

# 제 41강

## 정렬

교재: x

# 목차

## 1. 정렬

1. 선택 정렬
2. 삽입 정렬
3. 버블 정렬

# 알고리즘

문제를 해결하기 위한 절차적 해결 과정

## 정렬 알고리즘

데이터를 순서대로 나열하기 위한 절차적인 과정

## 정렬 알고리즘의 종류

1. 선택정렬(Selection Sort)
2. 삽입정렬(Insertoin Sort)
3. 버블정렬(Bublle Sort)

# 1. 선택정렬

: 최소값 혹은 최대값을 선택해서 가장 앞에다가 위치하여 선택할 위치를 이동하며 정렬하는 방법

## 장점

구현이 쉽다.

## 단점

다른 정렬에 비해 시간이 오래걸린다.

# 1. 선택정렬

## 기본 로직

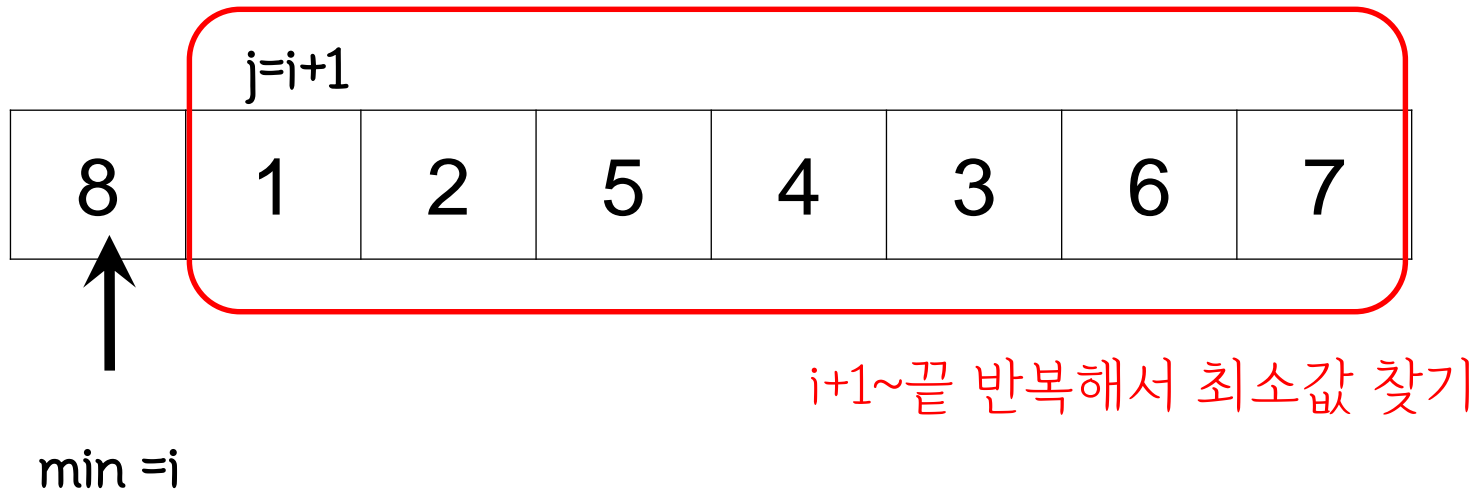
- 1) 정렬되지 않은 인덱스의 맨 앞에서부터 이를 포함한 그 이후의 값 중 가장 작은 값을 찾아 간다.
- 2) 가장 작은 값을 찾으면 그 값을 현재 인덱스의 값과 바꿔준다.
- 3) 다음 인덱스로 이동하여 위 과정을 반복한다.

## 시간복잡도

$$O(n^2)$$

# 1. 선택정렬

1) 정렬되지 않은 인덱스의 맨 앞에서부터 이를 포함한 그 이후의 값 중 가장 작은 값을 찾아간다.

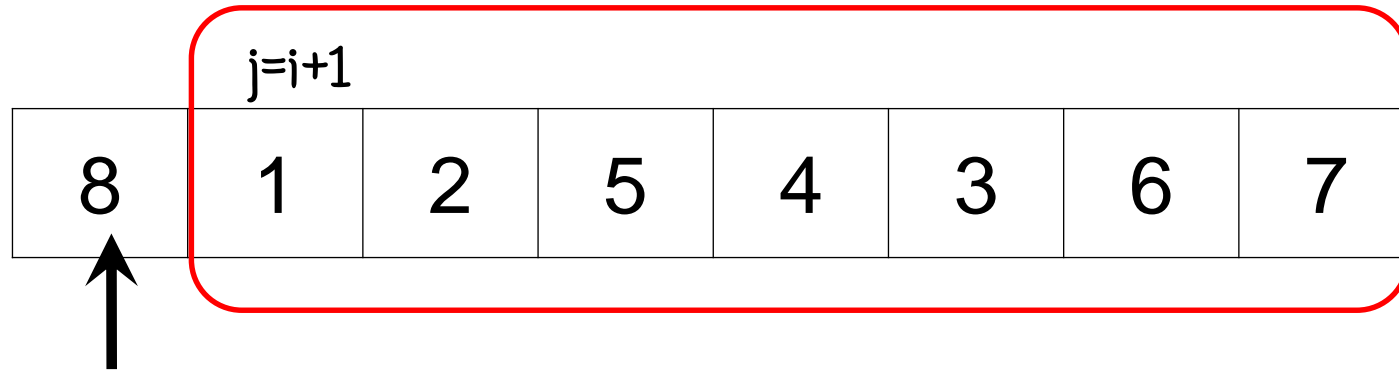


2) 가장 작은 값을 찾으면 그 값을 현재 인덱스의 값과 바꿔준다.

$ar[min] \leftrightarrow ar[i]$

3) 다음 인덱스로 이동하여 위 과정을 반복한다.

# 1. 선택정렬



$\text{min} = i$

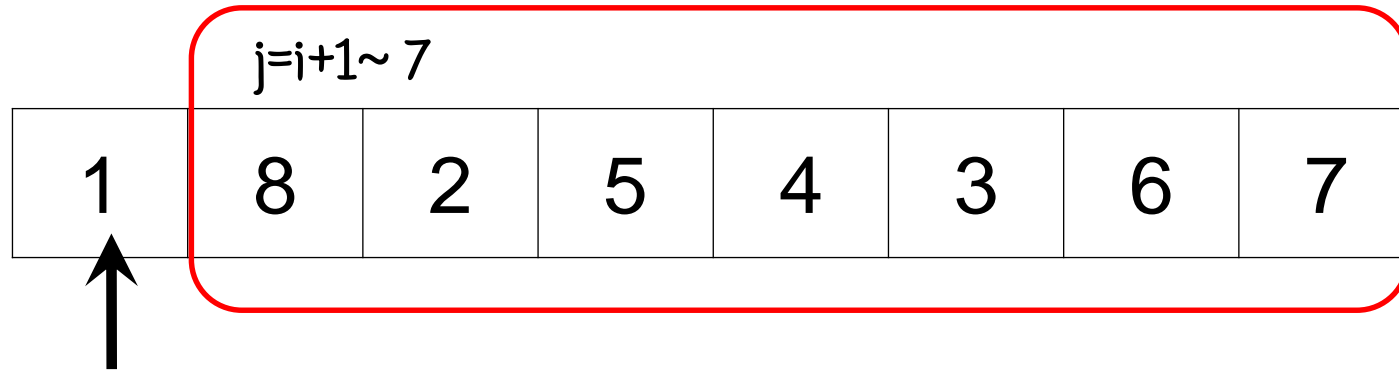
1)  $\text{min}: 0, j: 1$

만약,  $\text{ar}[0] > \text{ar}[1]$  이라면,  $\text{ar}[0] \leftrightarrow \text{ar}[1]$

→

1	8	2	5	4	3	6	7
---	---	---	---	---	---	---	---

# 1. 선택정렬



$\text{min} = i$

2)  $\text{min}: 0, j: 2$

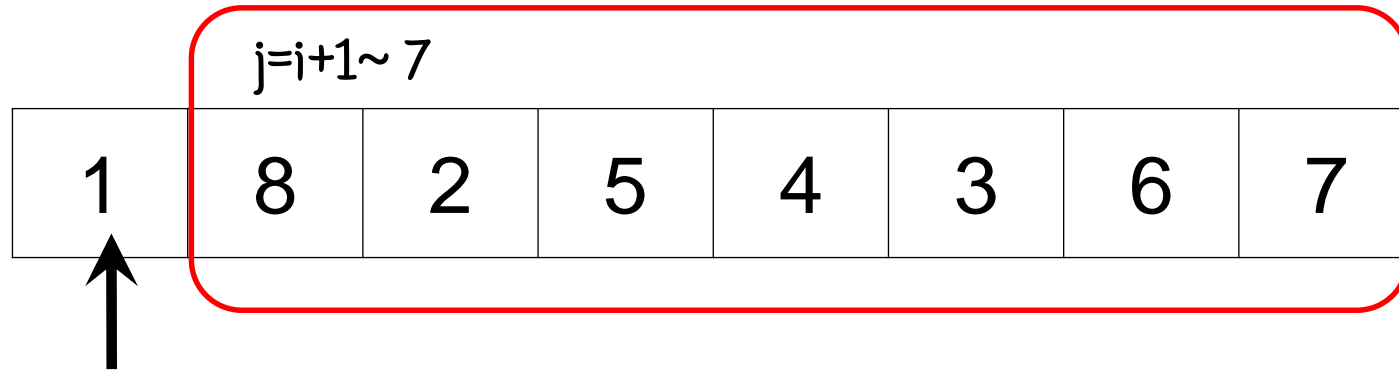
만약,  $\text{ar}[0] > \text{ar}[2]$  이라면,  $\text{ar}[0] \leftrightarrow \text{ar}[2]$  변경x

→

1	8	2	5	4	3	6	7
---	---	---	---	---	---	---	---



# 1. 선택정렬



$\text{min} = i$

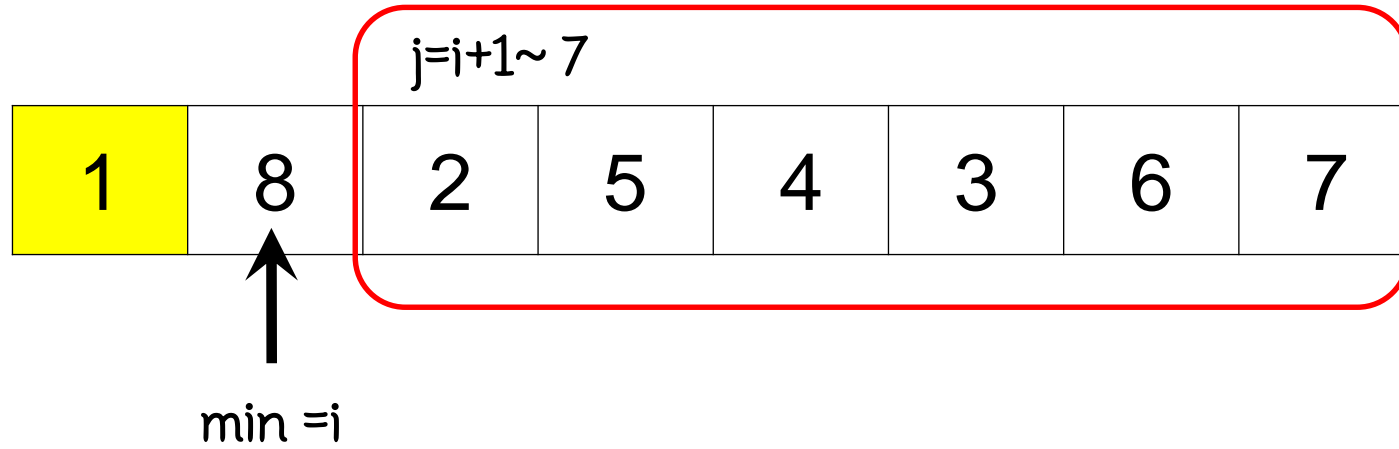
2)  $\text{min}: 0, j: 3$

만약,  $\text{ar}[0] > \text{ar}[3]$  이라면,  $\text{ar}[0] \leftrightarrow \text{ar}[3]$  변경x

→

1	8	2	5	4	3	6	7
---	---	---	---	---	---	---	---

# 1. 선택정렬

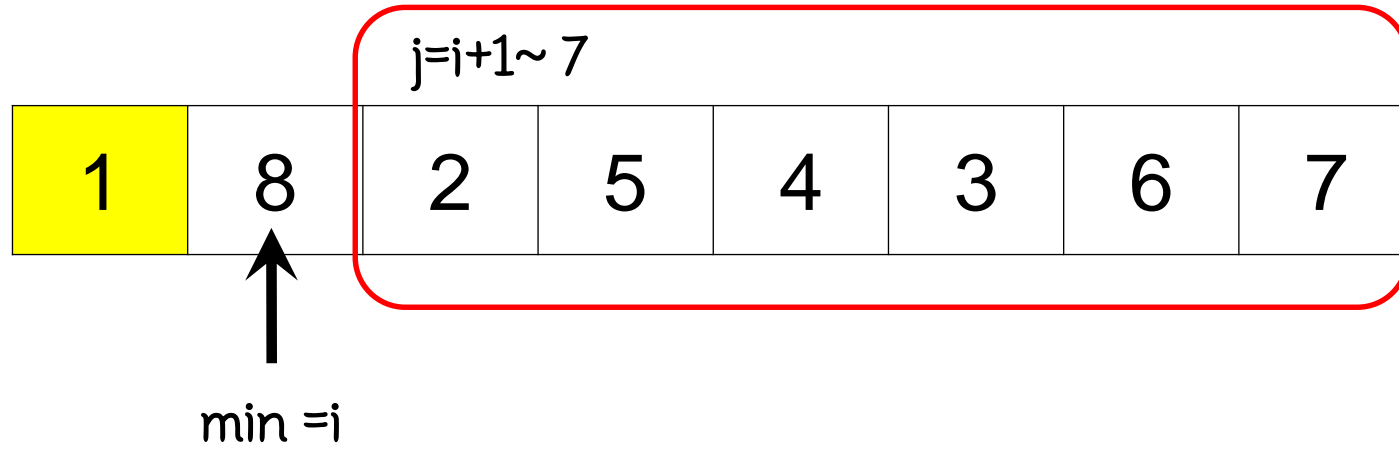


1)  $\text{min}: 1, j: 2$

만약,  $\text{ar}[1] > \text{ar}[2]$  이라면,  $\text{ar}[1] \leftrightarrow \text{ar}[2]$



# 1. 선택정렬

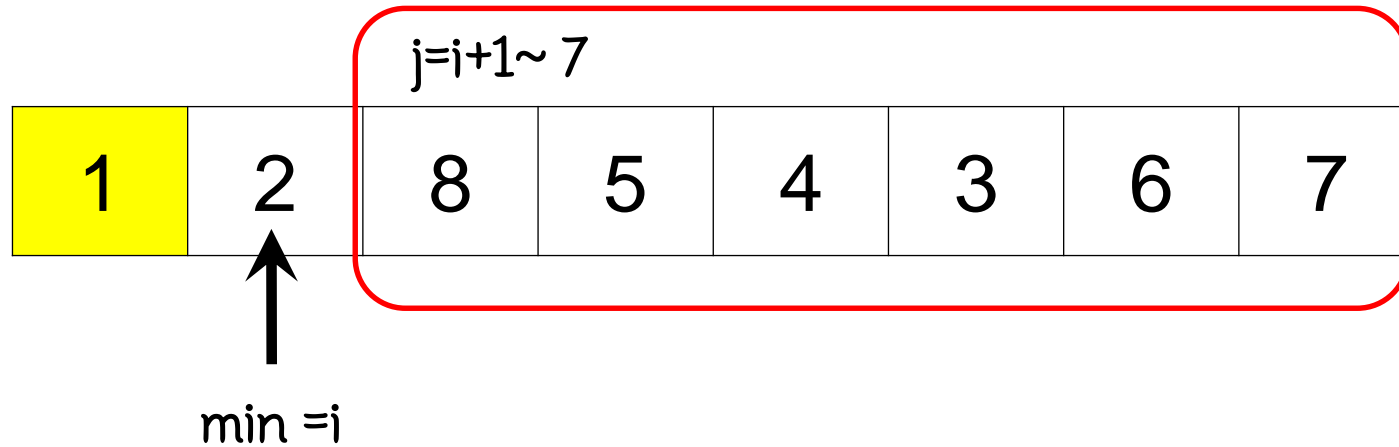


1) `min: 1, j: 2`

만약, `ar[1] > ar[2]` 이라면, `ar[1] <-> ar[2]`



# 1. 선택정렬

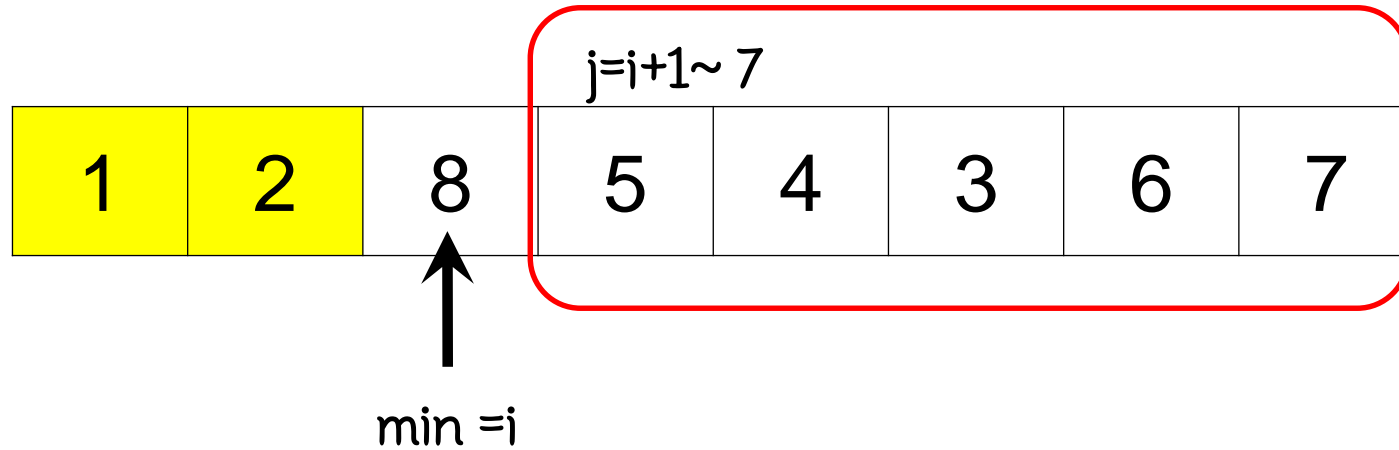


1) min: 1, j: 3

만약,  $ar[1] > ar[3]$  이라면,  $ar[1] \leftrightarrow ar[3]$  변경x

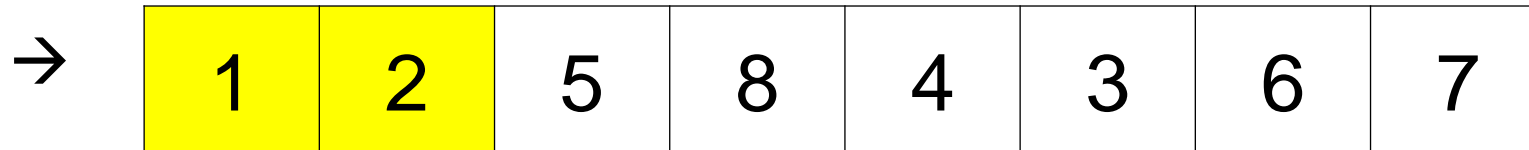


# 1. 선택정렬

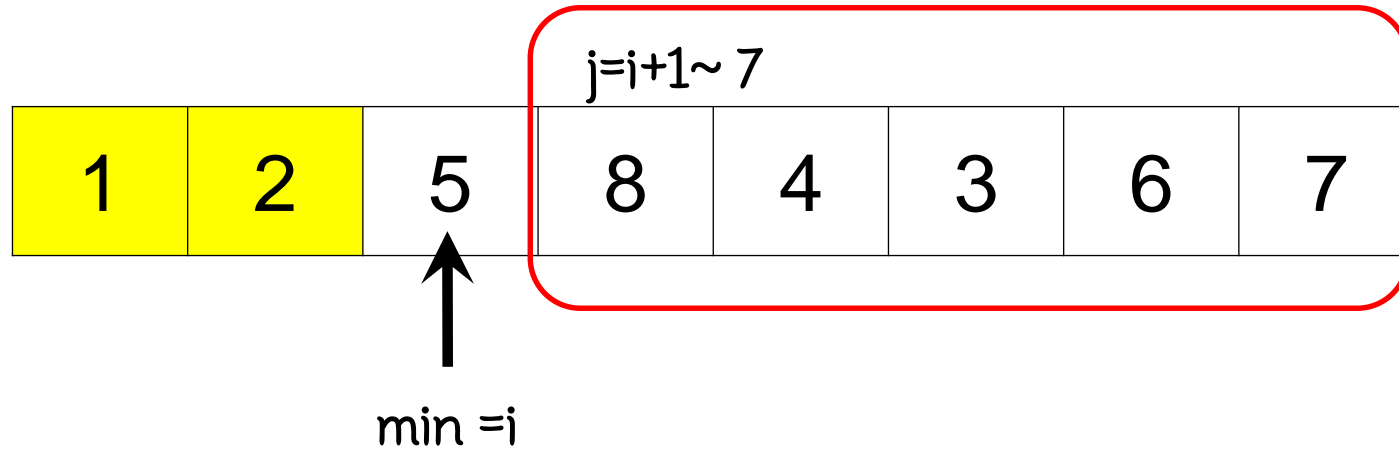


1)  $\text{min}: 2, j: 3$

만약,  $\text{ar}[2] > \text{ar}[3]$  이라면,  $\text{ar}[2] \leftrightarrow \text{ar}[3]$

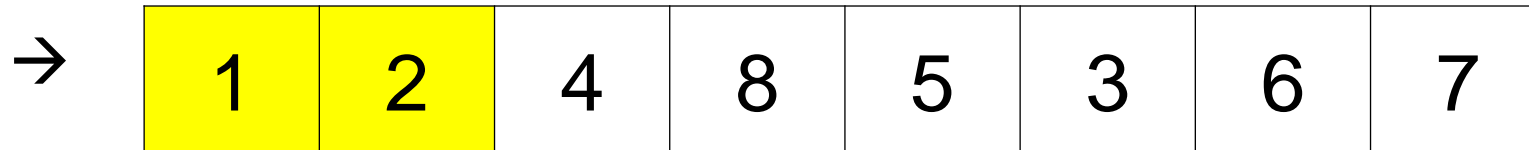


# 1. 선택정렬

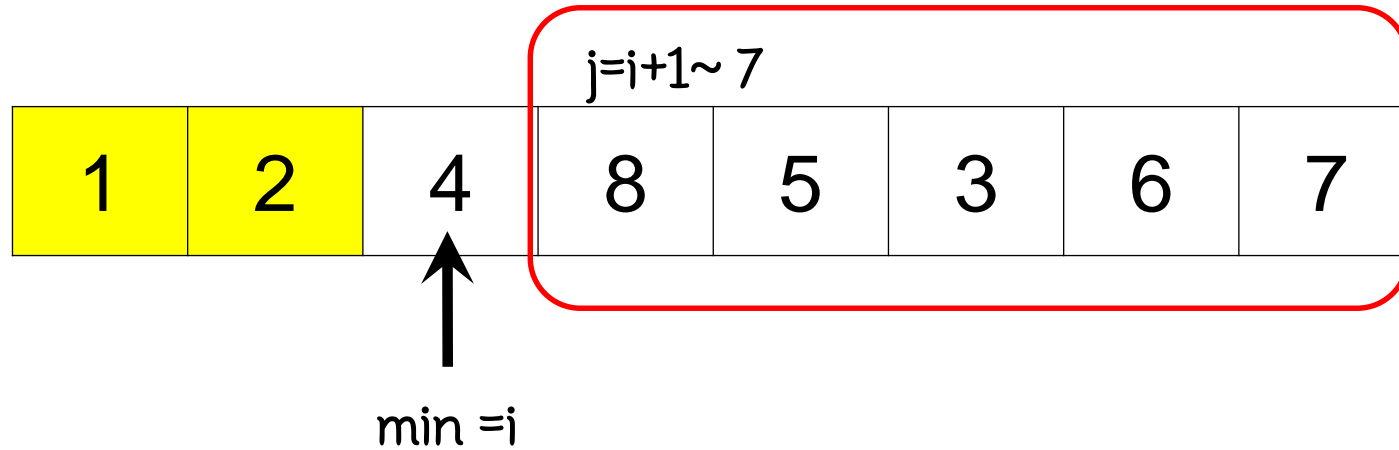


1)  $\text{min}: 2, j: 4$

만약,  $\text{ar}[2] > \text{ar}[4]$  이라면,  $\text{ar}[2] \leftrightarrow \text{ar}[4]$

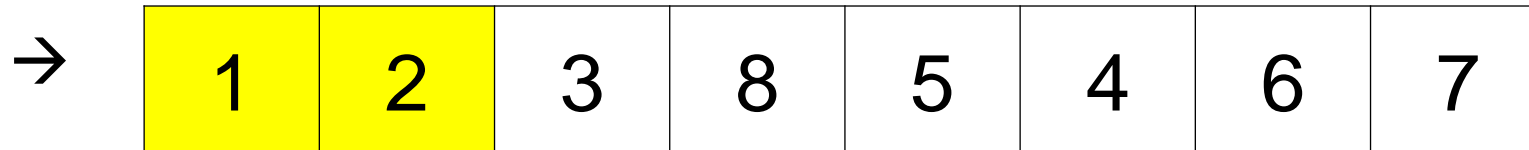


# 1. 선택정렬

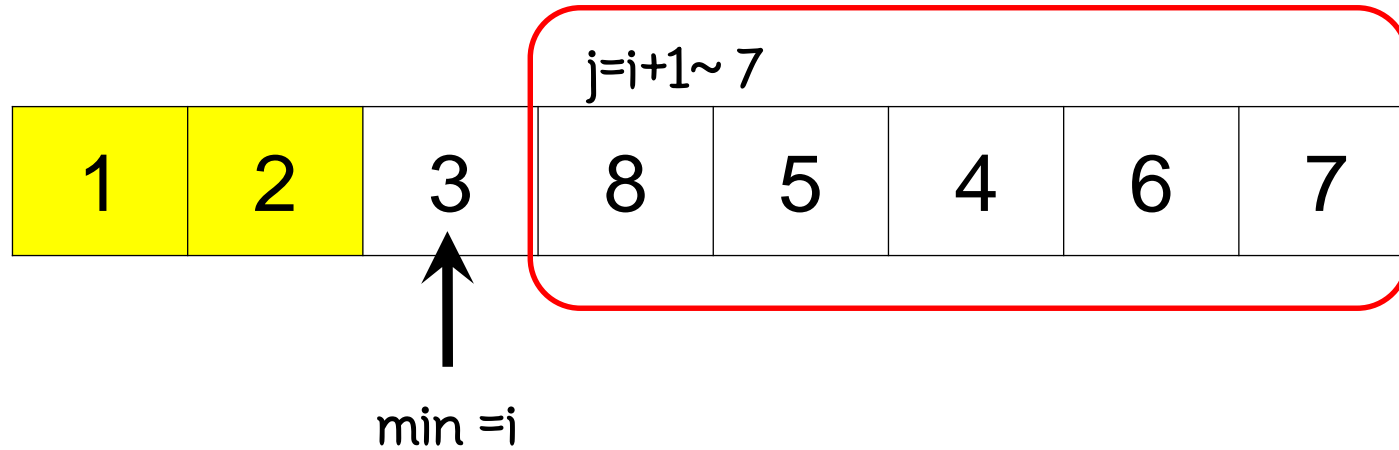


1)  $\text{min}: 2, j: 5$

만약,  $\text{ar}[2] > \text{ar}[5]$  이라면,  $\text{ar}[2] \leftrightarrow \text{ar}[5]$

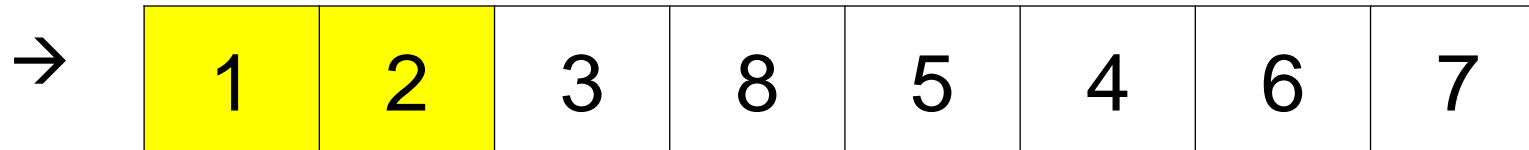


# 1. 선택정렬



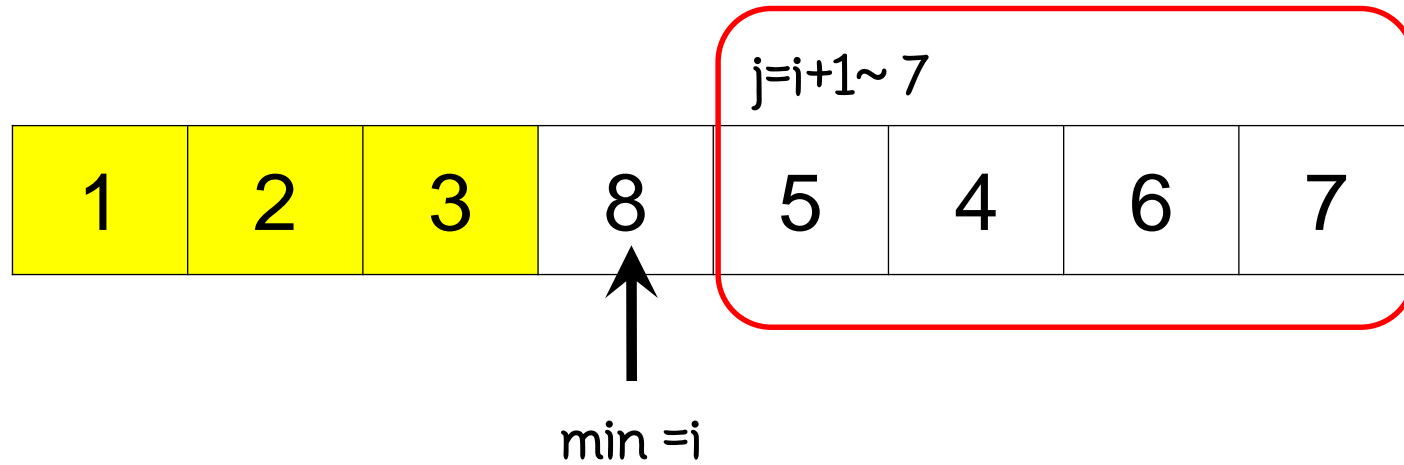
1)  $\text{min}: 2, j: 6$

만약,  $\text{ar}[2] > \text{ar}[5]$  이라면,  $\text{ar}[2] \leftrightarrow \text{ar}[6]$  변경x





# 1. 선택정렬



1)  $\text{min}: 3, j: 4$

만약,  $\text{ar}[3] > \text{ar}[4]$  이라면,  $\text{ar}[3] \leftrightarrow \text{ar}[4]$



# 1. 선택정렬

## <실습> SelectionSort.java

```
public class SelectionSort {  
    public static void main(String[] args) {  
        int ar[] = {8,1,2,5,4,3,6,7};  
        int min,tmp;  
        for(int i=0;i<ar.length-1;i++) {  
            min=i;  
            for(int j=i+1;j<ar.length;j++) {  
                if(ar[min]>ar[j]) {  
                    tmp=ar[min];  
                    ar[min]=ar[j];  
                    ar[j]=tmp;  
                }  
            }  
        }  
        for(int i=0;i<ar.length;i++) {  
            System.out.print(ar[i]+" ");  
        }  
        System.out.println();  
    }  
}
```

```
int ar[] = {8,1,2,5,4,3,6,7};
```

8	1	2	5	4	3	6	7
---	---	---	---	---	---	---	---

i: 최소값을 위치 시킬 가장 앞에 있는 idx

j: 맨 앞에 있는 idx(i)와 비교할 idx

```
for(int i=0;i<ar.length-1;i++) {
```

→ i의 범위: 0~ 마지막 idx-1

```
for(int j=i+1;j<ar.length;j++) {
```

→ j의 범위: i+1~ 마지막 idx

# 1. 선택정렬

## <실습> SelectionSort.java

```
public class SelectionSort {  
    public static void main(String[] args) {  
        int ar[] = {8,1,2,5,4,3,6,7};  
        int min,tmp;  
        for(int i=0;i<ar.length-1;i++) {  
            min=i;  
            for(int j=i+1;j<ar.length;j++) {  
                if(ar[min]>ar[j]) {  
                    tmp=ar[min];  
                    ar[min]=ar[j];  
                    ar[j]=tmp;  
                }  
            }  
        }  
        for(int i=0;i<ar.length;i++) {  
            System.out.print(ar[i]+" ");  
        }  
        System.out.println();  
    }  
}
```

```
if(ar[min]>ar[j]) {  
    tmp=ar[min];  
    ar[min]=ar[j];  
    ar[j]=tmp;  
}
```

min 위치의 값 이랑 j 위치의 값을 비교해서  
ar[min] < ar[j] 라면 swap!

## 2. 삽입 정렬

: 지정한 값의 삽입할 위치를 찾아 정렬하는 방법

(a)

3	7	2	5	1	4
---	---	---	---	---	---

(b)

3	7	2	5	1	4
---	---	---	---	---	---

(c)

2	3	7	5	1	4
---	---	---	---	---	---

(d)

2	3	5	7	1	4
---	---	---	---	---	---

(e)

1	2	3	5	7	4
---	---	---	---	---	---

(f)

1	2	3	4	5	7
---	---	---	---	---	---

## 2. 삽입 정렬

### 기본 로직

현재위치:i, 비교위치:j

- |                        |          |
|------------------------|----------|
| 1) 삽입정렬은 두 번째 idx부터 시작 | i=1      |
| 2) 현재idx는 별도의 변수에 저장   | idx=i    |
| 3) 비교 idx=현재idx -1     | j=i-1    |
| 4) idx값 < 비교idx의 값 :   | idx<-> j |

idx=j

\*idx랑 j 값을 바꾸므로, 현재idx의 값으로 계속 비교해줘야함

- 5) idx > 비교idx의 값: 비교idx--

## 2. 삽입 정렬

비교idx



8	1	2	5	4	3	6	7
---	---	---	---	---	---	---	---



현재idx

비교idx: -->

현재idx: -->

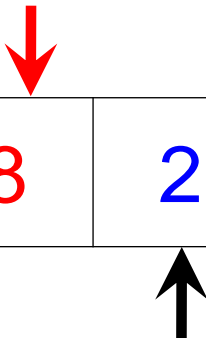
4) idx값 < 비교idx의 값 :

idx <-> j

idx = j

1	8	2	5	4	3	6	7
---	---	---	---	---	---	---	---

## 2. 삽입 정렬



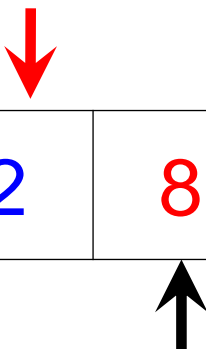
1	8	2	5	4	3	6	7
---	---	---	---	---	---	---	---

비교idx: -->

현재idx: -->

4) idx값 < 비교idx의 값 :

idx<-> j  
idx=j



1	2	8	5	4	3	6	7
---	---	---	---	---	---	---	---

처음 비교하는 값  $ar[idx]$ 는 2였으나,  $8 \leftrightarrow 2$  swap으로  $ar[idx]$ 위치 값이 8로 변경됨  
따라서 swap 후, idx 값을 j로 바꿔줘야 계속 2로 비교할 수 있다!

## 2. 삽입 정렬

비교idx: -->

현재idx: -->

1	2	8	5	4	3	6	7
---	---	---	---	---	---	---	---



4)  $\text{idx값} < \text{비교idx의 값}$  :

$\text{idx} \leftrightarrow j$

$\text{idx} = j$

1	2	5	8	4	3	6	7
---	---	---	---	---	---	---	---



5)  $\text{idx} > \text{비교idx의 값}$  : 비교idx--

1	2	5	8	4	3	6	7
---	---	---	---	---	---	---	---





## 2. 삽입 정렬

1	2	5	8	4	3	6	7
---	---	---	---	---	---	---	---



4)  $\text{idx값} < \text{비교idx의 값}$  :

$\text{idx} \leftrightarrow j$   
 $\text{idx} = j$

1	2	5	4	8	3	6	7
---	---	---	---	---	---	---	---



4)  $\text{idx값} < \text{비교idx의 값}$  :

$\text{idx} \leftrightarrow j$   
 $\text{idx} = j$

1	2	4	5	8	3	6	7
---	---	---	---	---	---	---	---



비교idx: -->

현재idx: -->

## 2. 삽입 정렬

### <실습> InsertionSort.java

```
public class InsertionSort {
    public static void main(String[] args) {
        int ar[] = {8,1,2,5,4,3,6,7};
        int tmp;
        int idx;
        for(int i=1; i<ar.length; i++) {
            idx=i;
            for(int j=i-1; j>=0; j--) {
                if(ar[idx]<ar[j]) {
                    tmp=ar[idx];
                    ar[idx]=ar[j];
                    ar[j]=tmp;
                    idx=j;
                }
                else {
                    break;
                }
            }
            for(int i=0; i<ar.length; i++) {
                System.out.print(ar[i]+" ");
            }
        }
    }
}
```

i: 위치를 찾을 idx  
→ 범위: 1~ 배열의 끝

j: 비교할 idx  
→ 범위: i-1~ 0까지

```
if(ar[idx]<ar[j]) {
    tmp=ar[idx];
    ar[idx]=ar[j];
    ar[j]=tmp;
    idx=j;
}
else {
    break;
}
```

만약, 현재 값 < 비교 idx 라면,  
값을 바꾸고, idx를 j로 변경

### 3. 버블 정렬

: 인접한 두 수를 비교해서 큰 수를 뒤로 보내는 알고리즘  
정렬과정이 거품이 일어나는 것과 비슷하다 하여 버블!

#### 장점

구현이 쉽고 코드가 직관적이다.

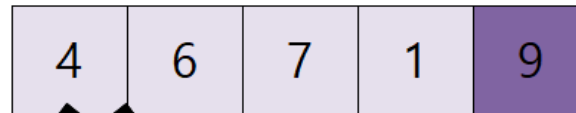
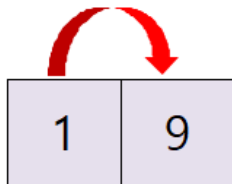
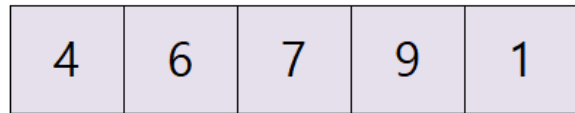
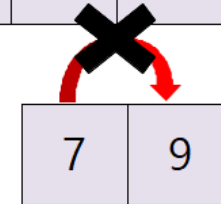
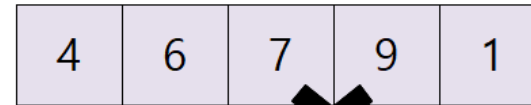
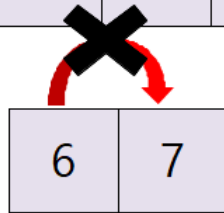
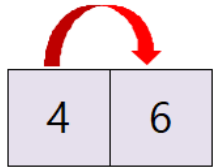
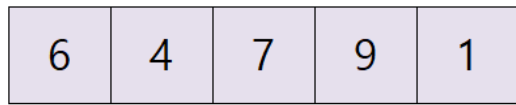
#### 단점

시간이 오래걸린다.

최선, 최악, 평균 모두  $O(n^2)$ 이라는 시간복잡도를 가진다.

### 3. 버블 정렬

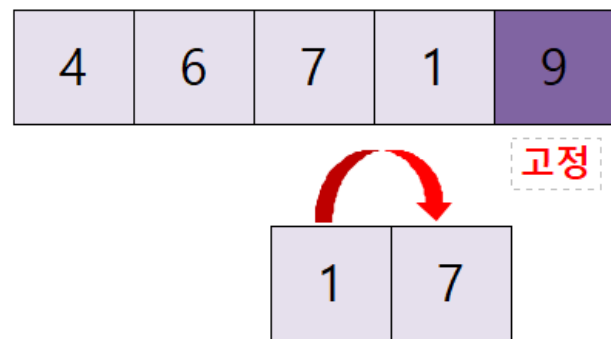
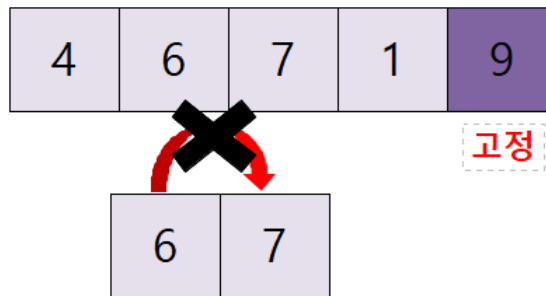
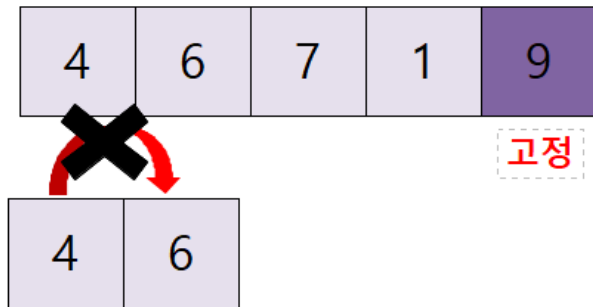
#### 기본 로직



고정

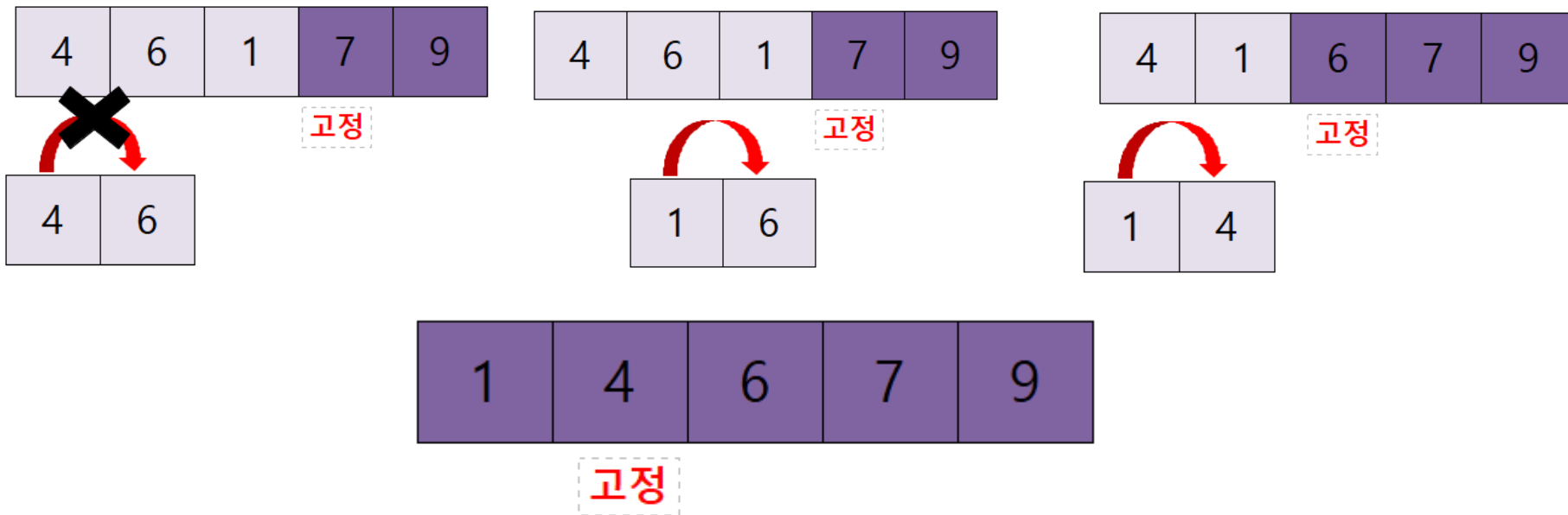
### 3. 버블 정렬

#### 기본 로직



### 3. 버블 정렬

#### 기본 로직



정렬 완료!!

### 3. 버블 정렬

#### <실습> BubbleSort.java

```
public class BubbleSort {  
    public static void main(String[] args) {  
        int ar[] = {6,4,7,9,1};  
        int tmp;  
        for(int i=ar.length-1; i>=0; i--) {  
            for(int j=0; j<i; j++) {  
                if(ar[j]>ar[j+1]) {  
                    tmp=ar[j];  
                    ar[j]=ar[j+1];  
                    ar[j+1]=tmp;  
                }  
            }  
        }  
        for(int i=0; i<ar.length; i++) {  
            System.out.print(ar[i]+" ");  
        }  
        System.out.println();  
    }  
}
```

i: 제일 큰 값이 올 위치

→ 범위: 배열끝~0

j: 비교할 idx

→ 범위: 0~ i까지