

Quick Sort,

🕒 작성일시	@2022년 8월 2일 오전 9:43
▼ 강의 번호	
📅 Date	@2022년 8월 2일
📎 자료	
☑ 복습	<input type="checkbox"/>

Quick Sort

데이터를 대소그룹 둘로 나누어 분해한 후에 전체를 최종적으로 정렬하는 방식의 알고리즘이다.

Divide and conquer (분할 정복법)

퀵정렬은 대량의 데이터를 정렬할때 매우 자주 사용된다. 유명한 알고리즘 중에서도 실제로 많이 사용되는 빈도가 가장 높고 중요한 알고리즘이기도 한다.

퀵정렬은 '기준값을 선택한 후 그 보다 작은 데이터 그룹과 큰 데이터 그룹으로 나눈다.' 라는 처리를 반복 수행하여 데이터를 정렬하게 된다.

Quick Sort

5	4	7	6	8	3	1	2	9
0	1	2	3	4	5	6	7	8

5	4	7	6	8	3	1	2	9
0	1	2	3	4	5	6	7	8

맨 앞의 공을 기준
으로 한다.

3	4	2	1	5	8	6	7	9
0	1	2	3	4	5	6	7	8

기준보다 큰 공들은
기준 뒤로 작은공들
은 기준 앞으로 이동

3	4	2	1
0	1	2	3

8	6	7	9
5	6	7	8

맨 앞의 공을 기준
으로 한다.

2	1	3	4
0	1	2	3

7	6	8	9
5	6	7	8

기준보다 큰 공들은
기준 뒤로 작은공들
은 기준 앞으로 이동

2	1
0	1

7	6
5	6

맨 앞의 공을 기준
으로 한다.

1	2
0	1

6	7
5	6

기준보다 큰 공들은
기준 뒤로 작은공들
은 기준 앞으로 이동

1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8

퀵 정렬의 알고리즘

퀵 정렬은 크게 2개의 처리로 구성된다.

1. 기준값을 경계로 데이터를 대소로 나누는 처리
2. 나눈 데이터에 대해 반복적으로 똑같은 작업을 실행

1. 기준값을 경계로 데이터를 대소로 나누는 처리

- 퀵정렬의 핵심은 데이터를 대소로 나누는 처리이다.
- 배열의 왼쪽과 오른쪽부터 각각 변수를 움직여 대소로 정렬하자.

기준값보다 작은 공을 기준값의 앞으로 이동시키고 기준값보다 큰공은 뒤로 이동시키는 것이 바로 퀵 정렬의 초석이되는 처리이다.

배열 설정 : 먼저 배열을 준비하자 정수형 배열로 이름은 arr로 요소수는 9개로 정한다. 따라서 첨자는 0 부터 8까지 사용된다.

arr

5	4	7	6	8	3	1	2	9
0	1	2	3	4	5	6	7	8

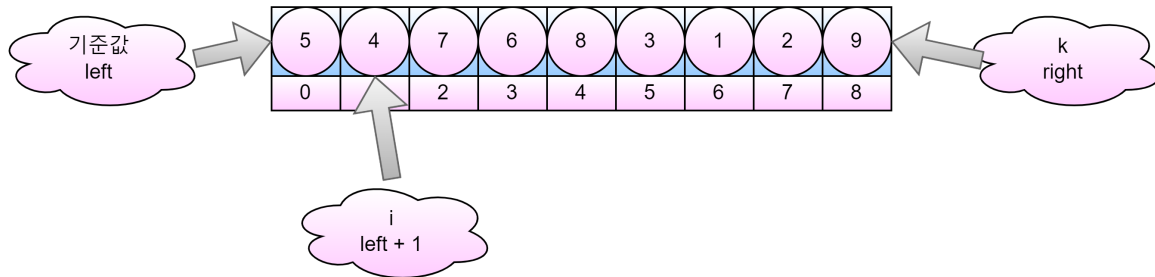
변수 설정

변수는 5개를 준비한다.

- left - 정렬 범위에서 맨 앞 요소에 첨자를 넣는 변수
- right - 정렬 범위에서 맨 끝 요소에 첨자를 넣는 변수
- i - 기준값보다. 큰 요소를 찾기 위한 변수
- k - 기준값보다 작은 요소를 찾기 위한 변수


- w - 데이터 교환용 임시 변수 temp

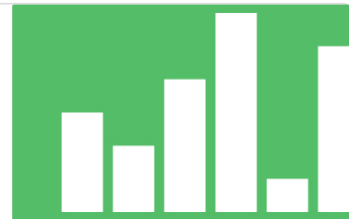
이 다섯개의 변수를 사용하여 우선 left와 right에 각각 정렬 범위 맨 앞 요소의 첨자와 마지막 요소의 첨자를 대입한다. 따라서 이번에는 (처음에는) left 0 right 8 이 된다. 기준은 맨 앞 요소로 하기 때문에 arr[left]이 된다. 그리고 i에 left 의 하나 오른쪽 left + 1 로 정하고 k에는 right를 대입한다.



Sorting (Bubble, Selection, Insertion, Merge, Quick, Counting, Radix) - VisuAlgo

Sorting is a very classic problem of reordering items (that can be compared, e.g., integers, floating-point numbers, strings, etc) of an array (or a list) in a certain order (increasing, non-decreasing (increasing or flat), decreasing, non-increasing (decreasing or flat)),

 <https://visualgo.net/en/sorting>



- 변수 i를 사용하여 기준값보다 큰 요소 찾기

i는 '기준값보다 큰 요소를 찾는 변수'이다. 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 있는 지 확인하고 발견되면 그곳에서 멈춘다.

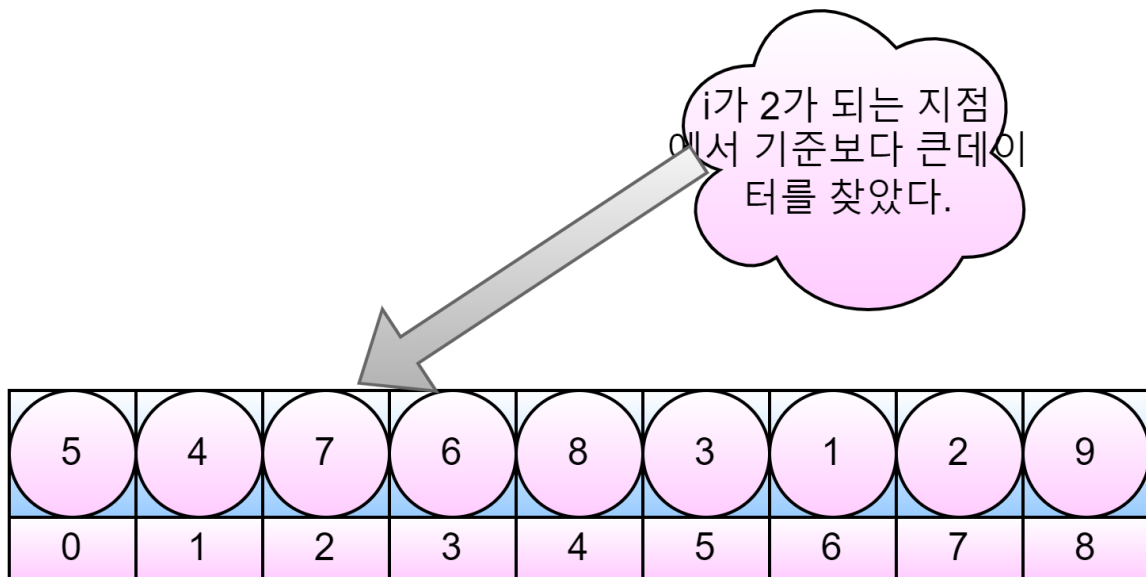
$arr[i] > arr[left]$

$arr[i] < arr[left]$
and $i < right$

$i = i + 1$



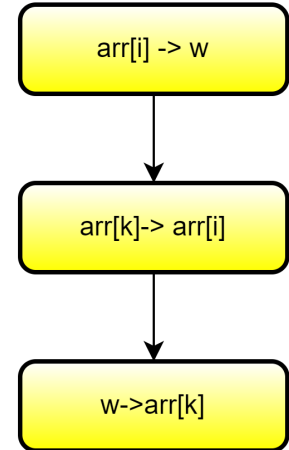
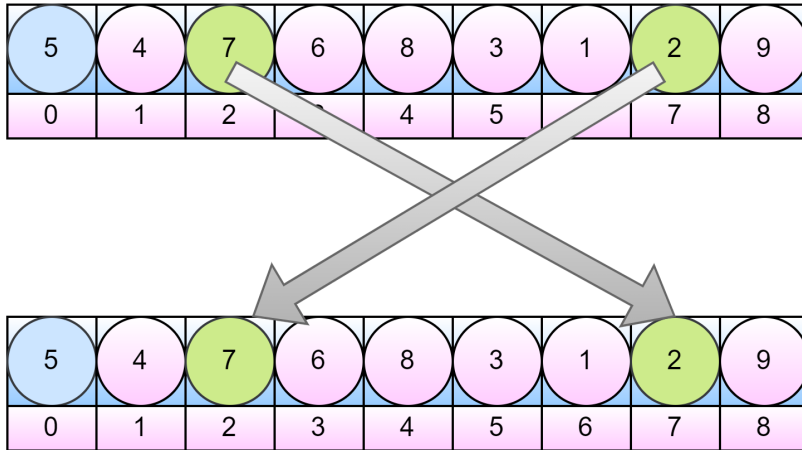
2가지 조건 , 기준값 $arr[left]$ 보다 큰값을 찾고 오른쪽 끝까지 찾지 못할때까지 반복

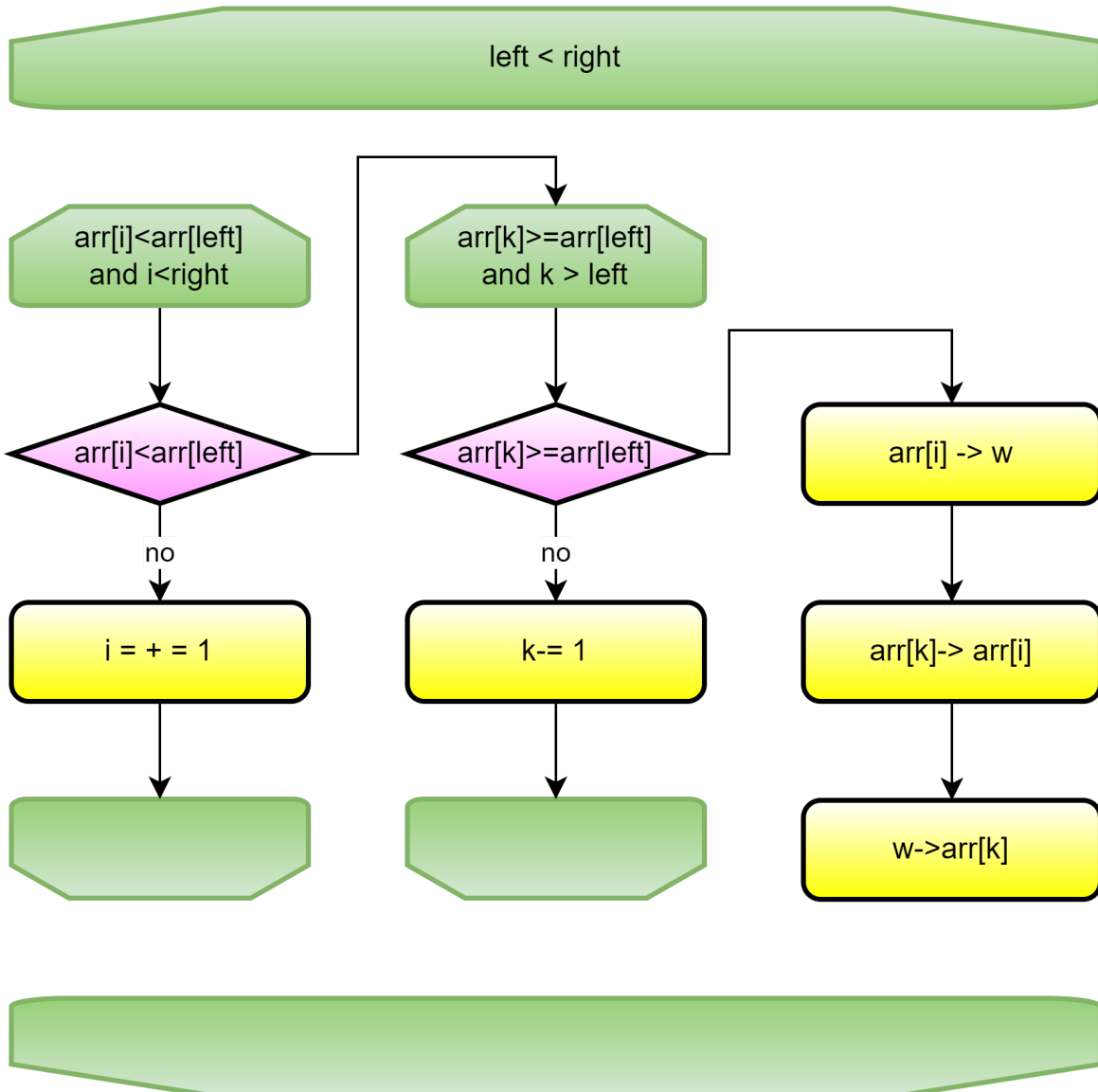


기준값보다 큰 요소를 발견했기 때문에 i 는 일단 여기에서 멈춘다. 그리고 반대쪽 변수 k 즉 작은 값 찾기로 넘어간다.

- 변수 k 를 사용하여 기준값보다 작은 요소 찾기

k 는 '기준값보다 작은 요소를 찾는 변수'이다. 현재 위치에서 하나씩 왼쪽으로 이동하면서 기준값보다 작은 요소가 있는 지 확인하고 발견되면 그곳에서 멈춘다.





```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    int[] arr = {5,4,7,6,8,3,1,2,9};
    arr = quickSort(arr,0,arr.length-1); //arr, 시작, 끝
    System.out.println(Arrays.toString(arr));
}

static int[] quickSort(int[] arr, int start, int end) {
    int p = partition(arr,start,end);

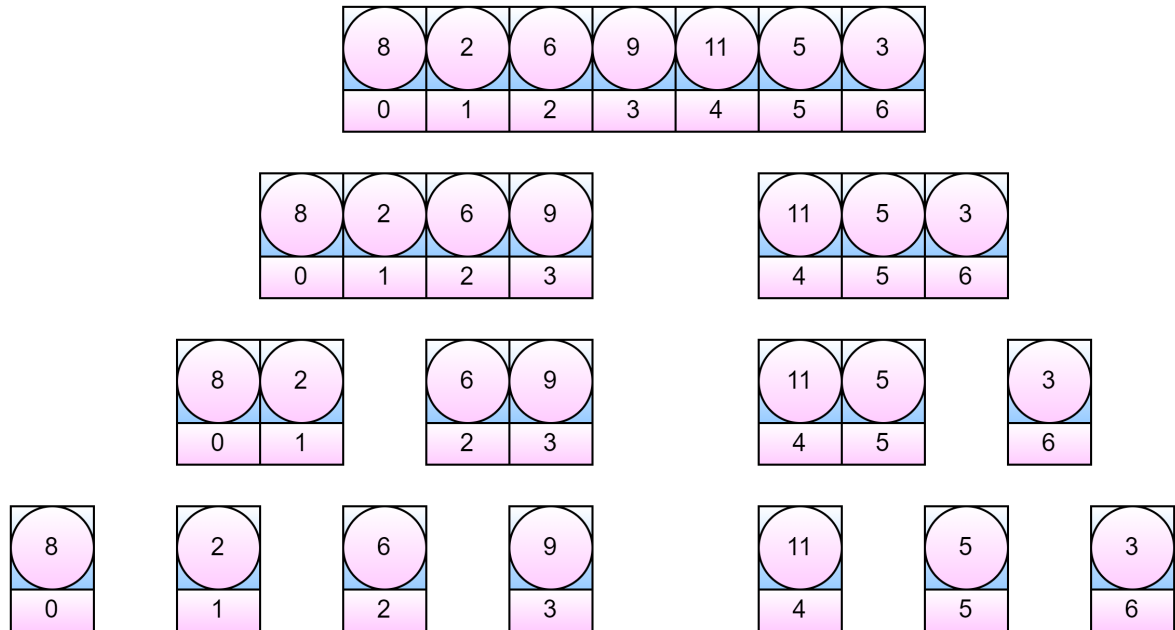
    if(start < p-1)
        quickSort(arr, start, p-1);
    if(p<end)
        quickSort(arr,p,end);
    return arr;
}

static int partition(int[] arr,int start,int end) {
    int pivot = arr[(start+end)/2]; // 중심
    while(start <= end) {
        while(arr[start]<pivot) start++;
        while(arr[end]>pivot) end--;
        if(start <= end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }
    return start;
}
}

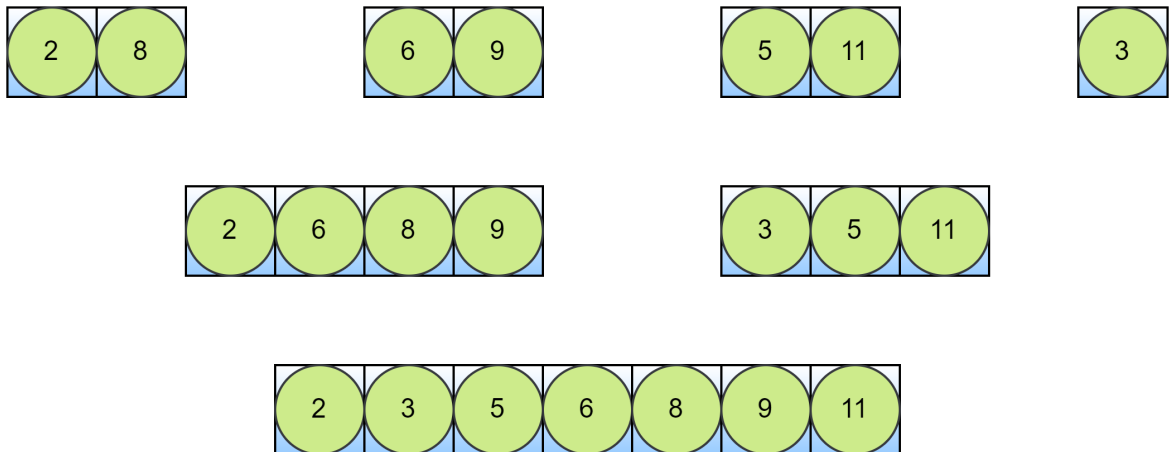
```

Merge Sort 병합 정렬

분할 정복(Divide and Conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할 한 후에 분할 된 문제를 해결하고 그 결과를 다시 결합하는 알고리즘이다.



먼저 데이터들을 정렬을 생각하지 않고 1개씩 될때까지 나눈다. 나누는 방법은 배열을 반으로 쪼개고 그 쪼개진 배열을 또 다시 반으로 쪼개고 이 과정을 반복한다. 위에서 7개의 데이터를 분할하기위해 이과정을 3번 거쳤다.



분할이 완료된 후에는 데이터를 비교하면서 결합한다. 제일 앞에 있는 2와 8을 다시 합치는데 작은 수 2가 앞으로 큰수 8이 뒤로 보낸상태로 결합한다. 이과정을 각각 2개씩 반복하여 합치고 그 합친 2개를 다시 합칩니다. 이때 각각의 그룹에서 먼저 제일 첫번째 값을 비교하여 작은 값을 추출한다. 그 다음 다시 한번 각각의 그룹의 제일 왼쪽 즉 작은 값을 또 다시 비교하여 둘 중 작은 값을 다시 추출한다. 이 과정을 반복하여 전체 정렬을 마치게 된다.

DB Chapter.4(필터링)

- 새로운 데이터 웨어하우스 피드를 준비할때 사용한 테이블에서 모든 데이터 제거
- 새열이 추가된 후 테이블의 모든 행 수정
- 메시제 큐 테이블에서 모든행 검색

조건 평가

```
WHERE title = 'Teller' AND start_date < '2007-01-01'
```

두 가지 조건을 모두 and 충족하는 타이틀이 텔러이고 동시에 날짜가 2007년 1월 1일 이전인 행만 결과 포함된다. 조건이 여러개 있어도 AND연산자로 구분될 경우에는 결과셋에 모든 조건이 True 인 경우만 포함된다. 즉 true 로 평가된다.

괄호사용

WHERE true AND (true OR true)	true
WHERE true AND (true OR false)	true
WHERE true AND (false OR true)	true
WHERE true AND (false OR false)	false

세가지 조건이 있을 경우 최종 결과는 괄호 안의 2가지의 조건의 결과와 최종 마지막 결과의 평가에 따라 최종 결과가 정해지게 된다.

not 연산자 사용

WHERE true AND NOT (true OR true)	false
WHERE true AND NOT (true OR false)	false
WHERE true AND NOT (false OR true)	false
WHERE true AND NOT (false OR false)	true
WHERE false AND NOT (true OR true)	false
WHERE false AND NOT (true OR false)	false
WHERE false AND NOT (false OR true)	false
WHERE false AND NOT (false OR false)	false

not 연산자를 사용하여 평가 결과를 반대로 true 의 경우는 false 로 false의 경우에는 true 결과를 뒤집을 수 있다.

조건 작성

표현식은 아래 의 내용들로 구성할 수 있다.

- 숫자
- 테이블 또는 뷰의 컬럼
- 텔러와 같은 문자열
- concat 과 같은 내장 함수들
- 서브쿼리, 헤드텔러 등등과 같은 표현식 목록

조건의 유형

동등조건 '열 = 값'

부등조건 두 표현식이 동일 하지 않을때 사용 <>

동등조건을 사용한 데이터 수정

```
DELETE FROM account
WHERE status = 'CLOSED' AND YEAR(close_date) = 2002;
```

범위 조건

```
SELECT emp_id fname, lname, start_date
FROM employee
WHERE start_date < '2007-01-01';
```

특정 범위내에 원하는 조건이 있는지를 확인하는 범위 조건을 작성할 수 있다.

이 유형은 보통 숫자 또는 시간데이터로 작업할때 주로 발생한다.

between 연산자

- 범위의 상한과 하한이 모두 있을때 사용한 between 연산자를 사용하여 하나의 조건으로 사용할 수 있다.

```

SELECT emp_id fname, lname, start_date
FROM employee
WHERE start_date BETWEEN '2005-01-01' AND '2007-01-01';
-----
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE avail_balance BETWEEN 300 AND 5000;

```

와일드카드 사용하기

- 특정 문자로 시작/종료하는 문자열
- 부분 문자열로 시작/ 종료하는 문자열
- 문자열 내에 특정 문자를 포함하는 모든 문자열
- 문자열 내에 특정 문자열을 포함하는 모든 문자열
- 개별 문자에 관계 없이 특정한 형식의 문자열

Table 4-4. Wildcard characters

Wildcard character	Matches
_	Exactly one character
%	Any number of characters (including 0)

_ 정확히 한글자

% 0을 포함한 개수에 상관 없이 모든 문자

```

mysql> SELECT lname
-> FROM employee
-> WHERE lname LIKE '_a%e%';

```

```

+-----+
| lname  |
+-----+
| Barker |
| Hawthorne |
| Parker |
| Jameson |
+-----+

```

```

SELECT lname
FROM employee
WHERE lname LIKE '_a%e%';

```

```
mysql> SELECT cust_id, fed_id
-> FROM customer
-> WHERE fed_id LIKE '___-__-____';
```

cust_id	fed_id
1	111-11-1111
2	222-22-2222
3	333-33-3333
4	444-44-4444
5	555-55-5555

```
SELECT cust_id , fed_id
FROM customer
WHERE fed_id LIKE '___-__-____';
```

```
mysql> SELECT emp_id, fname, lname
-> FROM employee
-> WHERE lname LIKE 'F%' OR lname LIKE 'G%';
```

emp_id	fname	lname
5	John	Gooding
6	Helen	Fleming
9	Jane	Grossman
17	Beth	Fowler

```
SELECT emp_id, fname, lname
FROM employee
WHERE lname LIKE 'F%' OR lname LIKE 'G%';
```

When working with null, you should remember: Null일수는 있지만 Null과 같을수는 없다.

- An expression can *be* null, but it can never *equal* null.
- Two nulls are never equal to each other. 두개의 null 서로 같지 않다.

```

mysql> SELECT emp_id, fname, lname, superior_emp_id
-> FROM employee
-> WHERE superior_emp_id IS NULL;
+-----+-----+-----+-----+
| emp_id | fname  | lname | superior_emp_id |
+-----+-----+-----+-----+
|      1 | Michael | Smith |                NULL |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT emp_id, fname, lname, superior_emp_id
-> FROM employee
-> WHERE superior_emp_id = NULL;
Empty set (0.01 sec)

mysql> SELECT emp_id, fname, lname, superior_emp_id
-> FROM employee
-> WHERE superior_emp_id IS NOT NULL;

```

Quiz 7

월급이 3000인 직원들의 이름, 월급, 직업을 출력해 보겠습니다.

ENAME	SAL	JOB
FORD	3000	ANALYST
SCOTT	3000	ANALYST

월급이 3000 이상인 직원들의 이름과 월급을 출력;

이름	월급
KING	5000
FORD	3000
SCOTT	3000

```

1 SELECT ENAME, SAL, JOB
2 FROM emp
3 WHERE SAL = 3000;

```

emp (2r × 3c)		
ENAME	SAL	JOB
FORD	3,000	ANALYST
SCOTT	3,000	ANALYST

```

1 SELECT ENAME AS '이름', SAL
2 FROM emp
3 WHERE SAL >= 3000;

```

emp (3r × 2c)	
이름	SAL
KING	5,000
FORD	3,000
SCOTT	3,000

Quiz 8

이름이 SCOTT인 사원의 이름, 월급, 직업, 입사일, 부서 번호를 출력해 보겠습니다.

ENAME	SAL	JOB	HIREDATE	DEPTNO
SCOTT	3000	ANALYST	82/12/22	20

```

1 SELECT ENAME, SAL, JOB, HIREDATE, DEPTNO
2 FROM emp
3 WHERE ENAME = 'SCOTT'

```

emp (1r × 5c)				
ENAME	SAL	JOB	HIREDATE	DEPTNO
SCOTT	3,000	ANALYST	1982-12-22	20

Quiz 9

연봉이 36000 이상인 직원들의 이름과 연봉을 출력해 보겠습니다.

ENAME	연봉
KING	60000
FORD	36000
SCOTT	36000

1	SELECT	ename,	sal*12	연봉
2	FROM	emp		
3	WHERE	sal*12	>=36000;	

emp (3r × 2c)	
ename	연봉
KING	60,000
FORD	36,000
SCOTT	36,000

Quiz 10

월급이 1000에서 3000 사이인 직원들의 이름과 월급을 출력해 보겠습니다.

ENAME	SAL
BLAKE	2850
CLARK	2450
:	:
ADAMS	1100
MILLER	1300

```

1 SELECT ENAME, Sal
2 FROM emp
3 WHERE sal BETWEEN 1000 AND 3000;

```

emp (11r x 2c)

ENAME	Sal
BLAKE	2,850
CLARK	2,450
JONES	2,975
MARTIN	1,250
ALLEN	1,600
TURNER	1,500
WARD	1,250
FORD	3,000
SCOTT	3,000
ADAMS	1,100
MILLER	1,300

Quiz 11

1982년도에 입사한 직원들의 이름과 입사일을 조회:

ENAME	HIREDATE
SCOTT	82/12/22
MILLER	82/01/11

Q11. + 1982년도에 입사한 직원들의 이름과 입사일을 조회:

```
SELECT ename, hiredate
FROM emp
WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31';
```

emp (2r x 2c)	
ename	hiredate
SCOTT	1982-12-22
MILLER	1982-01-11

Quiz 12

이름의 두 번째 철자가 M인 직원의 이름을 출력:

ENAME

SMITH

이름이 A가 포함된 직원들을 전부 검색:

```
1 SELECT ENAME
2 FROM emp
3 WHERE ENAME LIKE '_M%'
4
```

emp (1r x 1c)

ENAME
SMITH

```
1 SELECT ENAME
2 FROM emp
3 WHERE ENAME LIKE '%A%'
4
```

emp (7r x 1c)

ENAME
BLAKE
CLARK
MARTIN
ALLEN
JAMES
WARD
ADAMS

Quiz 13

커미션이 NULL인 사원들의 이름과 커미션을 출력해 보겠습니다.

ENAME	COMM
KING	
:	:
MILLER	

```
SELECT ENAME, COMM
FROM emp
WHERE COMM = NULL
```

Quiz 14

직업이 SALESMAN, ANALYST, MANAGER인 사원들의 이름, 월급, 직업을 출력해 보겠습니다.

ENAME	SAL	JOB
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
JONES	2975	MANAGER
MARTIN	1250	SALESMAN
ALLEN	1600	SALESMAN
TURNER	1500	SALESMAN
WARD	1250	SALESMAN
FORD	3000	ANALYST
SCOTT	3000	ANALYST

```
SELECT ename, sal, job
FROM emp
WHERE (job = 'SALESMAN' or job='ANALYST' or job='MANAGER');
```

```

1 SELECT ENAME, SAL, JOB
2 FROM emp
3 WHERE JOB IN('SALESMAN', 'ANALYST', 'MANAGER');
4 |

```

emp (9r × 3c)		
ENAME	SAL	JOB
BLAKE	2,850	MANAGER
CLARK	2,450	MANAGER
JONES	2,975	MANAGER
MARTIN	1,250	SALESMAN
ALLEN	1,600	SALESMAN
TURNER	1,500	SALESMAN
WARD	1,250	SALESMAN
FORD	3,000	ANALYST
SCOTT	3,000	ANALYST

AND 연산자 진리 연산표

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR 연산자 진리 연산표

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT 연산자 진리 연산표

NOT	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL