $A[i, j] :$

}

$(N-1) \times 2 + 1$

$(N-2) \times 2$

↑

Read
Write

실행되는 명령의 횟수

Read = A

write A =

Comp. $A < B >$

$$N \times (N-2) \times 2 = 2N^2 - 4N = \mathcal{O}(N^2)$$

Quadratic time Alg.

#define MAX 9000000

for $i = \underline{1000}$ to $\underline{N-20000}$
for $j = 2$ to MAX
do {
 Read $A[i, j]$;
 Write $A[i, j]$;
}
}

$N - 20000 - 1000$
 $= N - 19000$
 $9000000 - 2 + 1$
8999999

$$(N - 19000) \times \underline{8999999} \times 2 = \underline{O(N)}$$

linear time alg.

References

- [1] <http://en.wikipedia.org/>
- [2] <http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf>