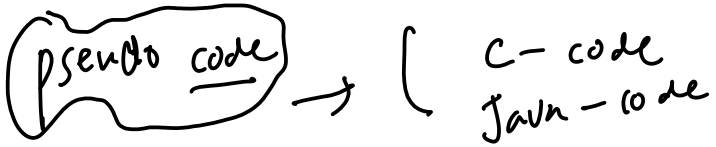
Algorithms – Bubble Sort (1B)

Young Won Lim
4/5/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Bubble Sort Algorithm



Rosen

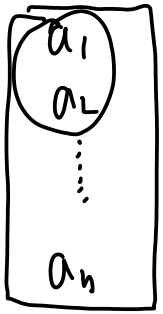
procedure bubblesort(a_1, … , a_n : real numbers with n ≥ 2)

for i := 1 **to** n-1

  for j := 1 **to** n – i

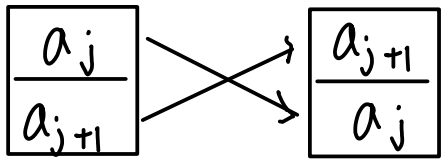
    if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}

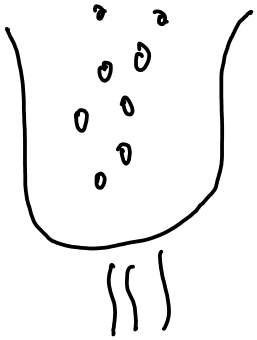
{a_1, …, a_n is in increasing order}



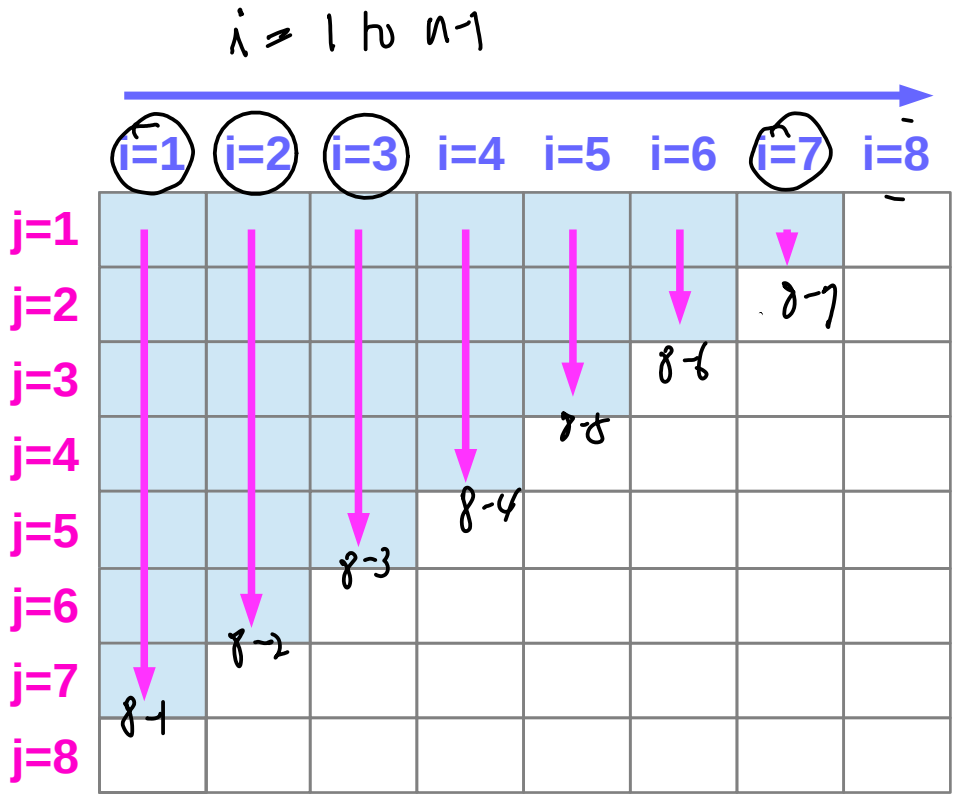
n = 8

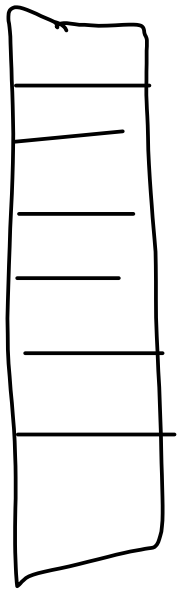
$a_j > a_{j+1}$





Nested loop iterations

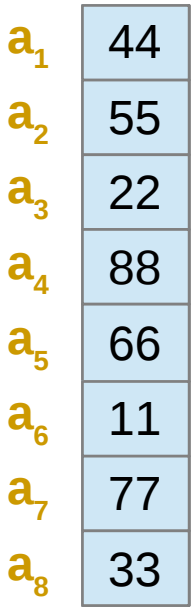


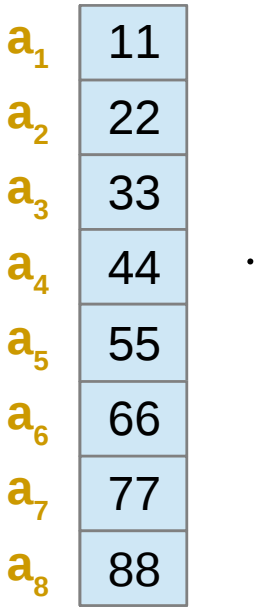


n = 8

```
for i := 1 to n-1
    for j := 1 to n – i
```

Input and Ouput





n=8

$a_1, \ldots, a_n$: real numbers

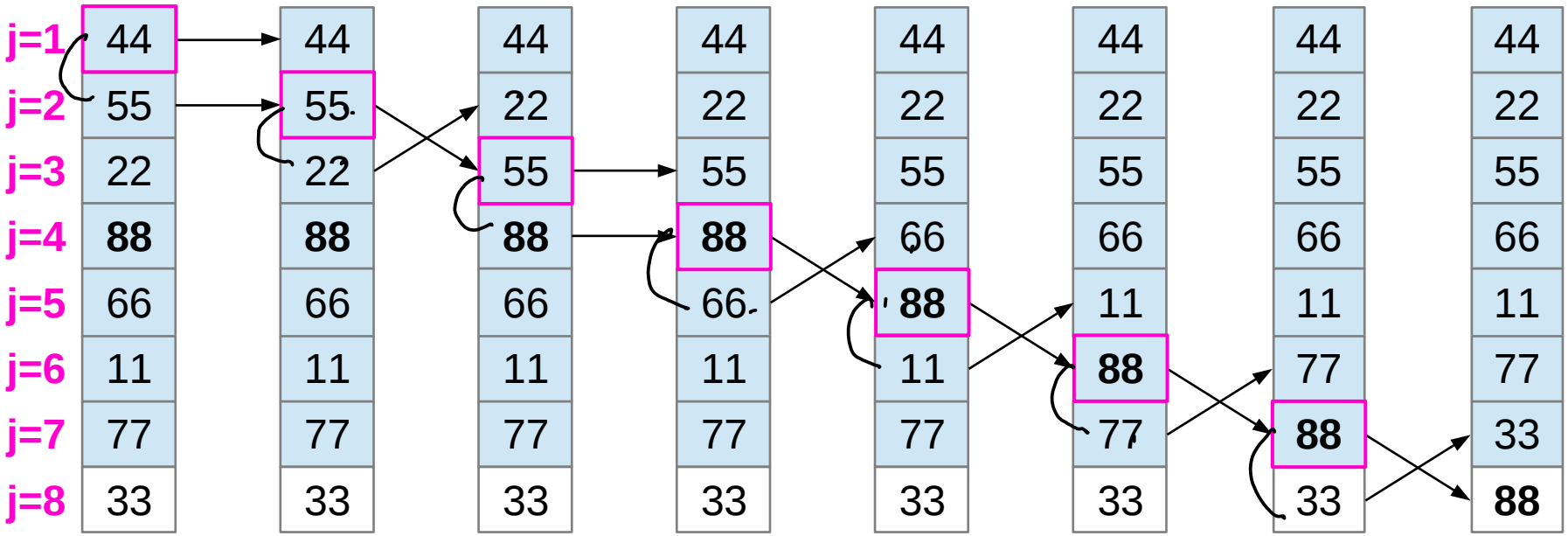
with $n \geq 2$

n=8

{$a_1, \ldots, a_n$ is in increasing order}

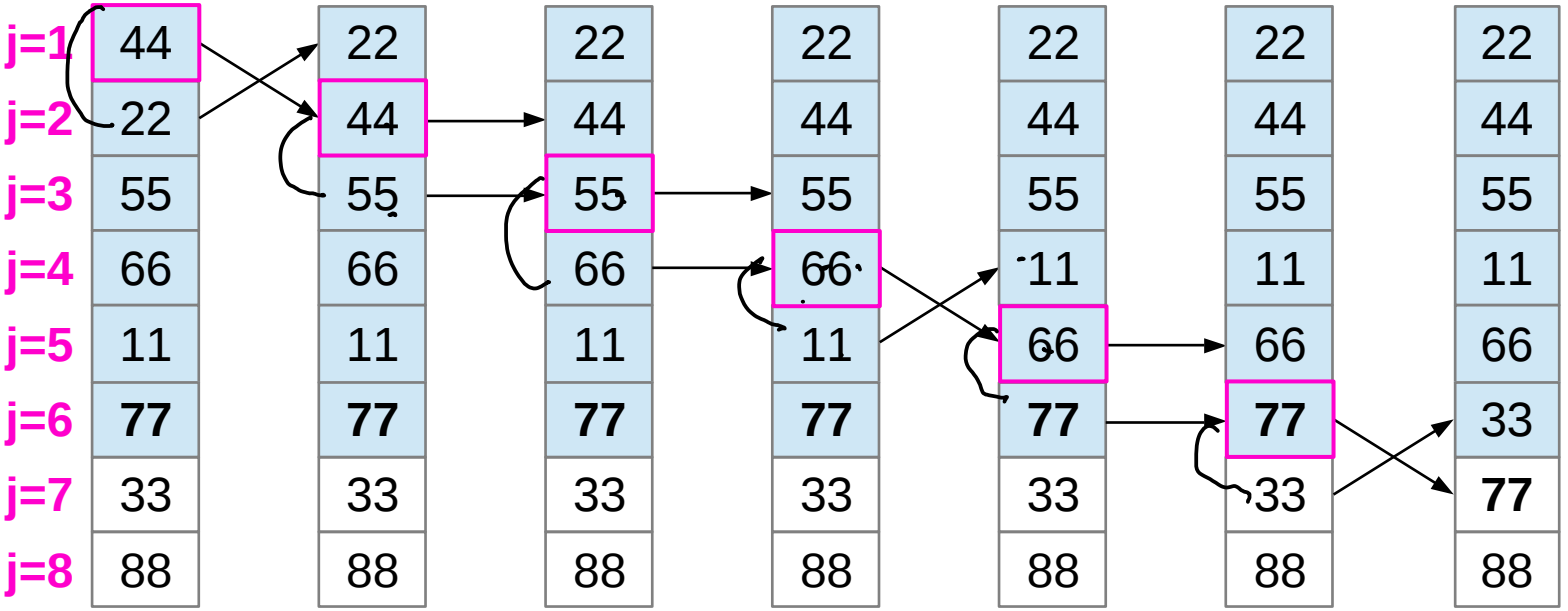
Step i=1

j = 1 ... 8-1




```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Step i=2

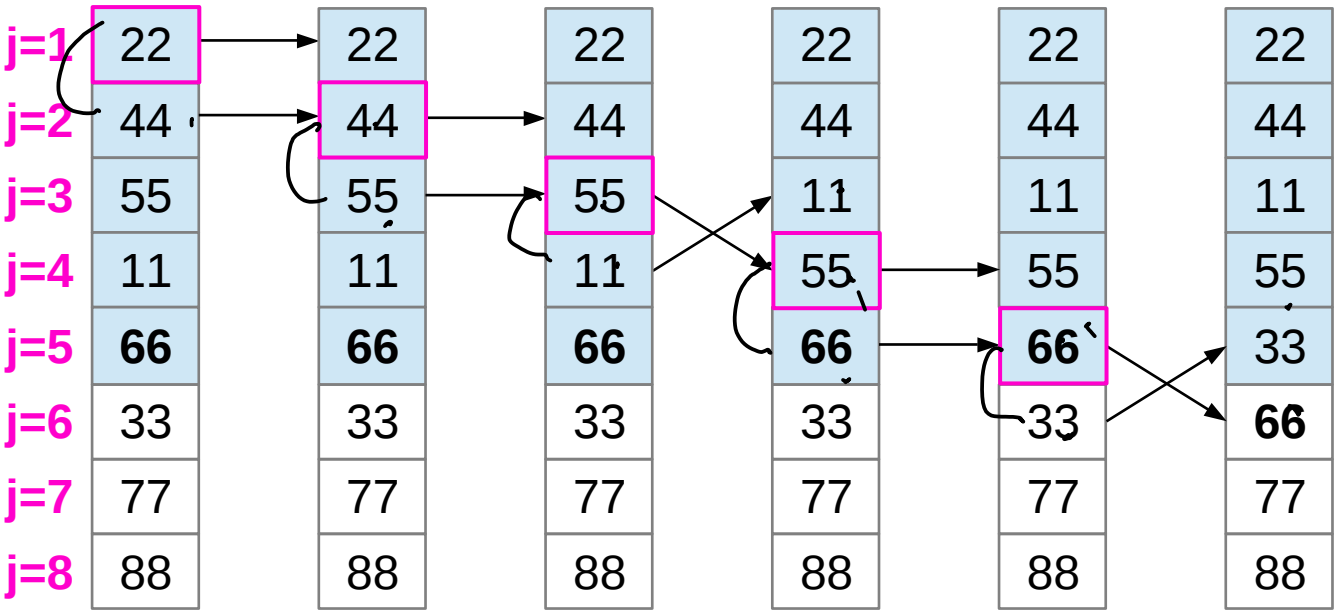


for i := 1 **to** n-1

for j := 1 **to** n – i

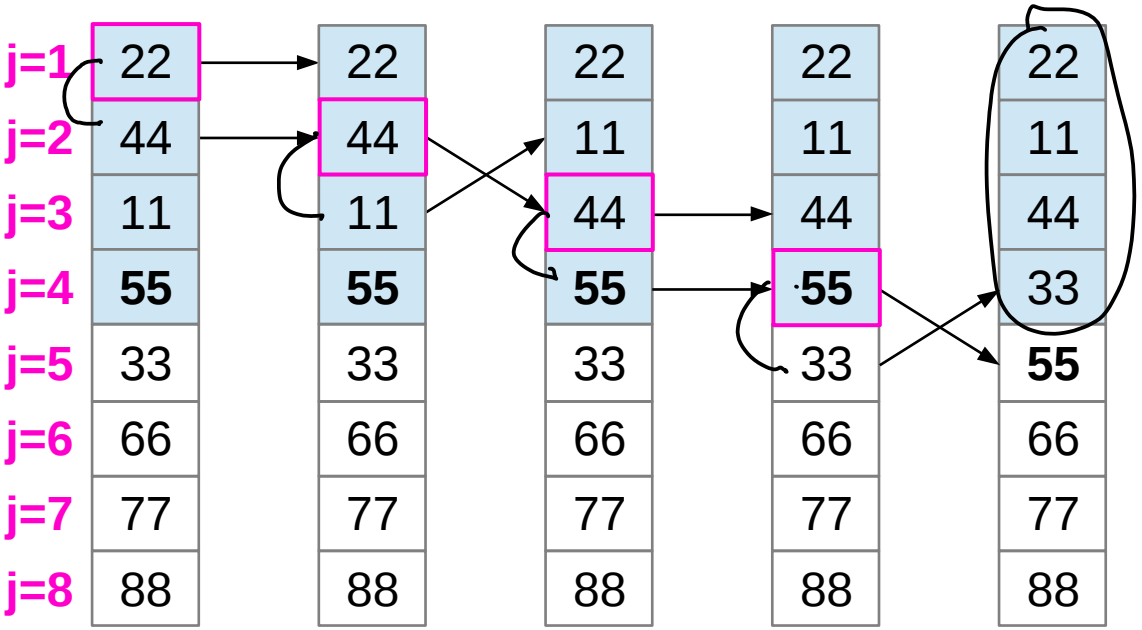
if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}

Step i=3



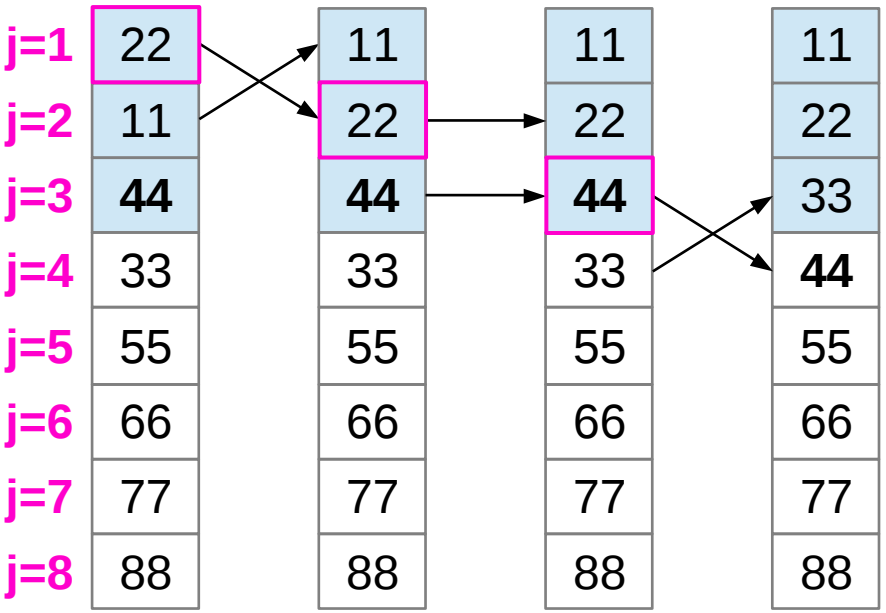

```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Step i=4



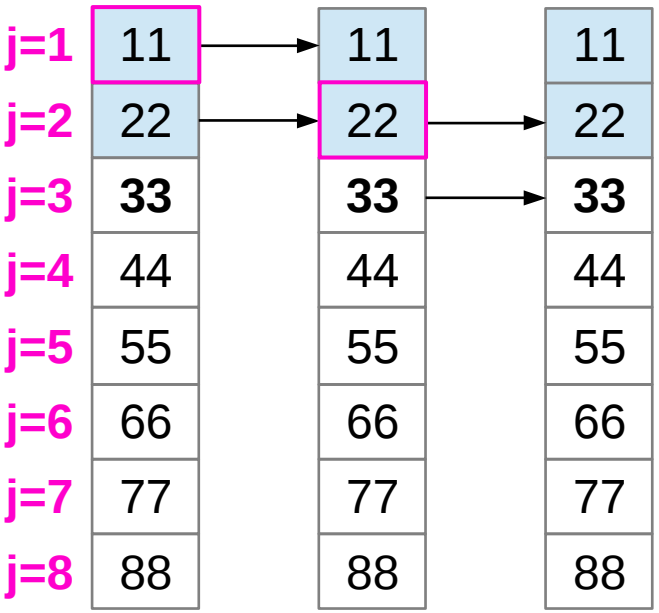

```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Step i=5



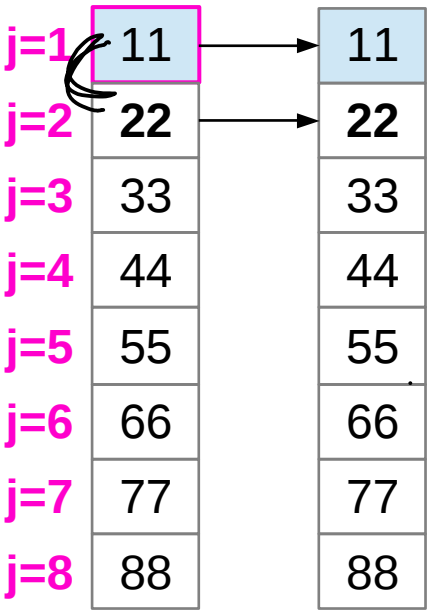

```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Step i=6



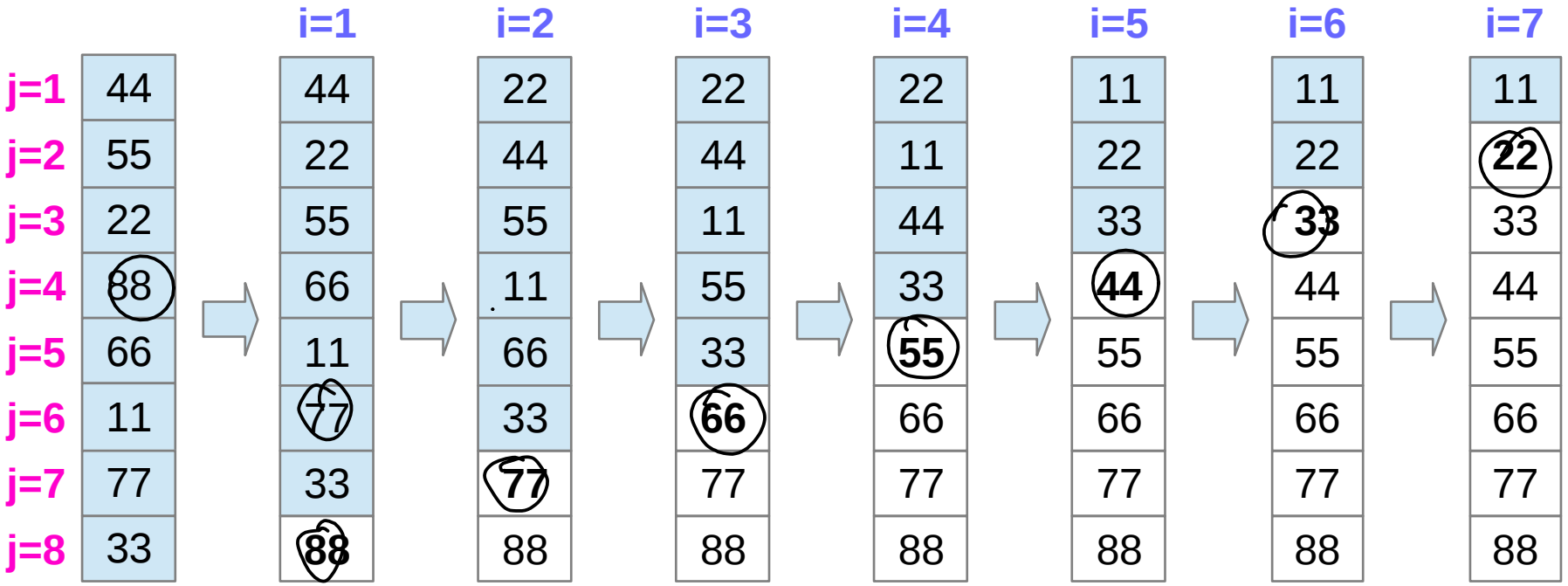

```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Step i=7




```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

Summary




```
for i := 1 to n-1
    for j := 1 to n – i
        if a_j > a_j+1 then interchange a_j and a_j+1
```

References

[1] http://en.wikipedia.org/
[2]

Algorithms – Insertion Sort (1C)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

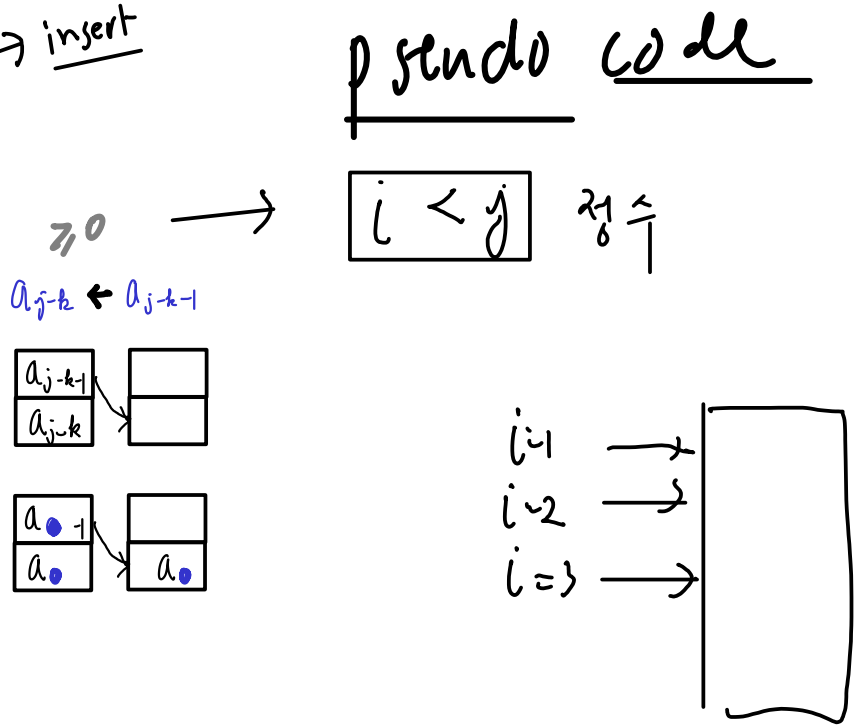
This document was produced by using LibreOffice and Octave.

Insertion Sort Algorithm

procedure insertion sort(a_1, … , a_n : real numbers with n ≥ 2)

```
for j := 2 to n
    i := 1
    while a_j > a_i
        i := i + 1
    m := a_j
    for k := 0 to j – i –1
        a_(j-k) = a_(j-k-1)
    a_i := m
```

{a_1, …, a_n is in increasing order}



$a_{\bigcirc} = a_{\bigcirc -1}$

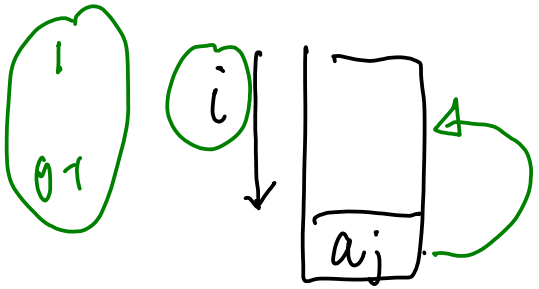
Nested loop k – constraints

0 ... 0
0 ... 1
0 ... 2

0 ... j-1-1 : (j-i)개

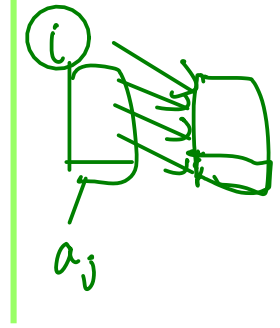
```
for k := 0 to j– i–1
    a_{j-k} = a_{j-k-1}
```

$j-i-1 \geq 0$ $\quad j \geq i+1$ $\quad i \leq j-1$ $\quad i < j$



$$a_{j-k} = a_{j-k-1}$$

$(k=0)$	$a_{j-0} = a_{j-0-1}$	$a_j = a_{j-1}$
$(k=1)$	$a_{j-1} = a_{j-1-1}$	$a_{j-1} = a_{j-2}$
$(k=2)$	$a_{j-2} = a_{j-2-1}$	$a_{j-2} = a_{j-3}$
	$\vdots = \vdots$	$\vdots = \vdots$
$(k=j-i-1)$	$a_{j-(j-i-1)} = a_{j-(j-i-1)-1}$	$a_{i+1} = a_i$



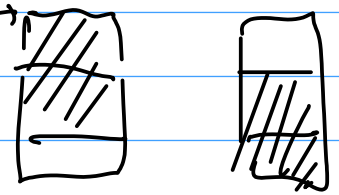
k = 0 to 1 2번

0 to 2 3번

0 to n n+1번

0 to j-i-1 (j-i)번

(3-1) = 2번



i=1

j=3

Nested loop k – rearranging

```
for k := 0 to j– i–1
    a_{j-k} = a_{j-k-1}
```

$j - i - 1 \geq 0 \qquad j \geq i + 1 \qquad i \leq j - 1 \qquad i < j$

$$a_{j-k} = a_{j-k-1}$$

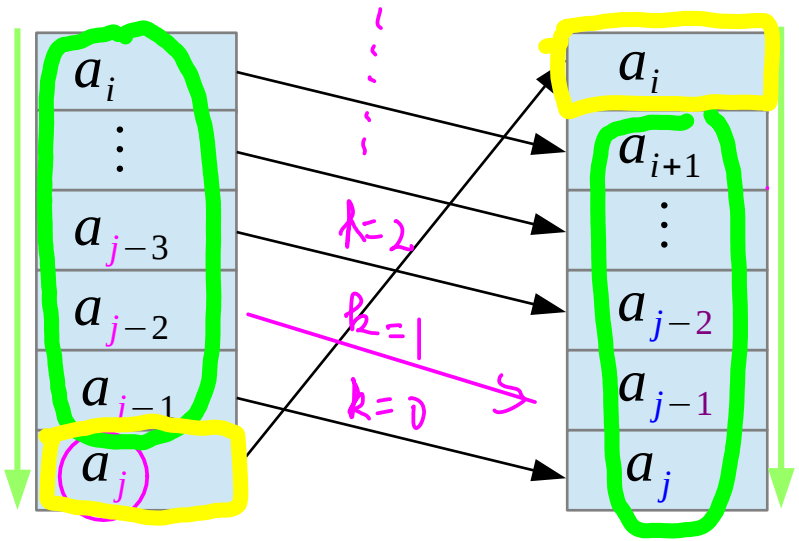
①	$a_j = a_{j-1}$
②	$a_{j-1} = a_{j-2}$
③	$a_{j-2} = a_{j-3}$
	$\vdots = \vdots$
i	$a_{i+1} = a_i$

	$a_{i+1} = a_i$
	$\vdots = \vdots$
③	$a_{j-2} = a_{j-3}$
②	$a_{j-1} = a_{j-2}$
①	$a_j = a_{j-1}$

Nested loop k – data movement

```
m := a_j
for k := 0 to j– i–1
        a_j-k = a_j-k-1
a_i := m
```

$i < j$



k=0
k=1
k=2

a_j	$= a_{j-1}$
a_{j-1}	$= a_{j-2}$
a_{j-2}	$= a_{j-3}$
$\vdots$	$= \vdots$
a_{i+1}	$= a_i$

a_{i+1}	$=$
$\vdots$	$=$
a_{j-2}	$=$
a_{j-1}	$=$
a_j	$=$

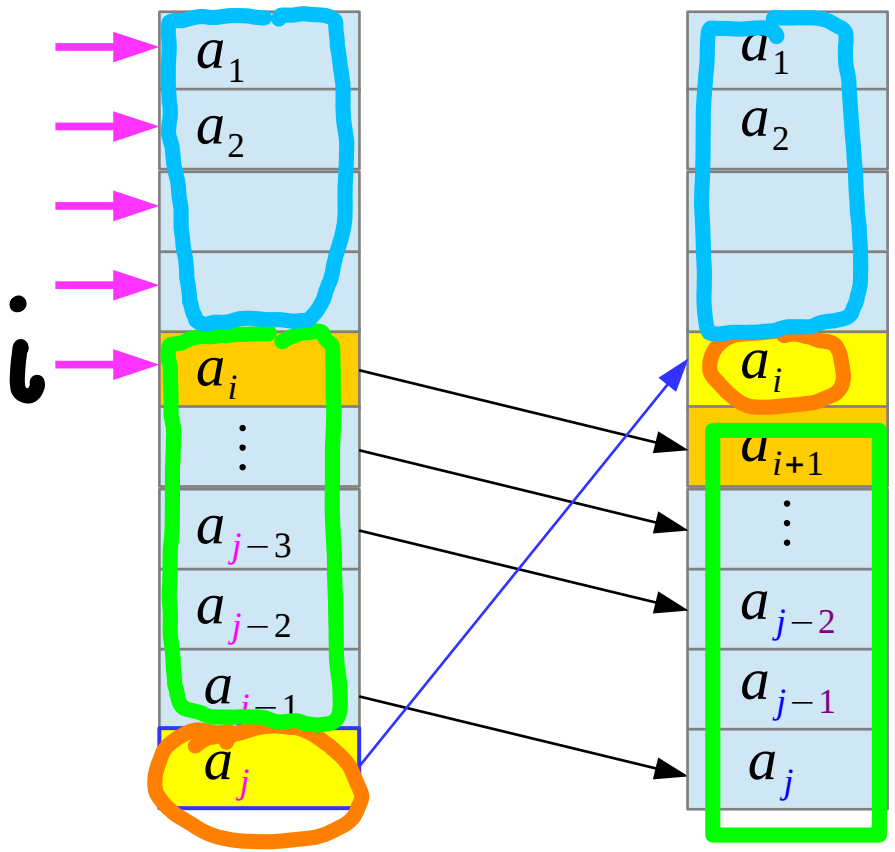
Nested loop i – finding out of order a_i

```
i := 1
while a_j > a_i
        i := i + 1
```

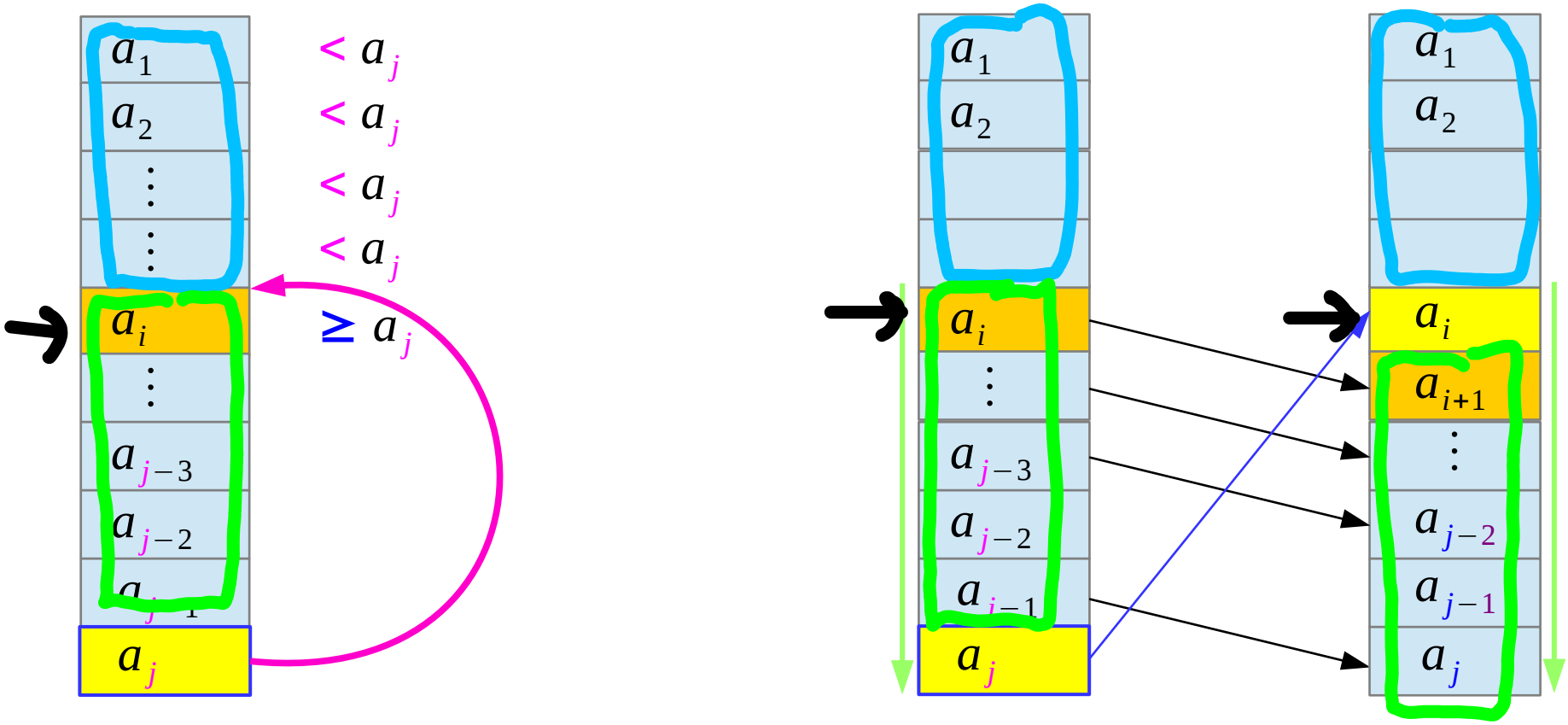
If a_i **<** a_j increment I

If a_i **>=** a_j break the loop

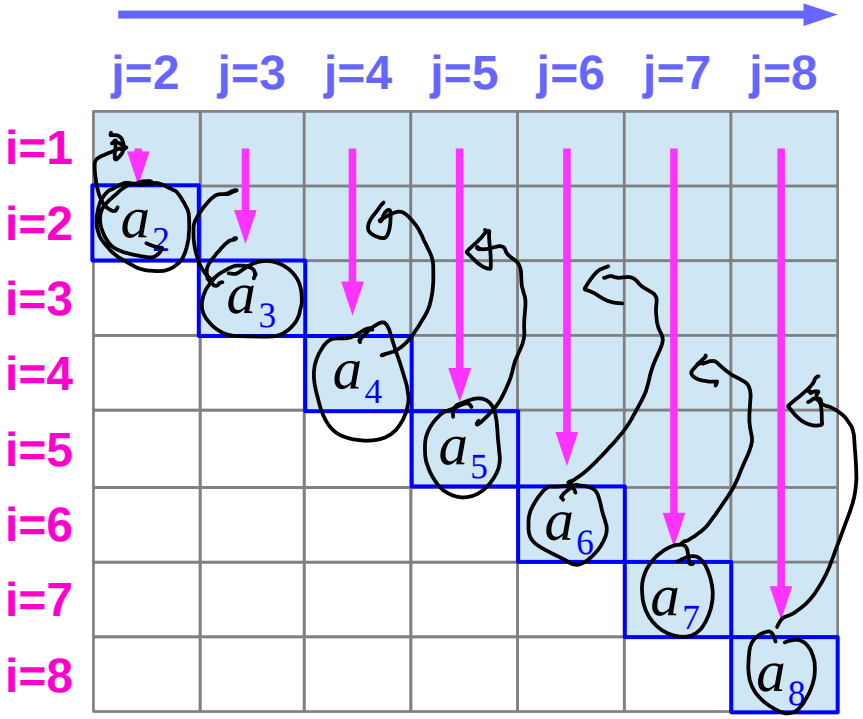
a_i **is the 1st one that is greater than** a_j



Nested loop i – inserting $\mathbf{a}_i$ at the correct position

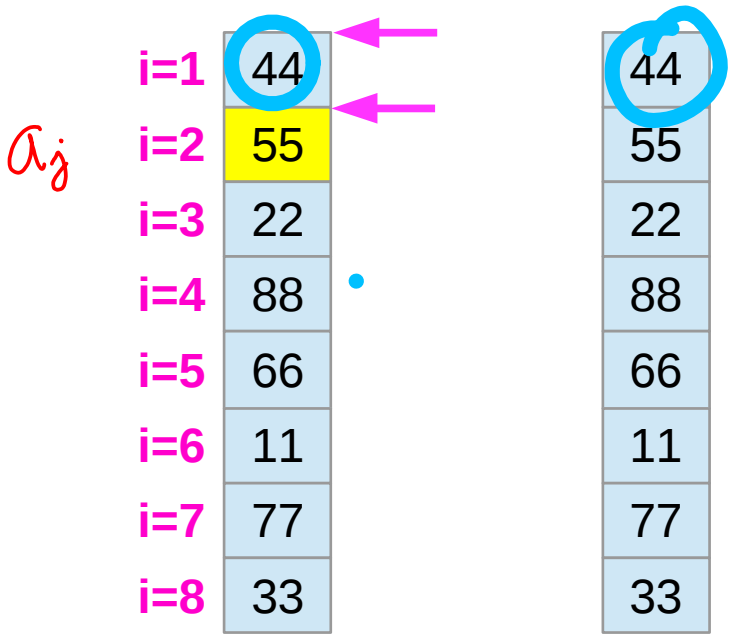


Nested loop iterations

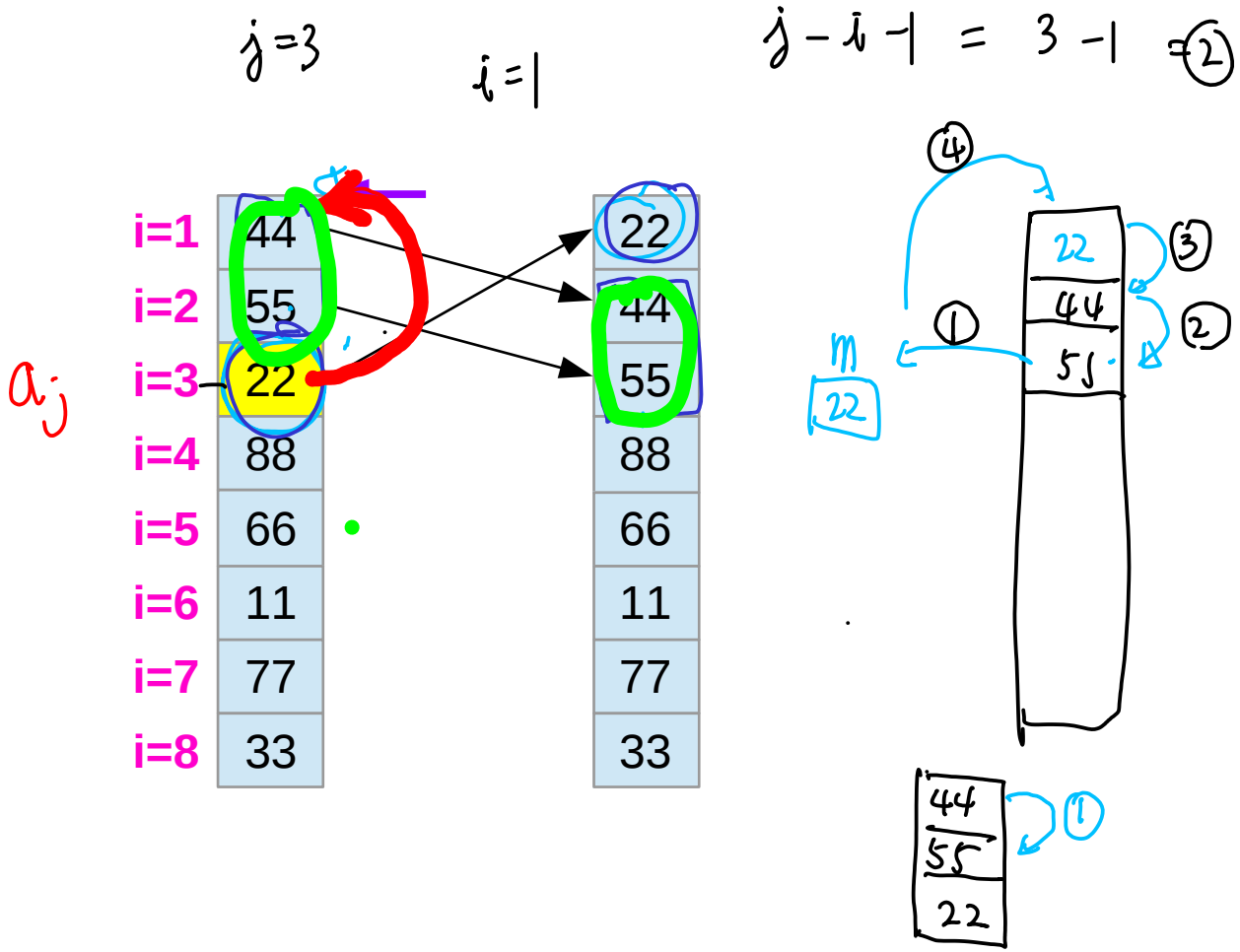


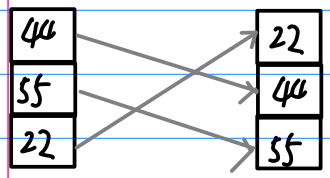

```
for j := 2 to n
    i := 1
    while aj > ai
        i := i + 1
    m := aj
    for k := 0 to j– i–1
        aj-k = aj-k-1
    ai := m
```

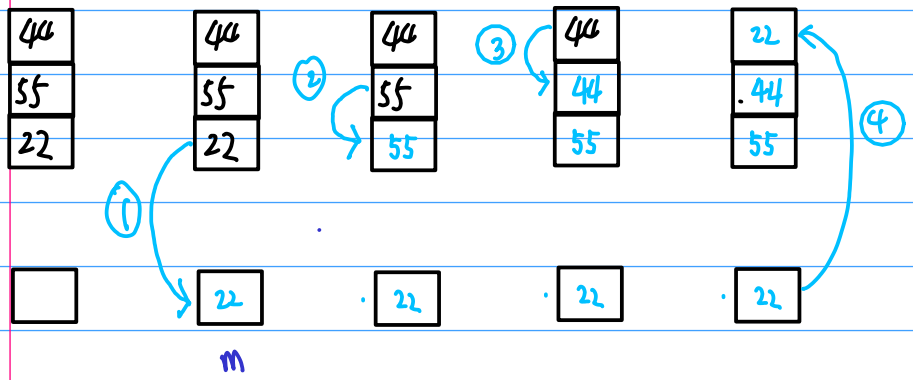
Step j=2

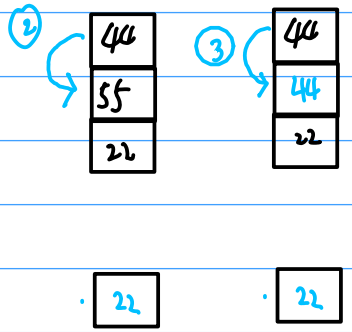

a_j
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
44
55
22
88
66
11
77
33
44
55
22
88
66
11
77
33

Step j=3

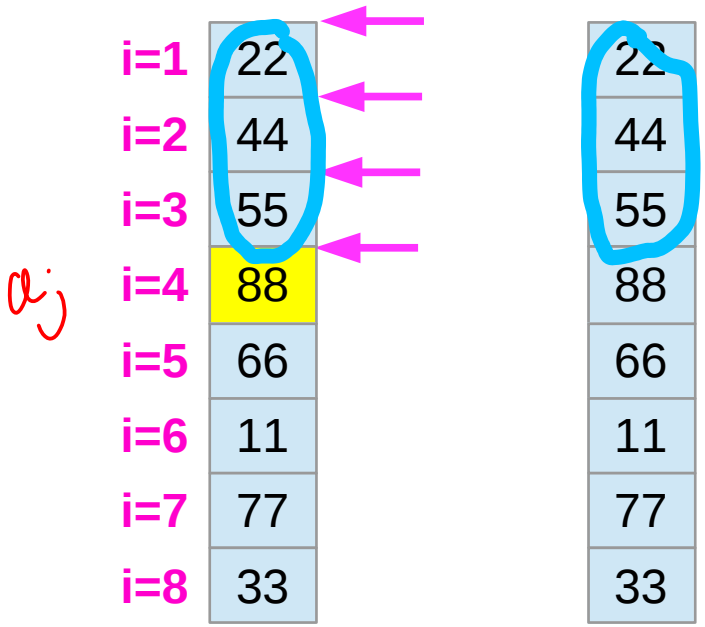

j=3
i=1
j - i - 1 = 3 - 1 = 2
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
44
55
22
88
66
11
77
33
22
44
55
88
66
11
77
33
aj
key
22
44
55
44
55
22


44
55
22
22
44
55

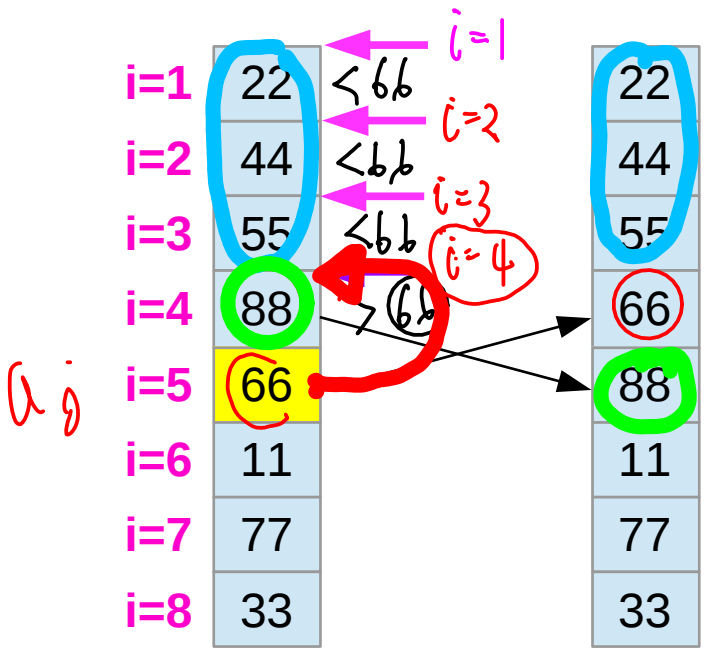

44
55
22
44
55
22
55
44
44
55
22
44
55
1
2
3
4
22
22
22
22
m


2
3
44
55
22
44
22
22
22

Step j=4

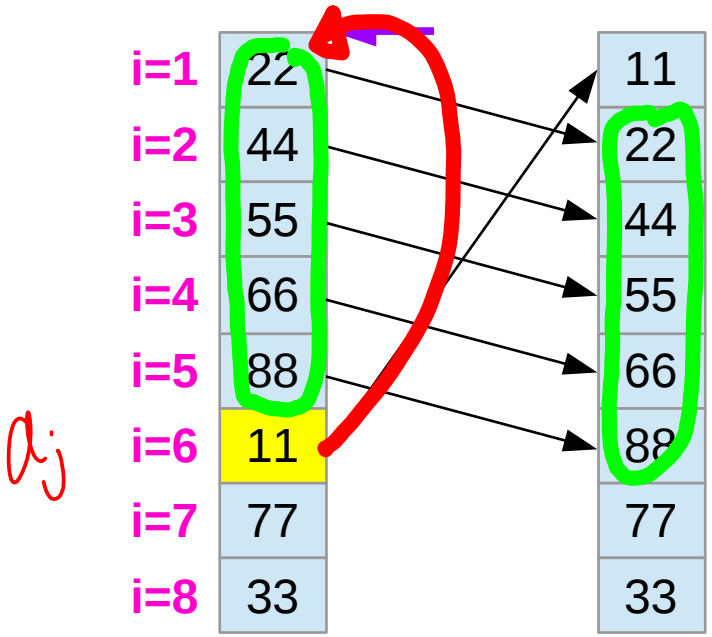


Step j=5

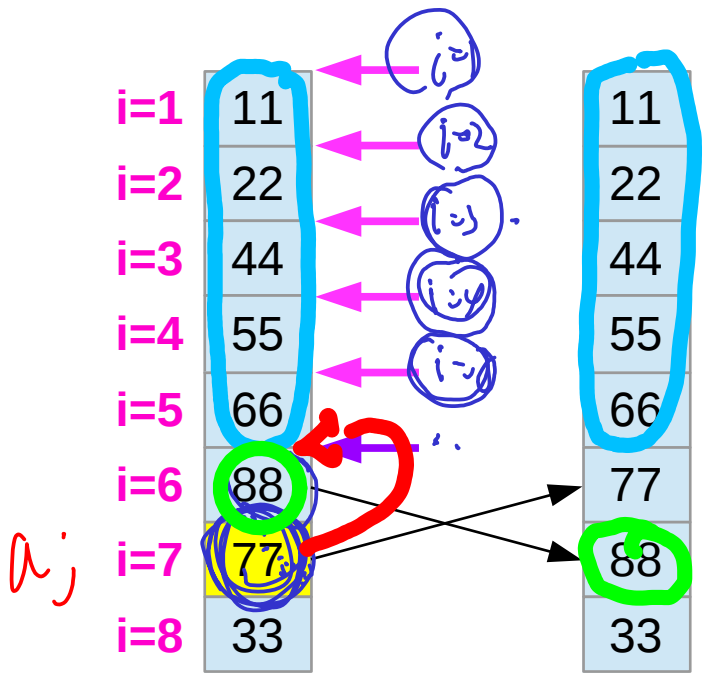



```
i := 1
while a[j] > a[i]
    i := i + 1
```

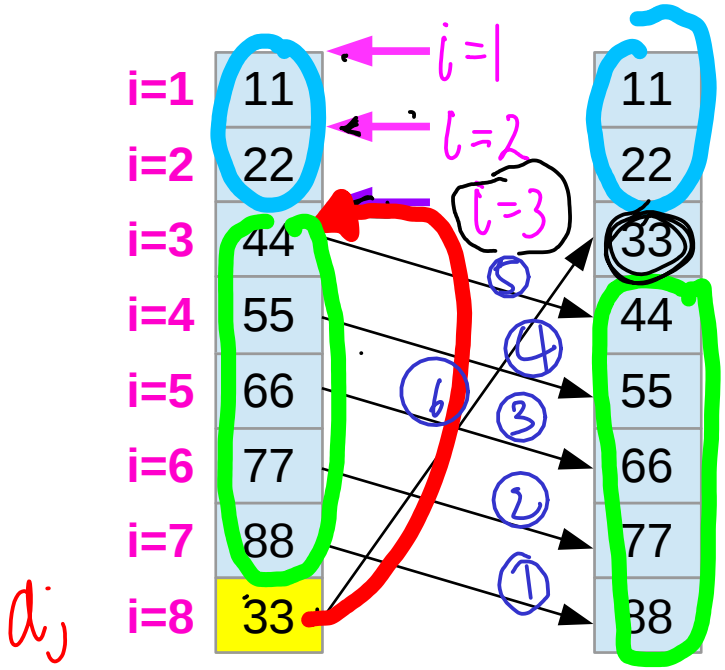
Step j=6

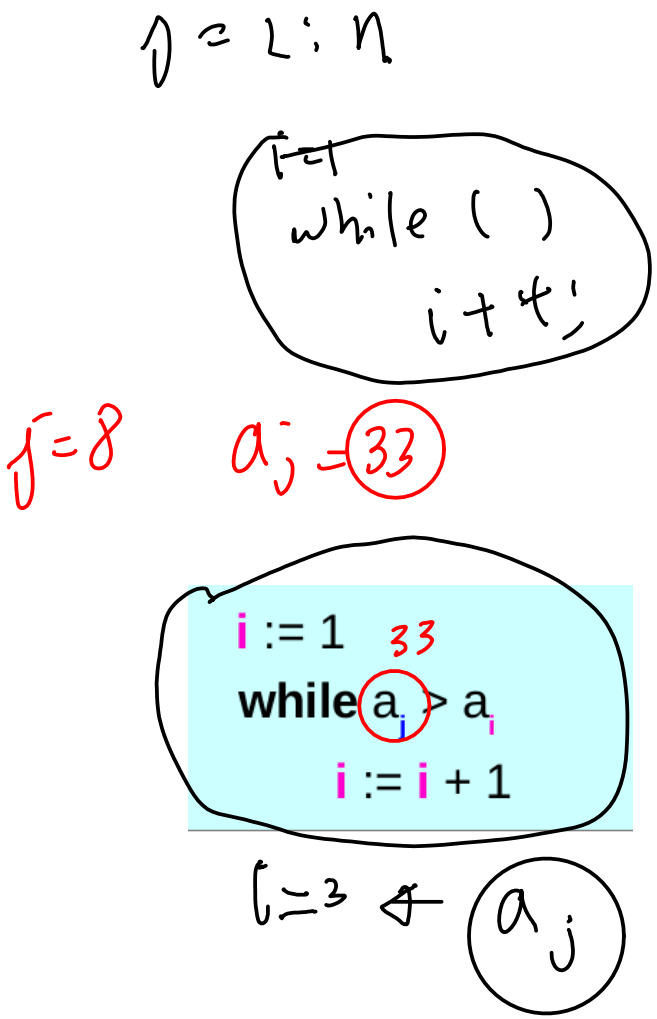


Step j=7



Step j=8


i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
11
22
44
55
66
77
88
33
11
22
33
44
55
66
77
88
i=1
i=2
i=3
1
2
3
4
5
6
aj


j = 2 : n
i=1
while ( )
i++;
j=8
aj = 33
i := 1
33
while a_j > a_i
i := i + 1
i=3
a_j

Nested loop iterations

References

[1] http://en.wikipedia.org/
[2]

Bubble 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

$i=0$

0	42	42	42	42	42	42	42	**13**
1	20	20	20	20	20	20	**13**	42
2	17	17	17	17	17	**13**	20	20
3	**13**	**13**	**13**	**13**	**13**	17	17	17
4	28	28	28	14	14	14	14	14
5	14	14	14	28	28	28	28	28
6	23	15	15	15	15	15	15	15
7	15	23	23	23	23	23	23	23
	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	

$i=1$

0	**13**	**13**	**13**	**13**	**13**	**13**	**13**
1	42	42	42	42	42	42	**14**
2	20	20	20	20	20	**14**	42
3	17	17	17	17	**14**	20	20
4	**14**	**14**	**14**	**14**	17	17	17
5	28	28	15	15	15	15	15
6	15	15	28	28	28	28	28
7	23	23	23	23	23	23	23
	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	

$i=2$

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

$i=3$

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

$i=4$

0	13	13	13	13
1	14	14	14	14
2	15	15	15	15
3	19	19	19	19
4	42	42	42	20
5	20	20	20	42
6	28	23	23	23
7	23	28	28	28
	$j=0$	$j=1$	$j=2$	

$i=5$

0	13	13	13
1	14	14	14
2	15	15	15
3	19	19	19
4	20	20	20
5	42	42	23
6	23	23	42
7	28	28	28
	$j=0$	$j=1$	

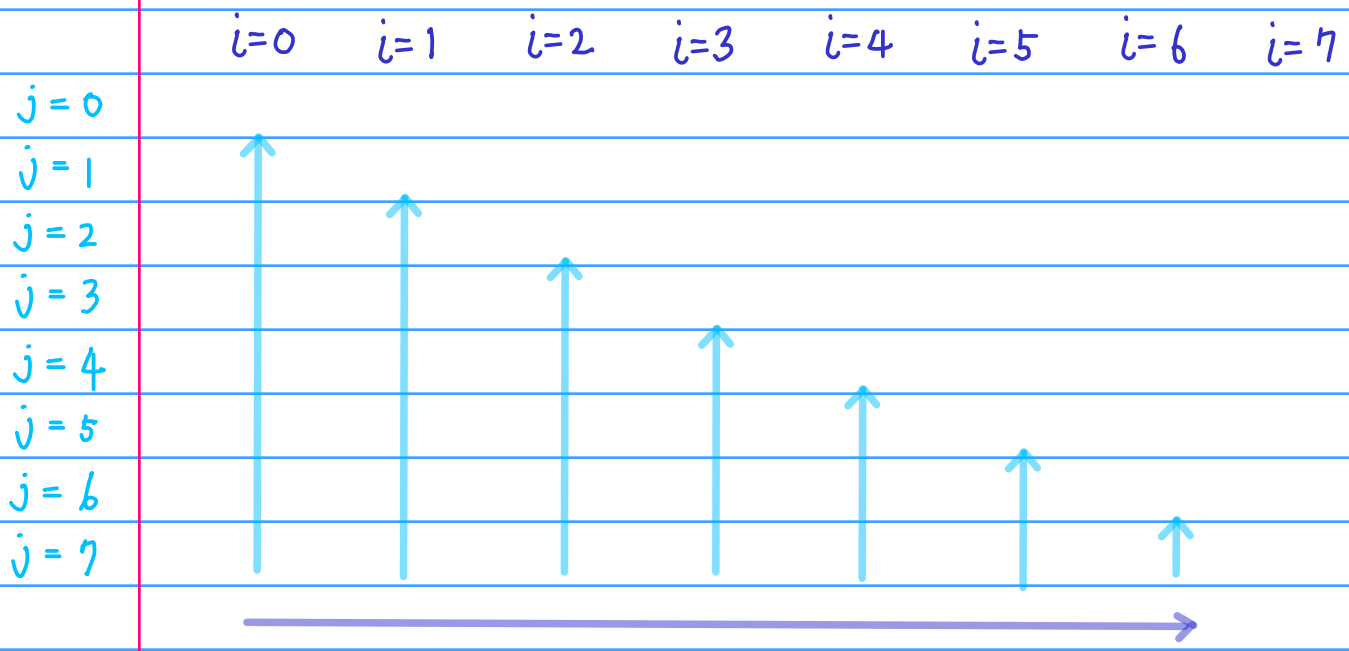
$i=6$

0	13	13
1	14	14
2	15	15
3	19	19
4	20	20
5	23	23
6	42	28
7	28	42
	$j=0$	

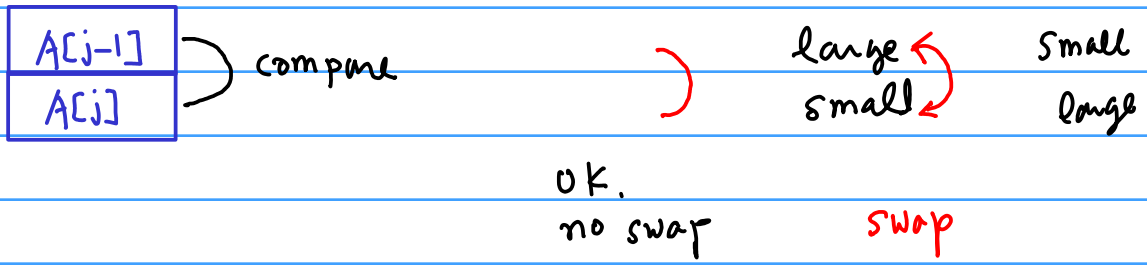
```
// Swap two elements in a generic array
template<typename E>
inline void swap(E A[], int i, int j) {
  E temp = A[i];
  A[i] = A[j];
  A[j] = temp;
}
// Random number generator functions
```

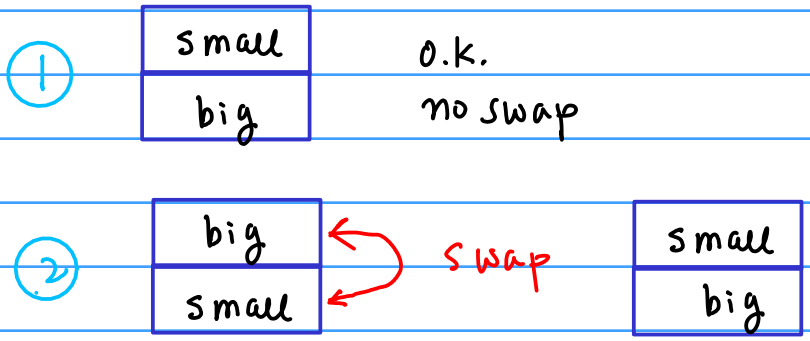
```
template <typename E, typename Comp>
void bubsort(E A[], int n) { // Bubble
  for (int i=0; i<n-1; i++)      // Bubl
    for (int j=n-1; j>i; j--)
      if (Comp::prior(A[j], A[j-1]))
        swap(A, j, j-1);
}
```

$n=8$

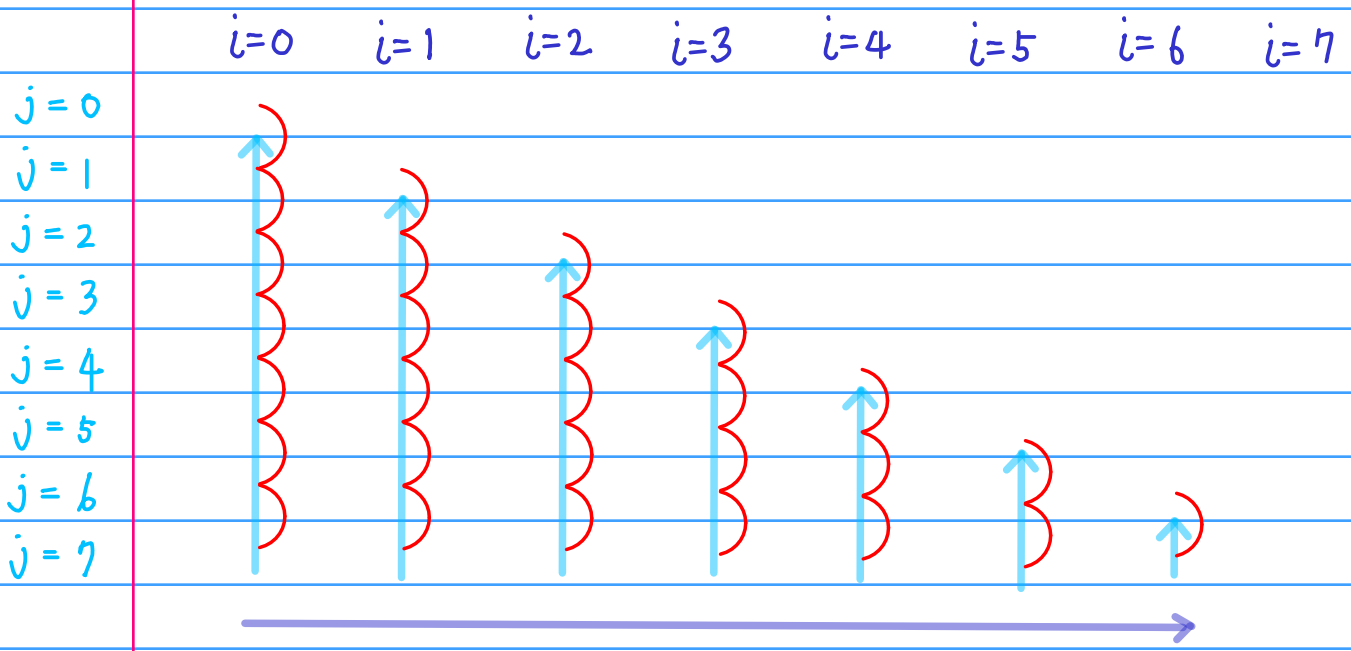


```
template <typename E, typename Comp>
void bubsort(E A[], int n) { // Bubble
  for (int i=0; i<n-1; i++)      // Bubl
    for (int j=n-1; j>i; j--)
      if (Comp::prior(A[j], A[j-1]))
        swap(A, j, j-1);
}
```

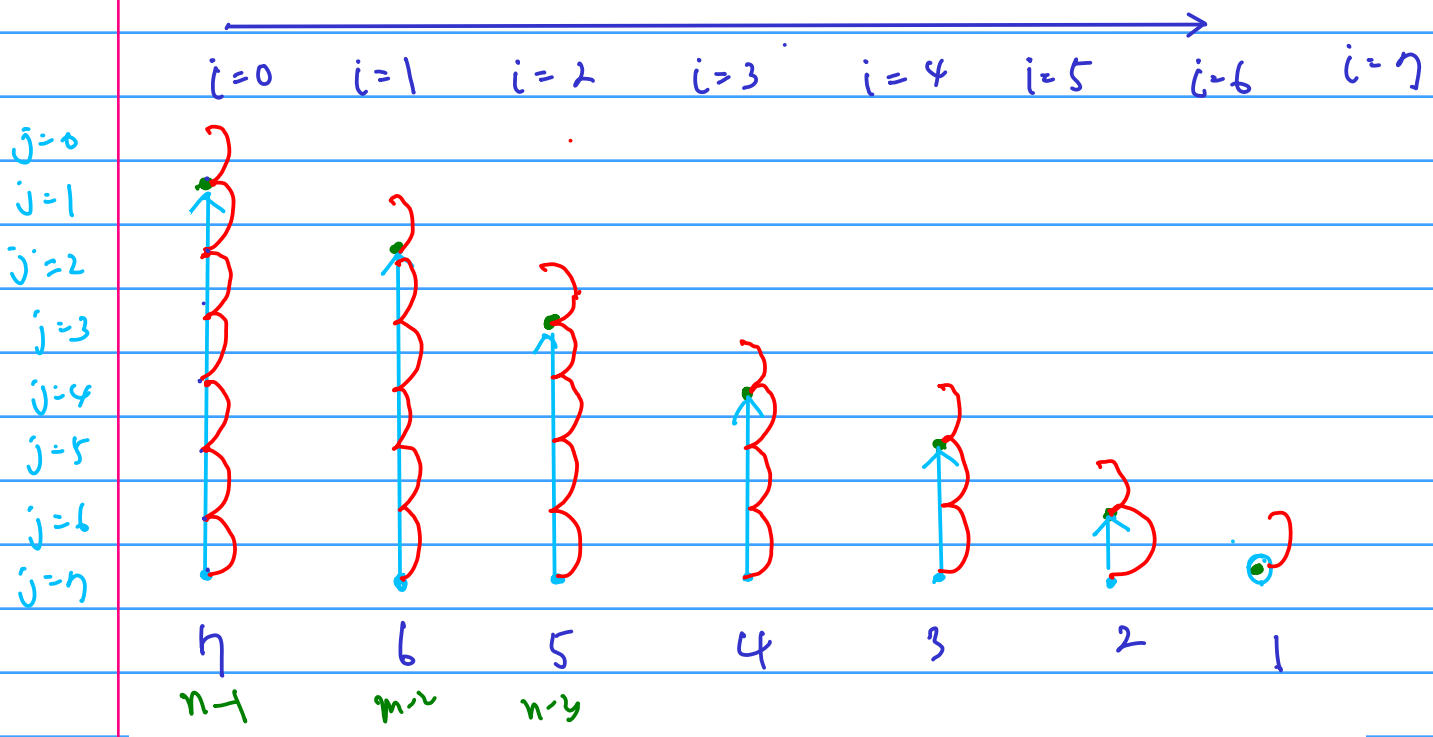


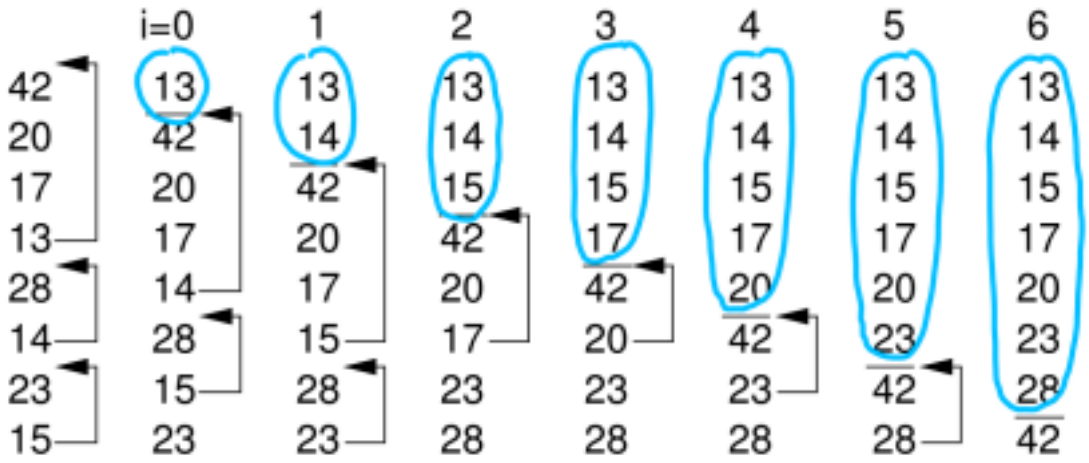


$n=8$



```
template <typename E, typename Comp>
void bubsort(E A[], int n) { // Bubble
  for (int i=0; i<n-1; i++)      // Bubl
    for (int j=n-1; j>i; j--)
      if (Comp::prior(A[j], A[j-1]))
        swap(A, j, j-1);
}
```





References

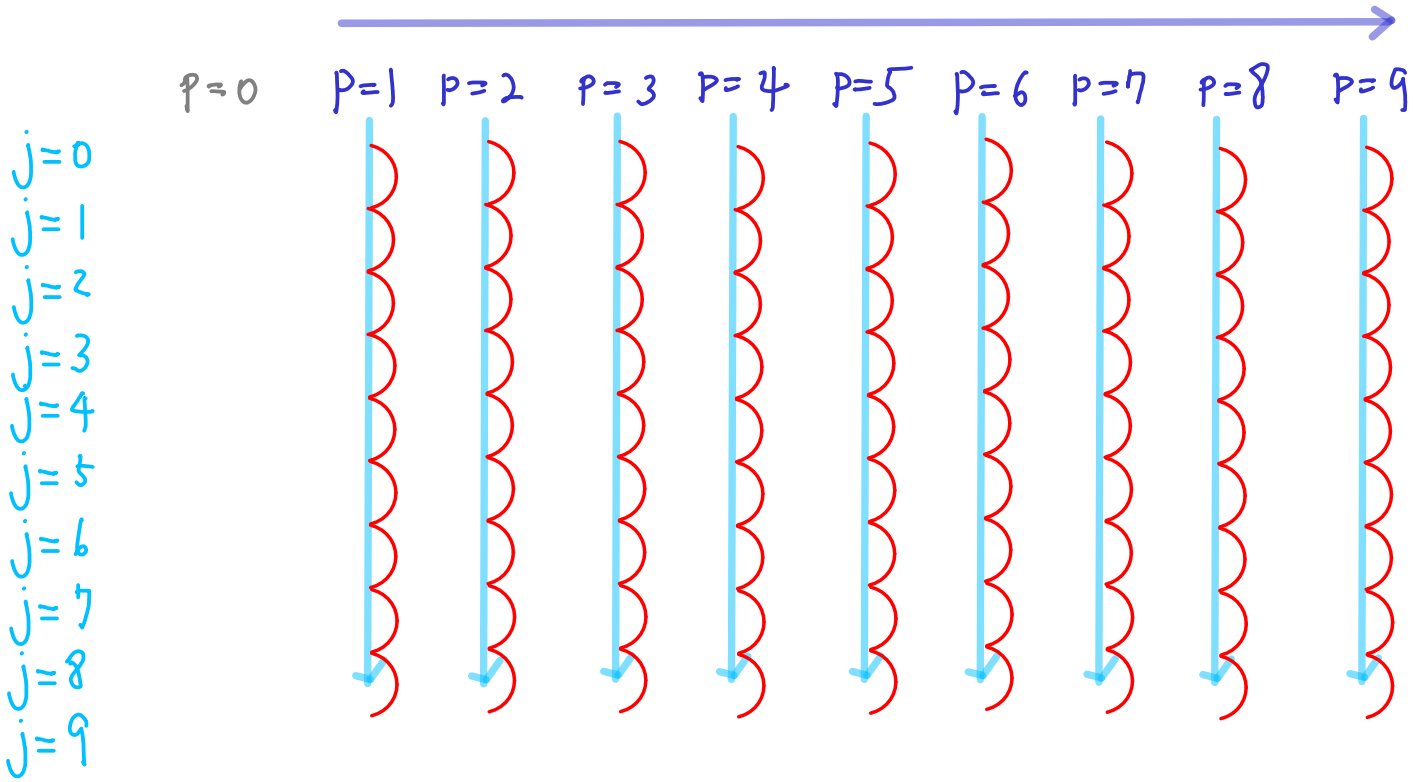
[1] http://en.wikipedia.org/
[2] http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf

```
#include <stdio.h>

void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j)
      if ( a[j] > a[j+1] )  {
        tmp = a[j];
        a[j] = a[j+1];
        a[j+1] = tmp;
      }
  }
}


int main(void) {
  int i;
  int a[] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

  bubbleSort(a, 10);


  for (i=0; i<10; ++i)
    printf("a[%d]=%d \n", i, a[i]);

}
```

```
a[0]=2
a[1]=4
a[2]=6
a[3]=8
a[4]=10
a[5]=12
a[6]=37
a[7]=45
a[8]=68
a[9]=89
```

```
a[0]=89
a[1]=68
a[2]=45
a[3]=37
a[4]=12
a[5]=10
a[6]=8
a[7]=6
a[8]=4
a[9]=2
```

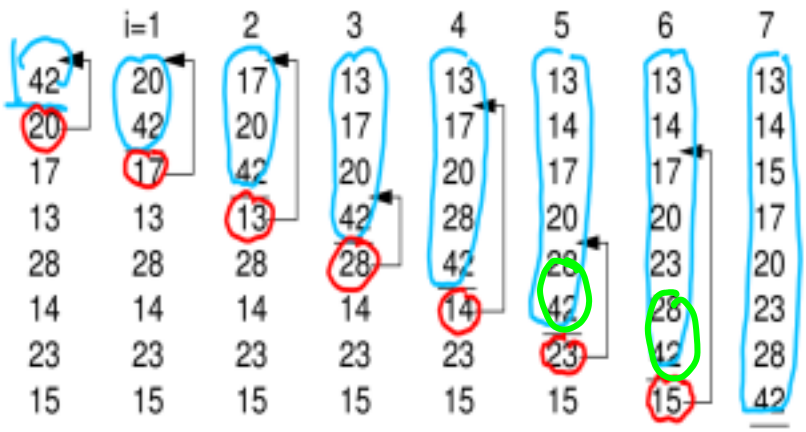
```
void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j)
      if ( a[j] > a[j+1] )  {
        tmp = a[j];
        a[j] = a[j+1];
        a[j+1] = tmp;
      }
  }
}
```



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



42

20

17

13

28

14

23

15

1-A

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

1 - B

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

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

C++ template

```
#include <stdio.h>

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


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);
  d2 = square<double>(d);

  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;

}
```

int
float
double

<float>
<double>

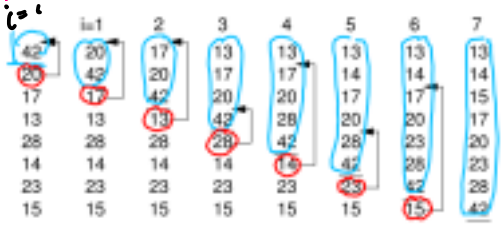
```
template <class T>
T square(T x) {
  return (x*x);
}
```

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

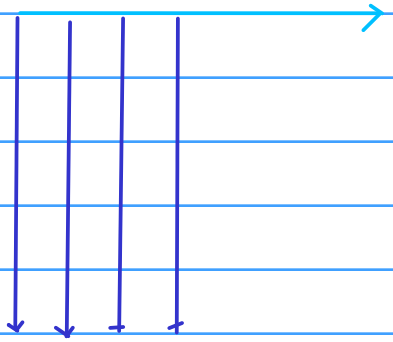
```
square<float>(f);

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);
}
```

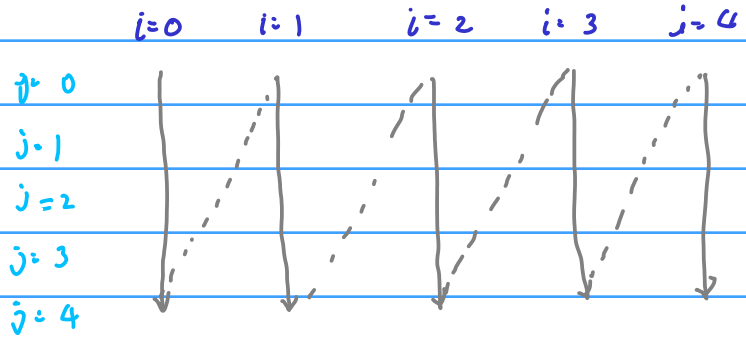
Swap

```
// 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
```

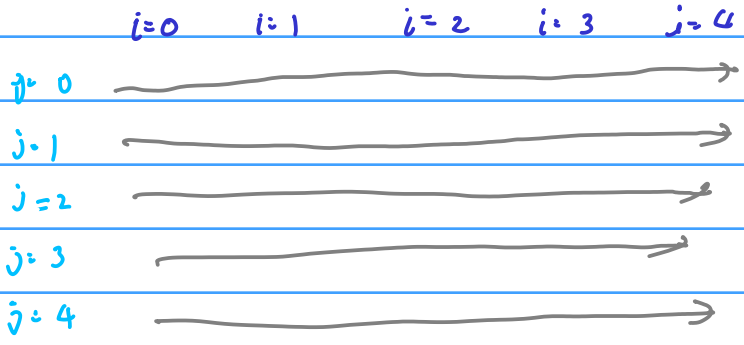



```
template <typename E, typename Comp>
void inssort(E A[], int n) { // Insertion Sort
  for (int i=1; i<n; i++)        // Insert i'th record
    for (int j=i; (j>0) && (Comp::prior(A[j], A[j-1])); j--)
      swap(A, j, j-1);
}
```

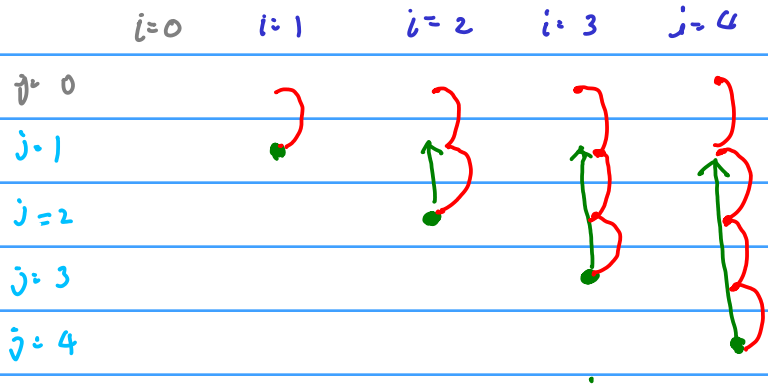




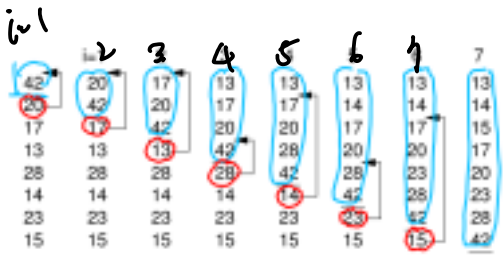

```
for (i=0; i<5; i++)
  for (j=0; j<5; j++)
```




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




```
for (i=1; i<5; ++i)
    for (j=i; j>0; --j)
        A[j] < A[j-1]
```




```
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)
        do {
            Read   A[i, j] ;
            Write  A[i, j] ;
        }
```

(2)

$(N-1) - 2 + 1$

$(N-2) \times 2$ ← Read, Write

실행되는 명령의 횟수

- Read = A
- write
- A =
- Comp A < B >

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

Quadratic time Alg.

```
#define MAX 9000000

for i = 1000 to N-20000
    for j = 2 to MAX
        do {
        Read A[i,j];
        write A[i,j];
    }
```

$N - 20000 - 1000$
$= N - 19000$

$9000000 - 2 + 1$
9999999

$(N - 19000) \times 9999999 \times 2 = O(N)$

Linear time alg.

References

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

Binary Search

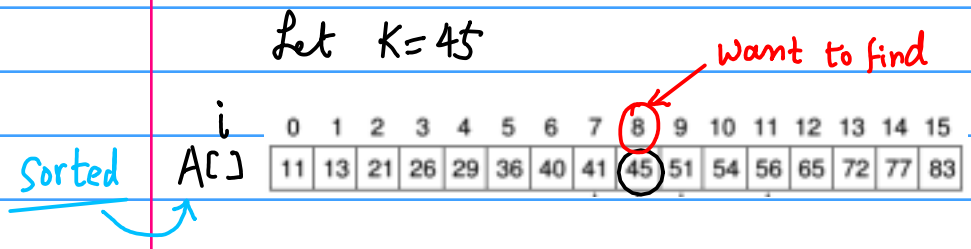
20161104

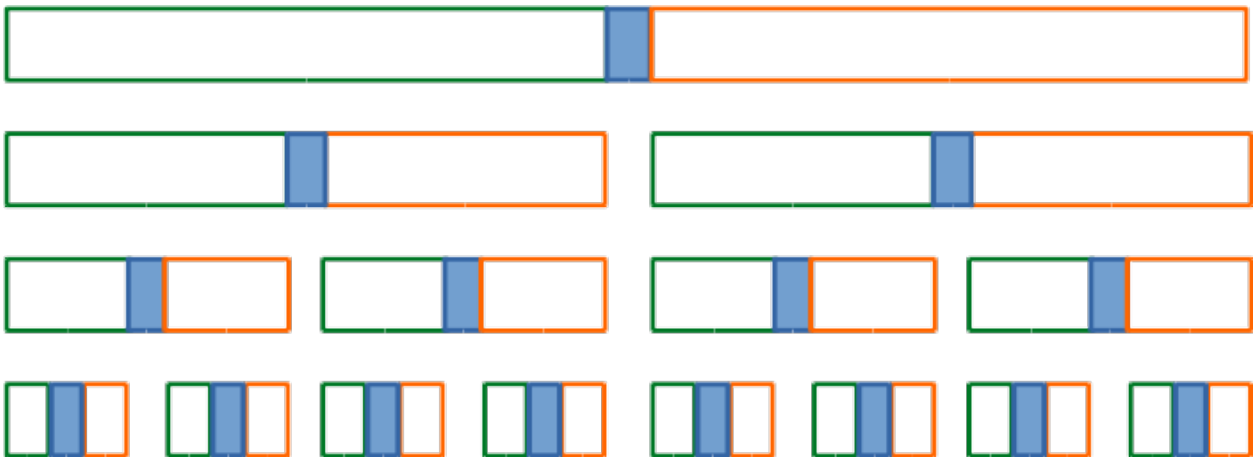
used some pictures and codes from
http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf
Data Structures and Algorithm Analysis
by Clifford A. Schaffer

Search the location of the value K (index of A[])

```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;           // l and r are beyond array bounds
  while (l+1 != r) {   // Stop when l and r meet
    int i = (l+r)/2;   // Check middle of remaining subarray
    if (K < A[i]) r = i;      // In left half
    if (K == A[i]) return i; // Found it
    if (K > A[i]) l = i;      // In right half
  }
  return n; // Search value not in A
}
```

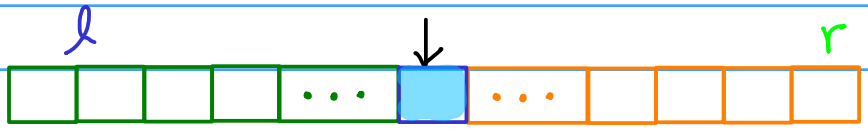
Let K=45





l : left r : right

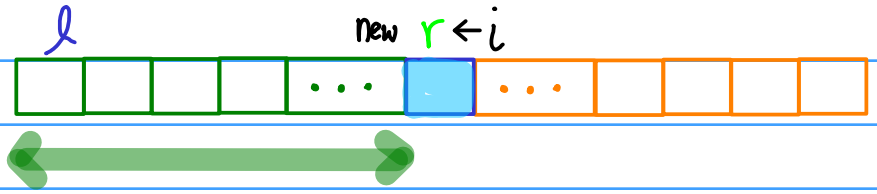
$i = \frac{1}{2}(l + r)$



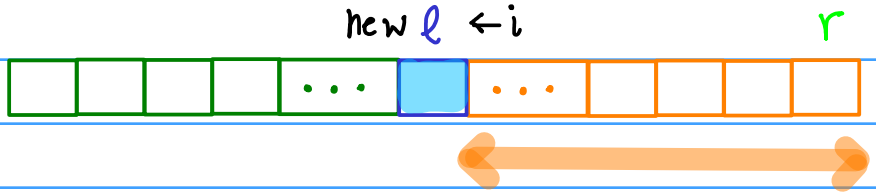
$< < < < A[i] < < < < <$ sorted already

case ① $K < A[i]$

next time search only this range



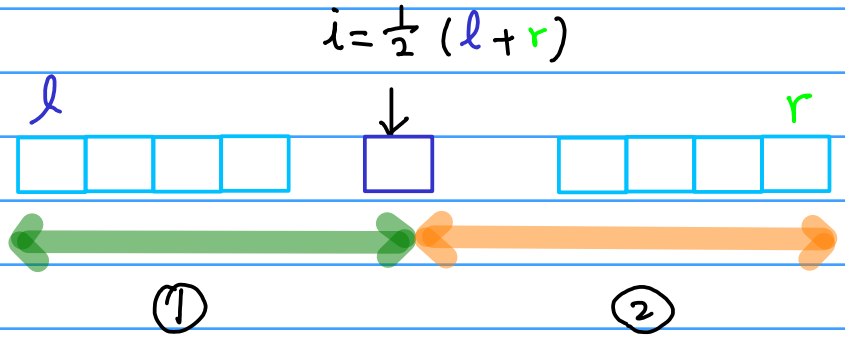
case ② $A[i] < K$



next time search only this range

case ③ $A[i] = K$

found the answer i



K < A[i] the next search will be done over (1)

K > A[i] (2)

because A[] is in the increasing order!

K < A[i] the adjusted range [l, i]

K > A[i] [i, r]

```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;             // l and r are beyond array bounds
  while (l+1 != r) {     // Stop when l and r meet
    int i = (l+r)/2;     // Check middle of remaining subarray
    if (K < A[i]) r = i;      // In left half
    if (K == A[i]) return i; // Found it
    if (K > A[i]) l = i;      // In right half
  }
  return n; // Search value not in A
}
```

Termination Condition

```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;           // l and r are beyond array bounds
  while (l+1 != r) {   // Stop when l and r meet
    int i = (l+r)/2;   // Check middle of remaining subarray
    if (K < A[i]) r = i;      // In left half
    if (K == A[i]) return i; // Found it
    if (K > A[i]) l = i;      // In right half
  }
  return n; // Search value not in A
}
```

l : left <= r : right

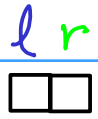
as the iteration goes on

l → moves to the right

l += 1

← r moves to the left

r -= 1



last iteration

l + 1 = r

l <= r

initial condition $l = 0$, $r = n-1$ } n element array.

Since the index (i) is used
don't have to use

new $r \leftarrow i-1$
ne $l \leftarrow i+1$

```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;           // l and r are beyond array bounds
  while (l+1 != r) {   // Stop when l and r meet
    int i = (l+r)/2;   // Check middle of remaining subarray
    if (K < A[i]) r = i;      // In left half
    if (K == A[i]) return i; // Found it
    if (K > A[i]) l = i;      // In right half
  }
  return n; // Search value not in A
}
```

```
int binary (int A[], int n, int K) {
        int l = 0;
        int r = n-1;
        int i ;
        while ( l <= r) {
                i = (l+r)/2;
                if (K  < A[i]) r = i-1;
                if (K == A[i]) return i;
                if (K  > A[i]) l = i+1;
        }
        return n;
}
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$l = 0 \quad r = 15 \quad i = \frac{1}{2}(l + r) = \frac{1}{2}(0 + 15) = 7$

$A[7] = 41 < 45$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

22

$l = 7 \quad r = 15 \quad i = \frac{1}{2}(l + r) = \frac{1}{2}(7 + 15) = 11$

$45 < A[11] = 56$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$l = 7 \quad r = 11 \quad i = \frac{1}{2}(l + r) = \frac{1}{2}(7 + 11) = 9$

$45 < A[9] = 51$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$l = 7 \quad r = 9 \quad i = \frac{1}{2}(l + r) = \frac{1}{2}(7 + 9) = 8$

$A[8] = 45$

Index = 8 !

References

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

```
#include <stdio.h>

void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j)
      if ( a[j] > a[j+1] )  {
        tmp = a[j];
        a[j] = a[j+1];
        a[j+1] = tmp;
      }
  }
}


int main(void) {
  int i;
  int a[] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

  bubbleSort(a, 10);


  for (i=0; i<10; ++i)
    printf("a[%d]=%d \n", i, a[i]);

}
```

```
a[0]=2
a[1]=4
a[2]=6
a[3]=8
a[4]=10
a[5]=12
a[6]=37
a[7]=45
a[8]=68
a[9]=89
```

```
a[0]=89
a[1]=68
a[2]=45
a[3]=37
a[4]=12
a[5]=10
a[6]=8
a[7]=6
a[8]=4
a[9]=2
```

```
void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j)
      if ( a[j] > a[j+1] )  {
        tmp = a[j];
        a[j] = a[j+1];
        a[j+1] = tmp;
      }
  }
}
```