Radix Sort
기수 정렬

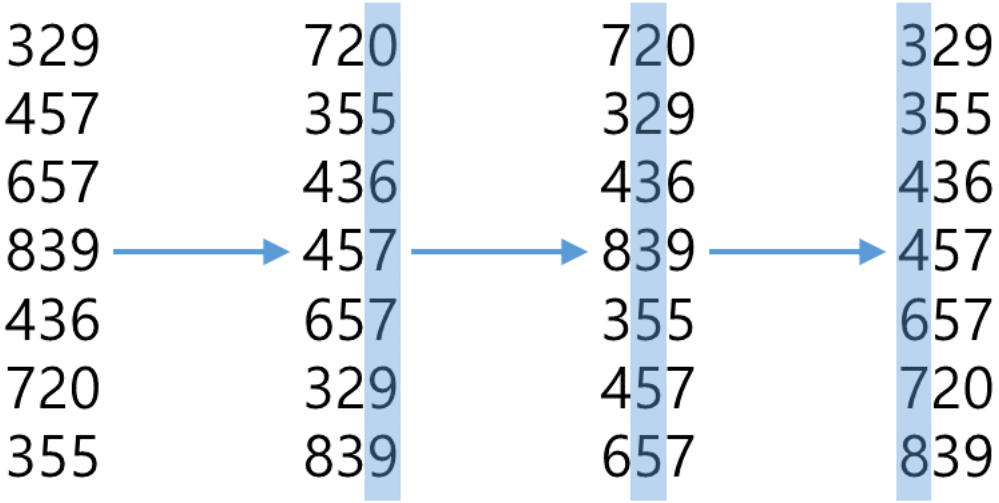
2023.01.12

허대현

Contents

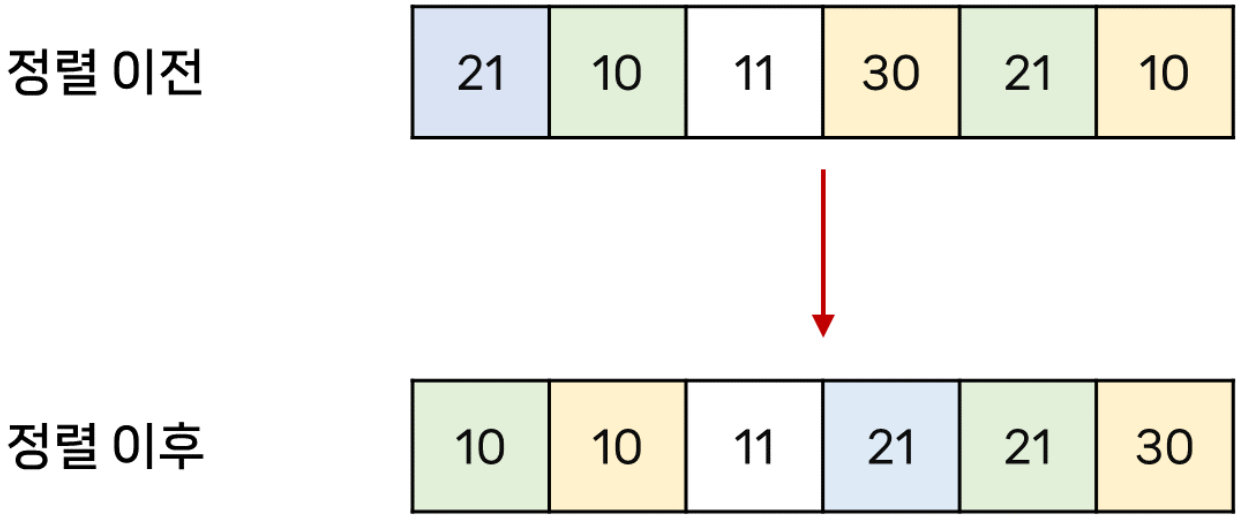
기수정렬

- 배열의 원소를 **기수(Radix)**에 맞는 버킷에 나누어 넣고 다시 리스트에 합쳐서 정렬하는 알고리즘
- 비교 연산을 수행하지 않는다
- 각 자리 정렬이 **안정적(Stable)** 이어야 된다
- **사전적으로** 정렬되는 정렬 순서를 사용
- 입력이 가지는 총 **자릿수(digits)**와 **기수의 정의역**에 대한 정보를 알아야 한다.



정렬에서의 안정성(Stablity)

출력 배열에서의 값이 같은 숫자가 입력 배열에 있던 것과 **같은 순서로** 나타나는 성질

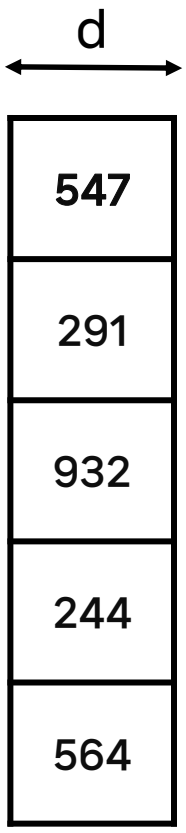


정렬되어 있는 데이터의 부속 데이터가 중요할 때 고려한다.

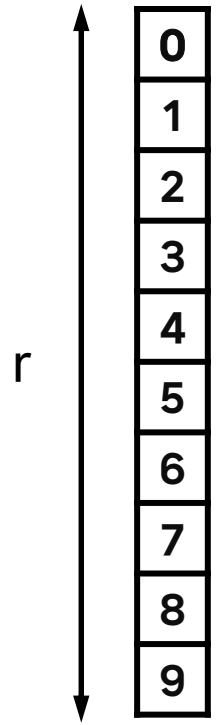
기수정렬에서는 **가장 낮은** 자릿수부터 먼저 정렬하여, 그 **순서를 유지**하여 정렬을 수행한다.

기수정렬 동작 예시

- n=5, d=3, r=10
- n: 원소의 총 개수, d:총 자릿수, r: 기수의 정의역(범위)



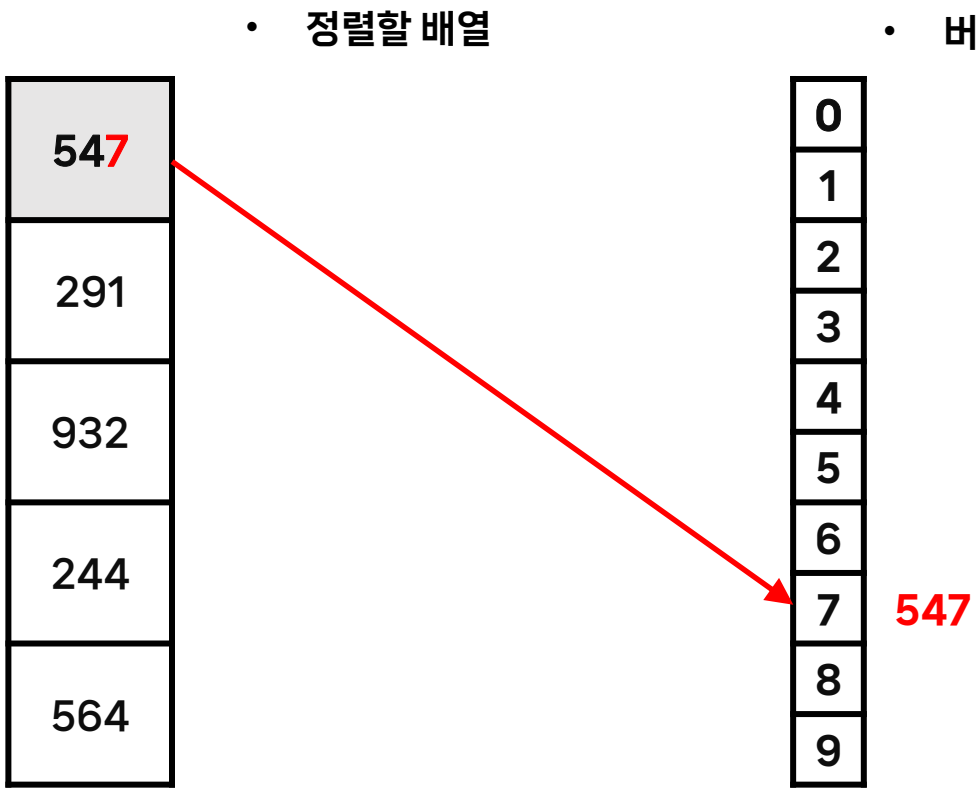
- **정렬할 배열**



- **버킷**
  - 각 자리에 올 수 있는 숫자를 index로 가지는 큐(Queue)들의 배열
  - 길이는 기수의 정의역의 크기와 같다

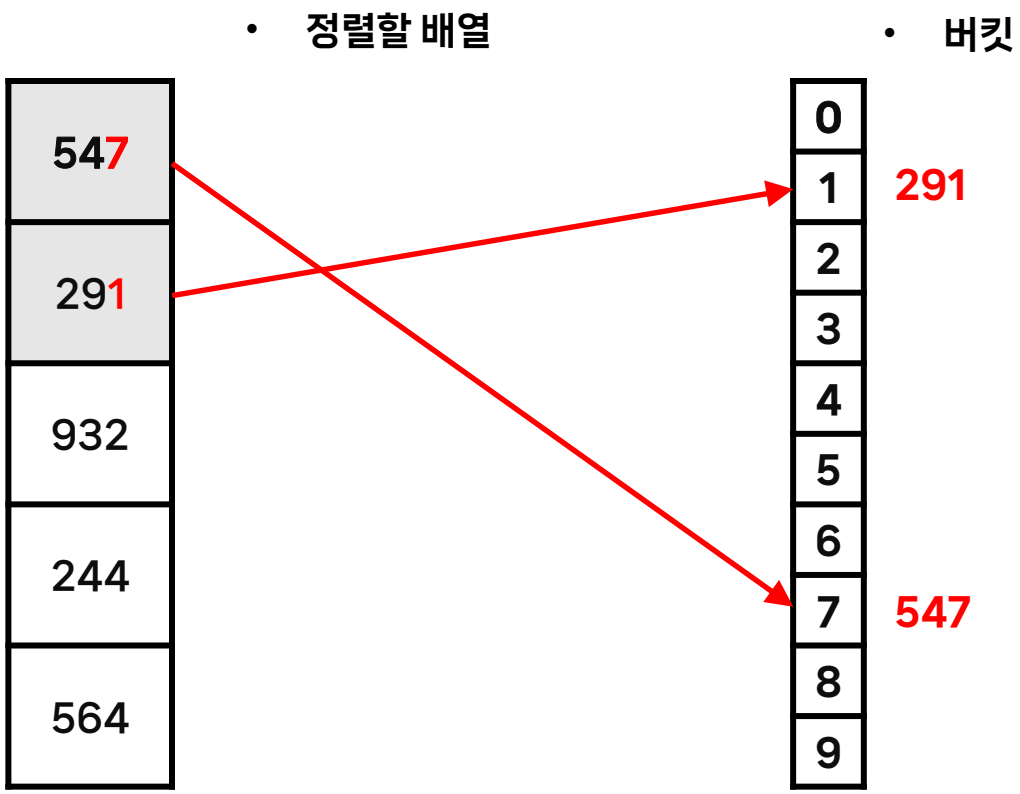
기수정렬 동작 예시

- 버킷에 집어 넣기



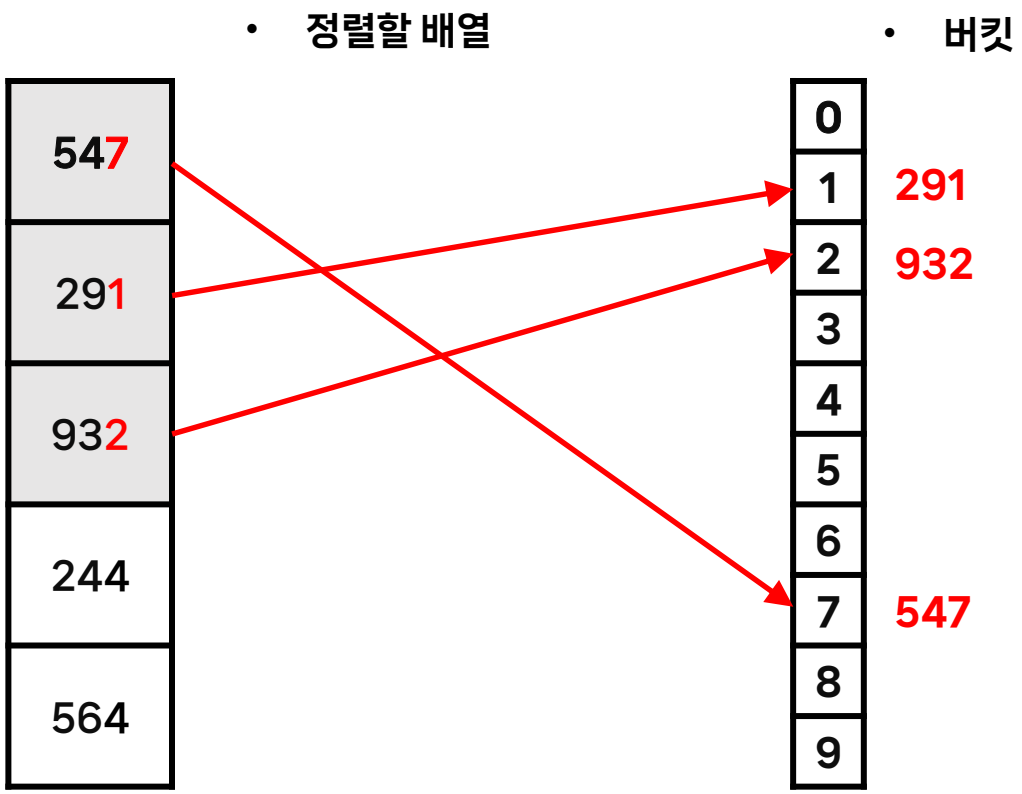
기수정렬 동작 예시

- 버킷에 집어 넣기



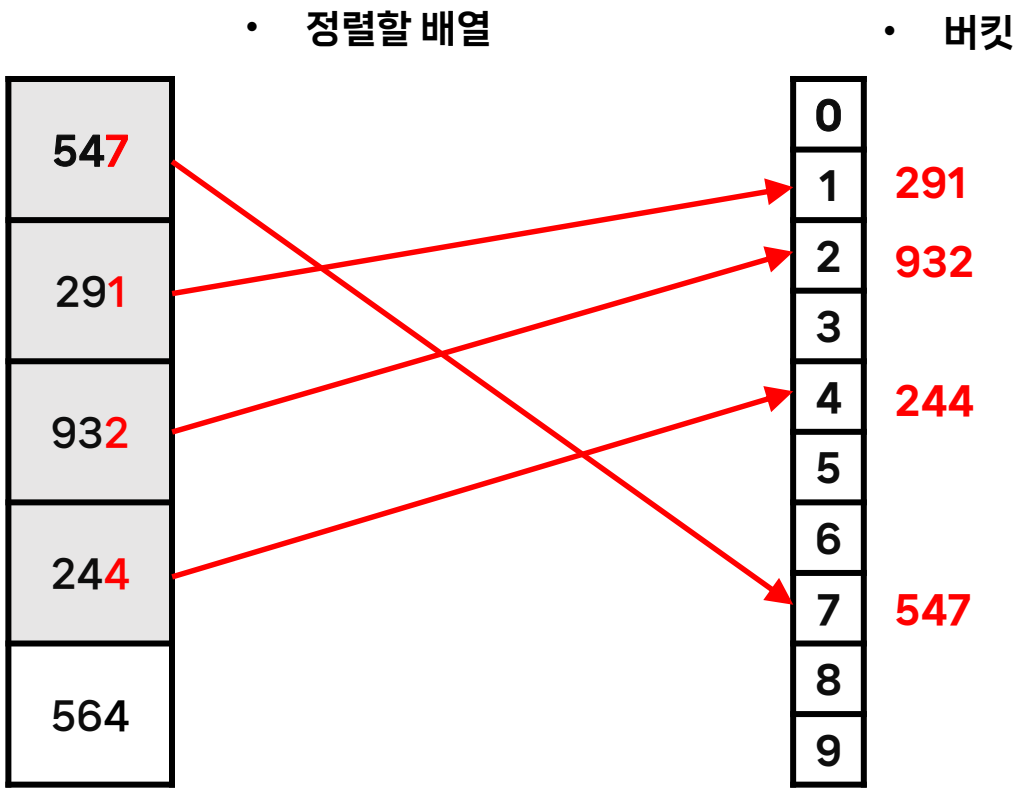
기수정렬 동작 예시

- 버킷에 집어 넣기



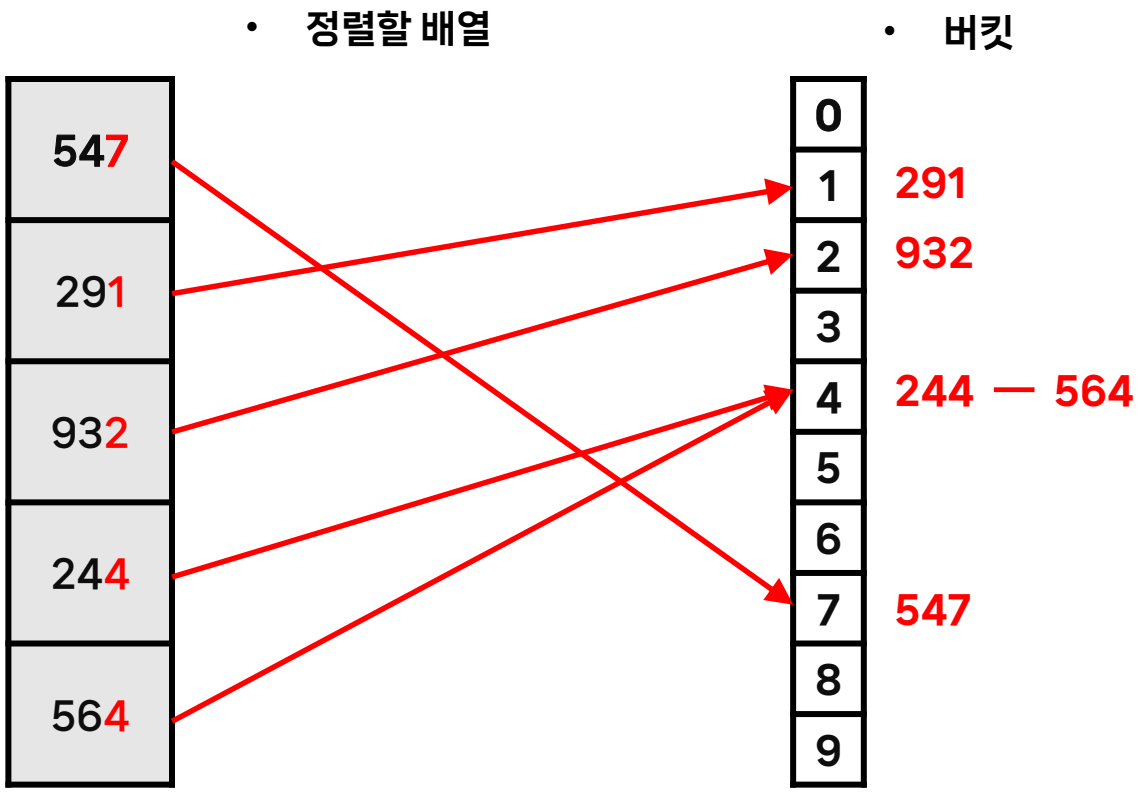
기수정렬 동작 예시

- 버킷에 집어 넣기



기수정렬 동작 예시

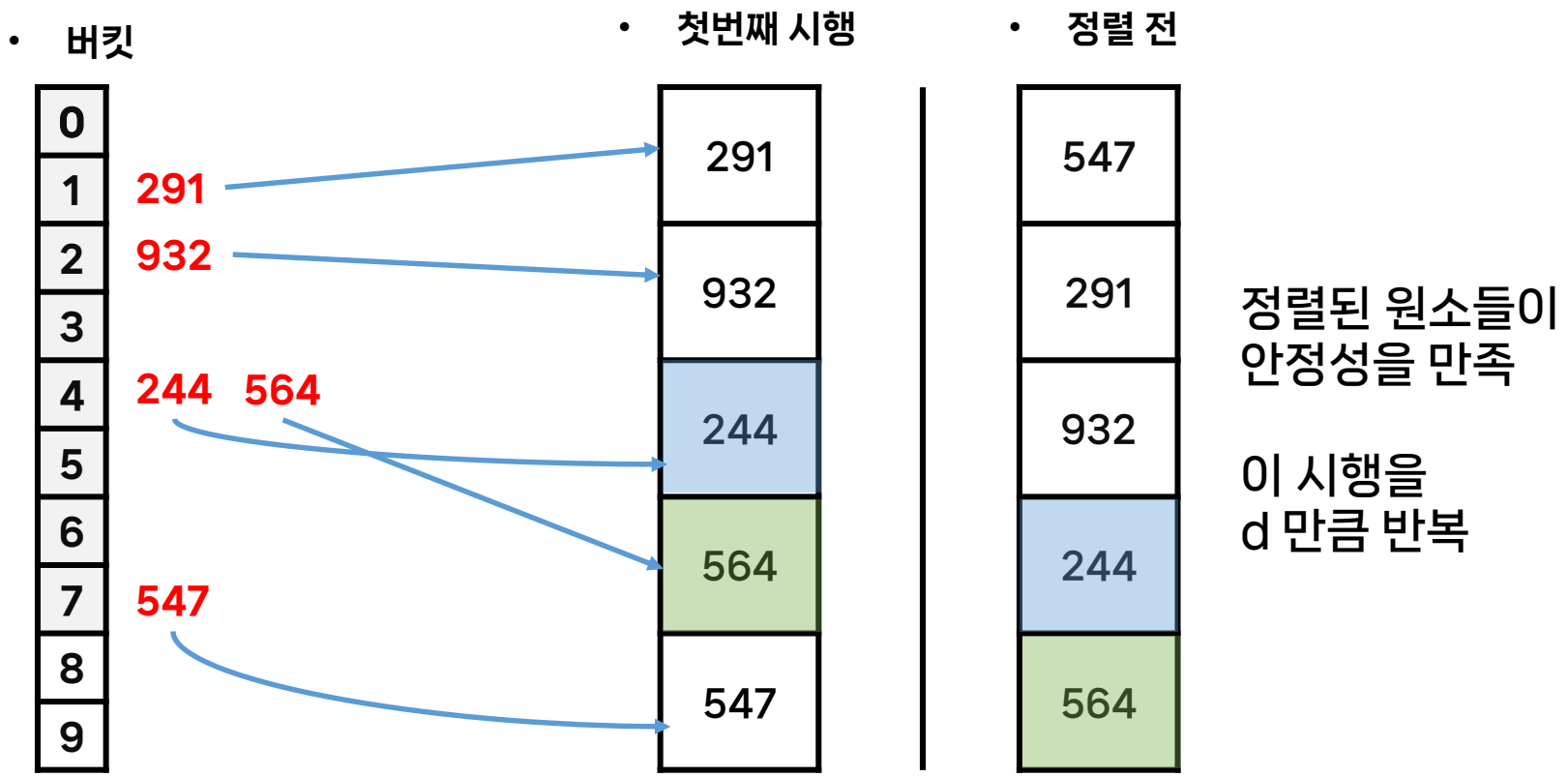
- 버킷에 집어 넣기



기수정렬 동작 예시

- 버킷에서 원래 리스트로 집어넣기

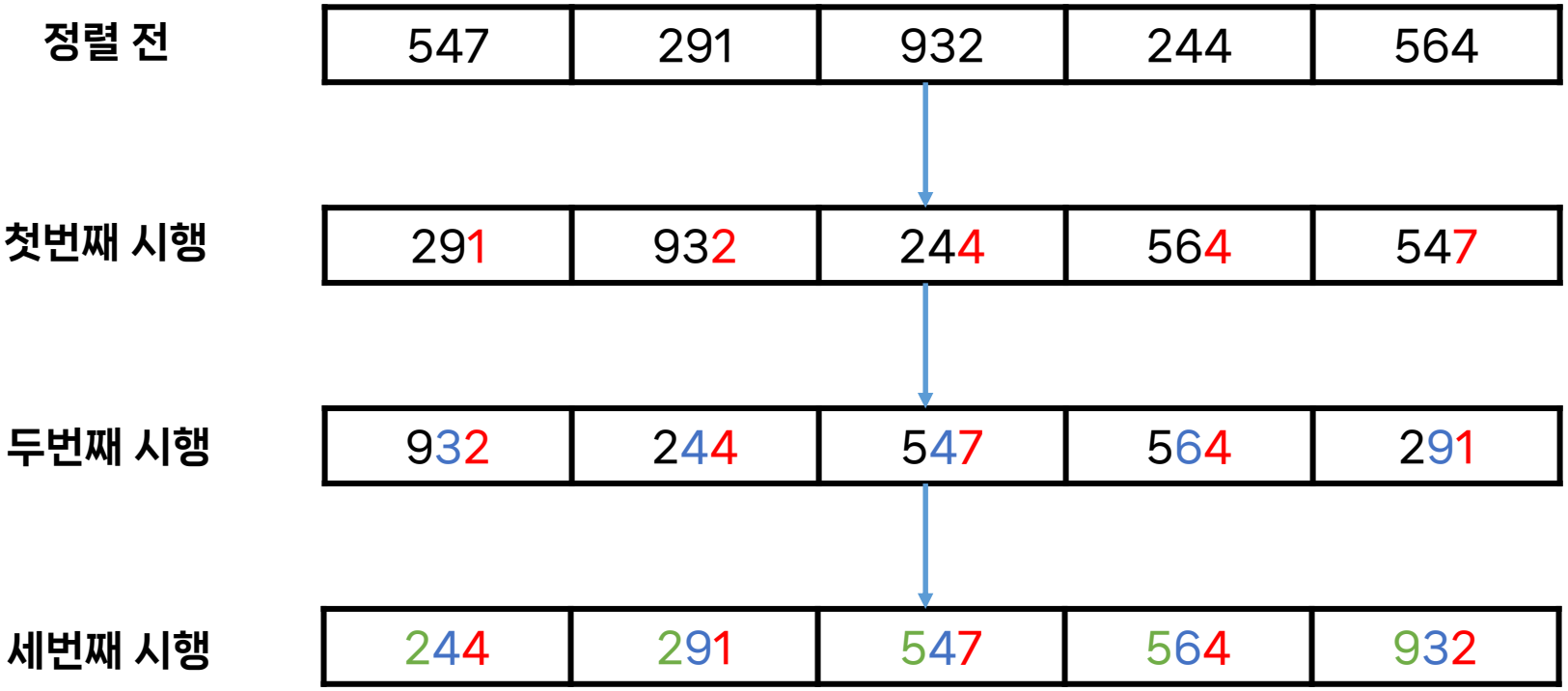
* 버킷의 해당 인덱스에서 원소를 선입선출식으로 출력



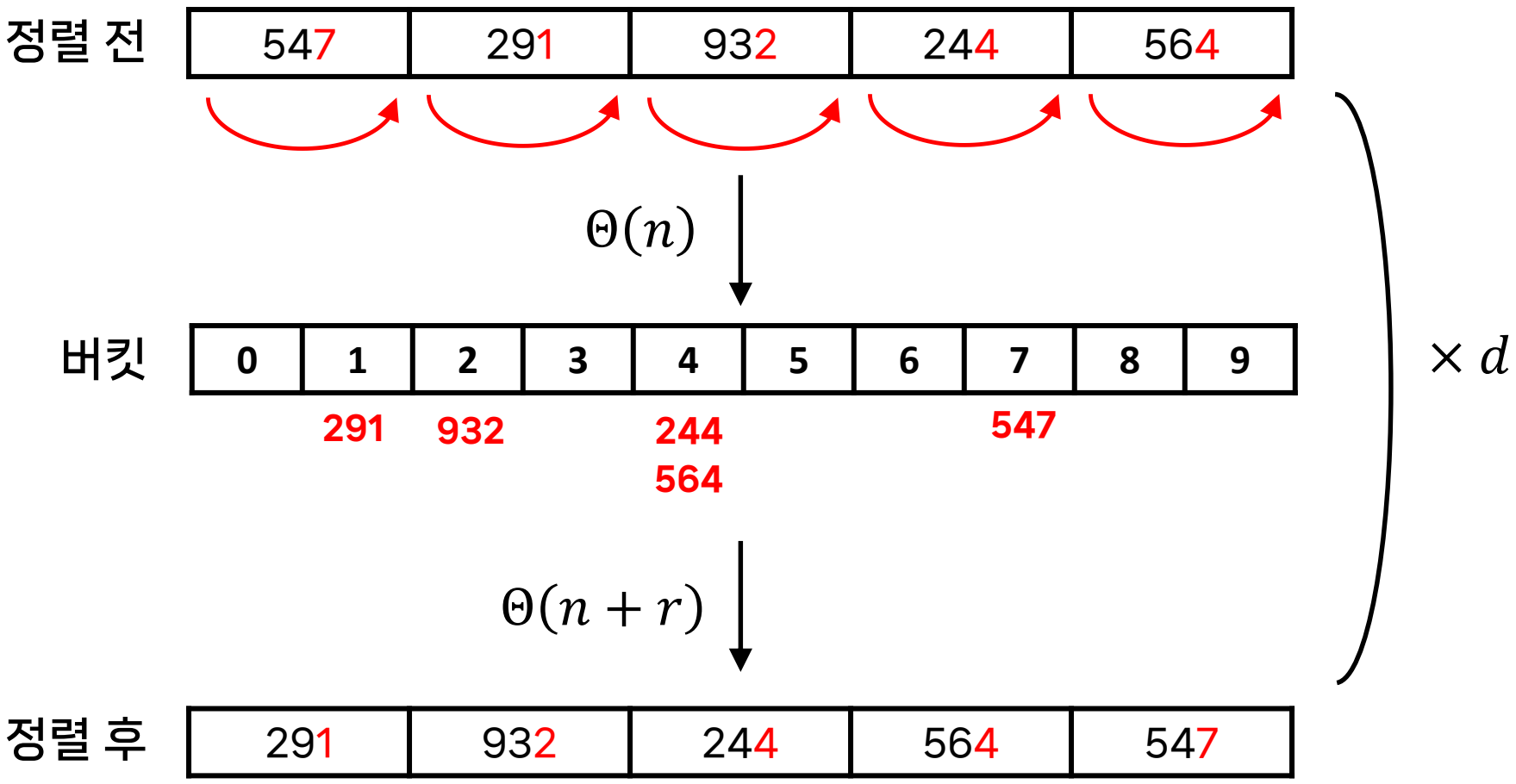
정렬된 원소들이
안정성을 만족

이 시행을
d 만큼 반복

기수정렬 동작 예시


정렬 전
547 291 932 244 564
첫번째 시행
291 932 244 564 547
두번째 시행
932 244 547 564 291
세번째 시행
244 291 547 564 932

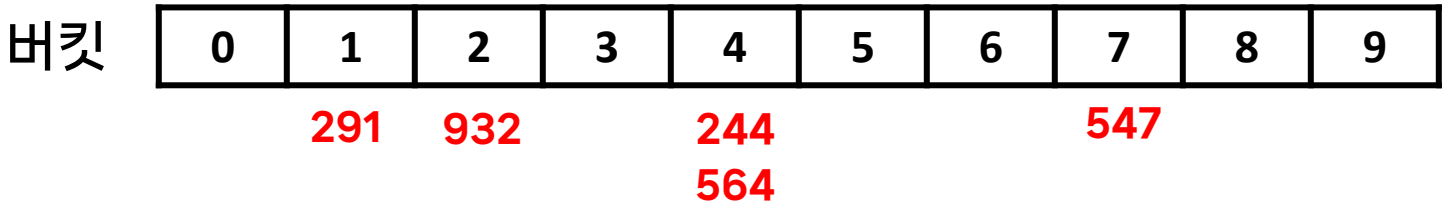
시간복잡도



$\therefore \Theta(d(n+r))$

d 가 상수, $r = O(n)$ 이면, $\Theta(n)$

공간복잡도



- 버킷은 원소를 저장할 수 있는 **큐(Queue)**의 배열
- r개의 빈 **큐(Queue)** 를 보유
- 버킷이 보유하는 모든 원소의 개수는 총 n 개

$\therefore \Theta(n+r)$

기수정렬의 장점과 단점

- 장점

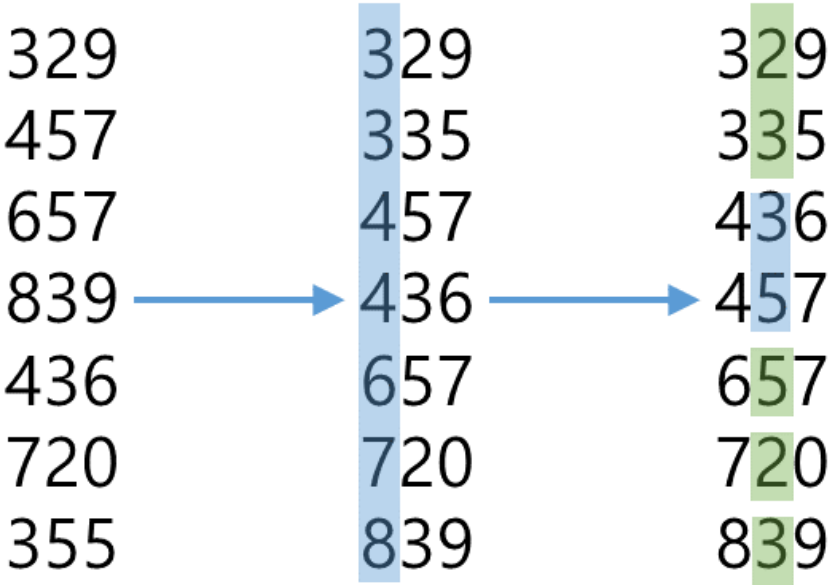
- 키가 필드 여러 개로 되어 있는 자료를 정렬할 때 유용하다.
  - 예) 날짜 (연/월/일 이라는 3개의 필드로 구성된 정보)
- 길이가 상수이고, 기수가 $O(n)$인 자료를 정렬할 때 $O(n)$ 안에 정렬
  - 예) 정수, 알파벳으로 구성된 문자열

- 단점

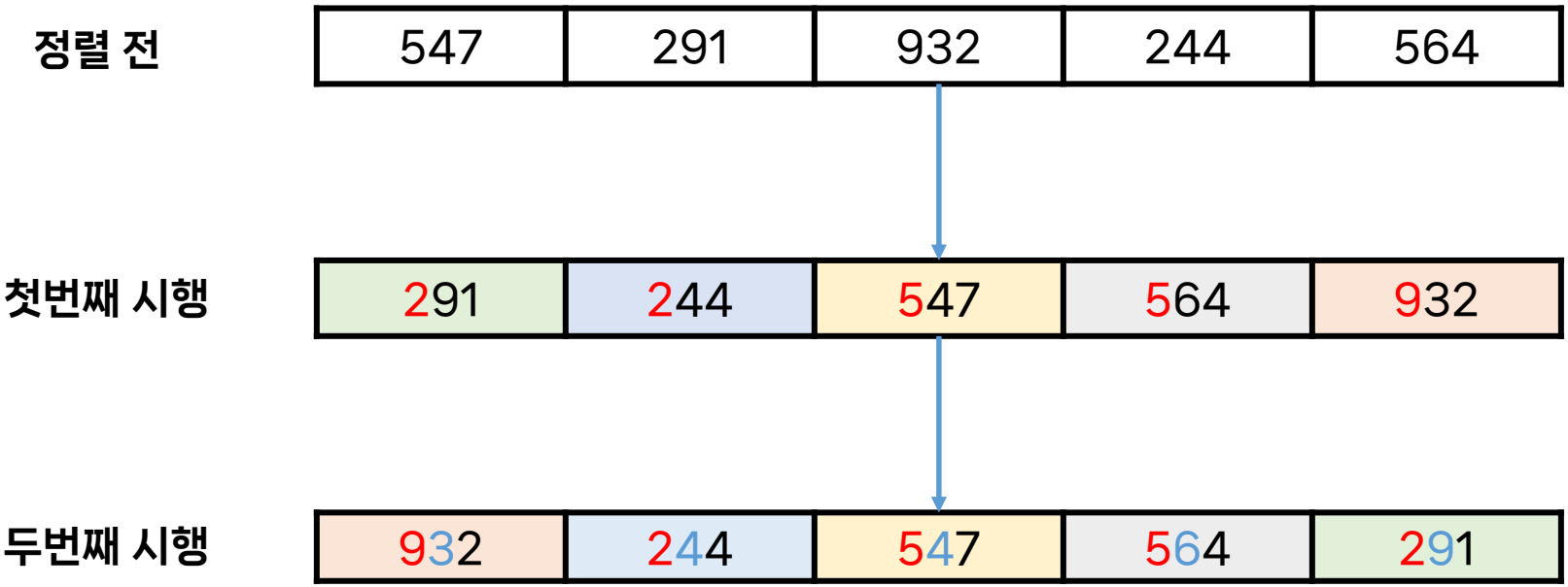
- 삽입 정렬 같은 알고리즘과 달리, 버킷을 만드는 데에 공간이 더 필요하다.

MSD 기수 정렬

- **최대 유효 숫자**를 기준으로 하여 **기수(Radix)**에 맞는 버킷에 나누어 넣고 다시 리스트에 합쳐서 정렬하는 알고리즘
- MSD(Most Significant Digit): 최대 유효 숫자.
- 각 자리 정렬이 **안정적(Stable) 일 필요는 없다.**

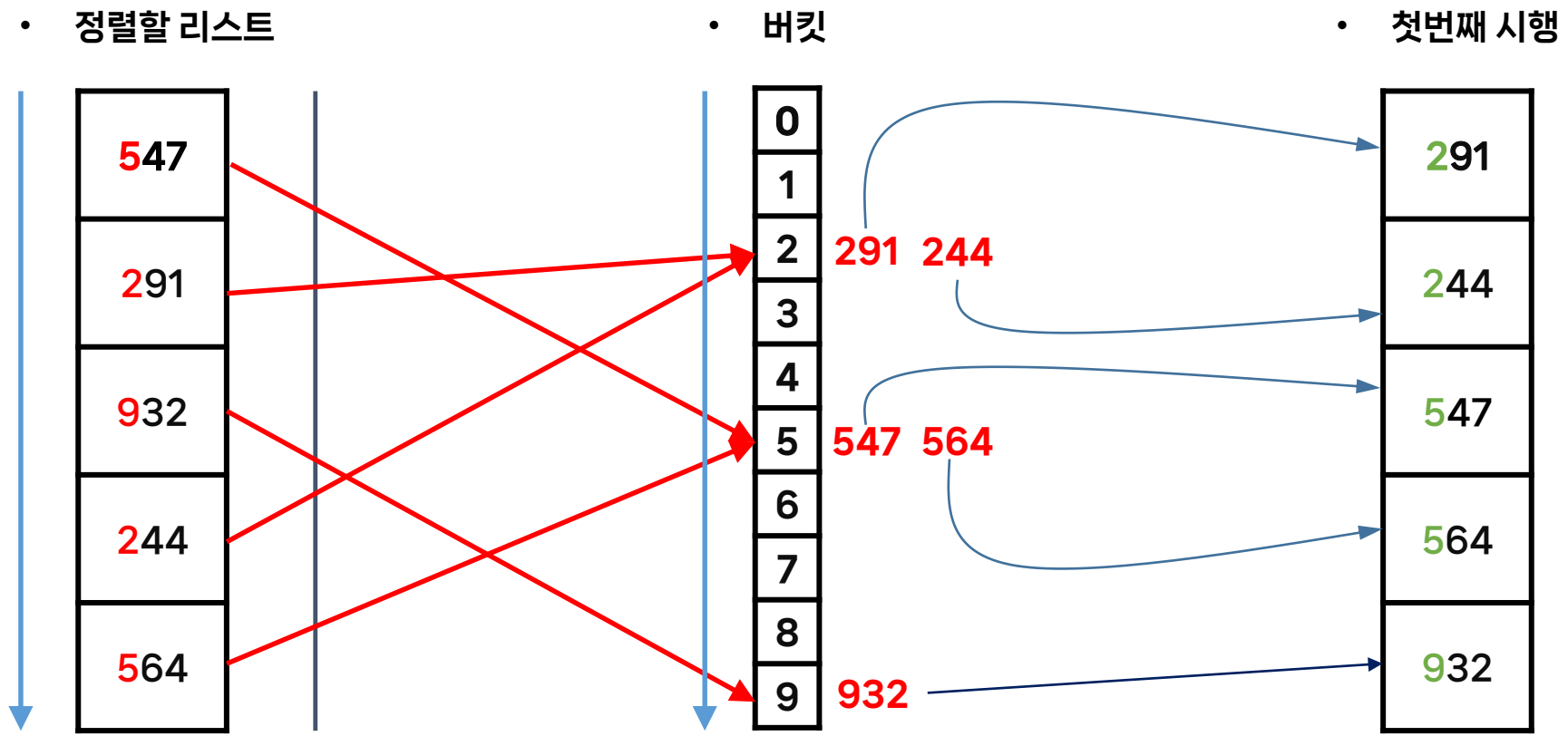


MSD 기수 정렬의 문제

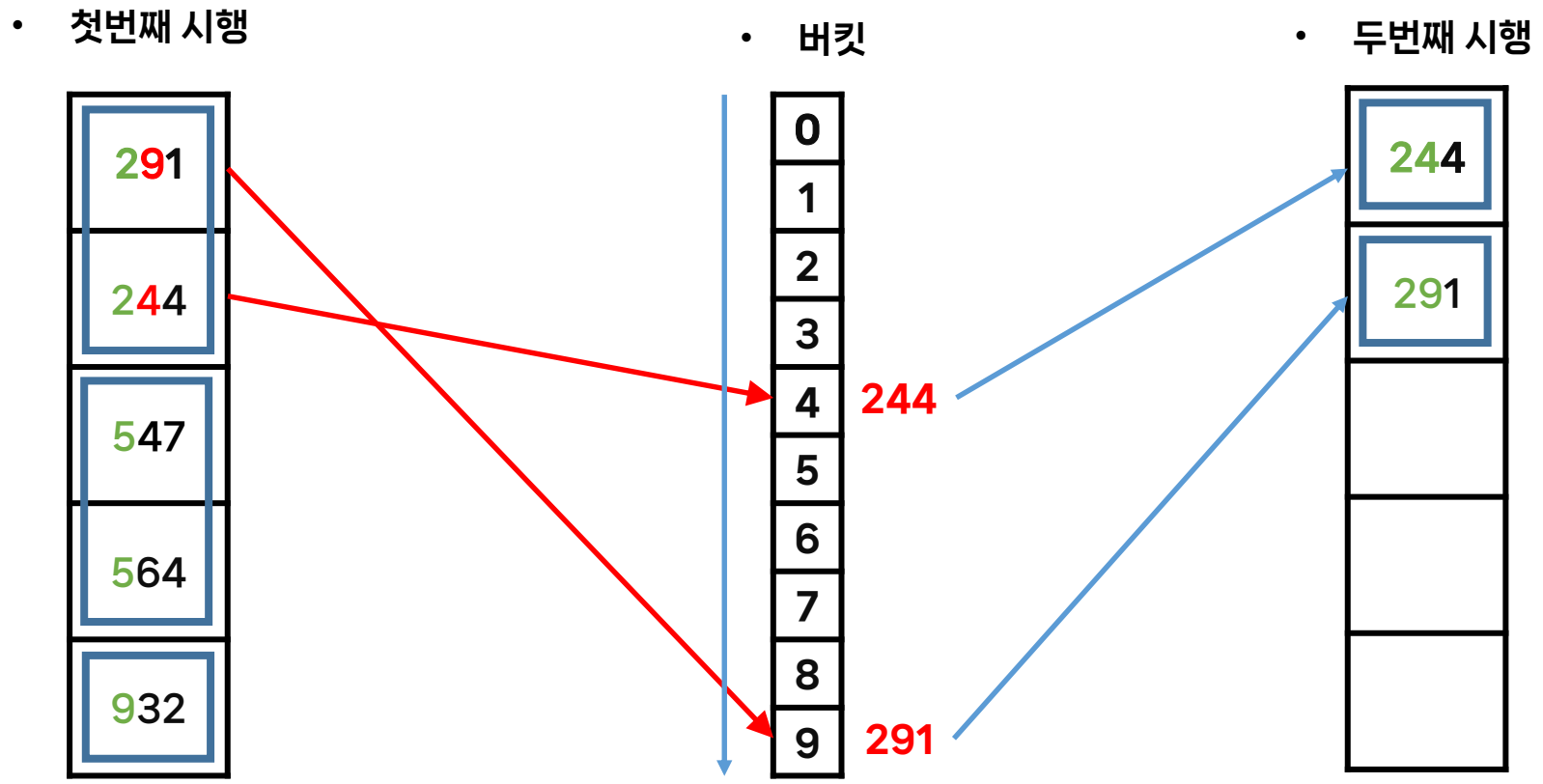


최대 유효 숫자에 대한 정렬 순서가 유지되지 않는다.

MSD 기수 정렬

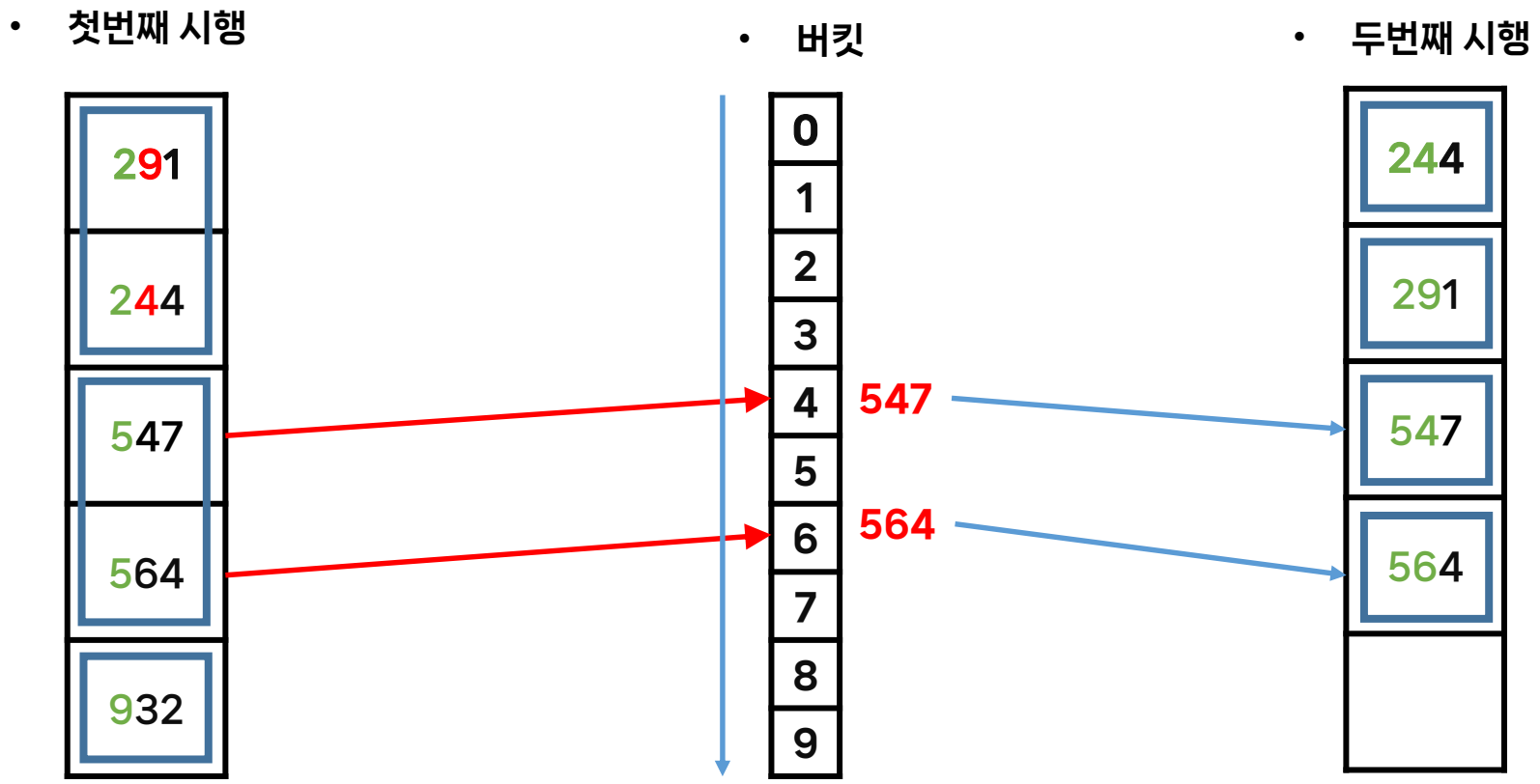

• 정렬할 리스트
• 버킷
• 첫번째 시행
547
291
932
244
564
0
1
2
3
4
5
6
7
8
9
291 244
547 564
932
291
244
547
564
932

MSD 기수 정렬



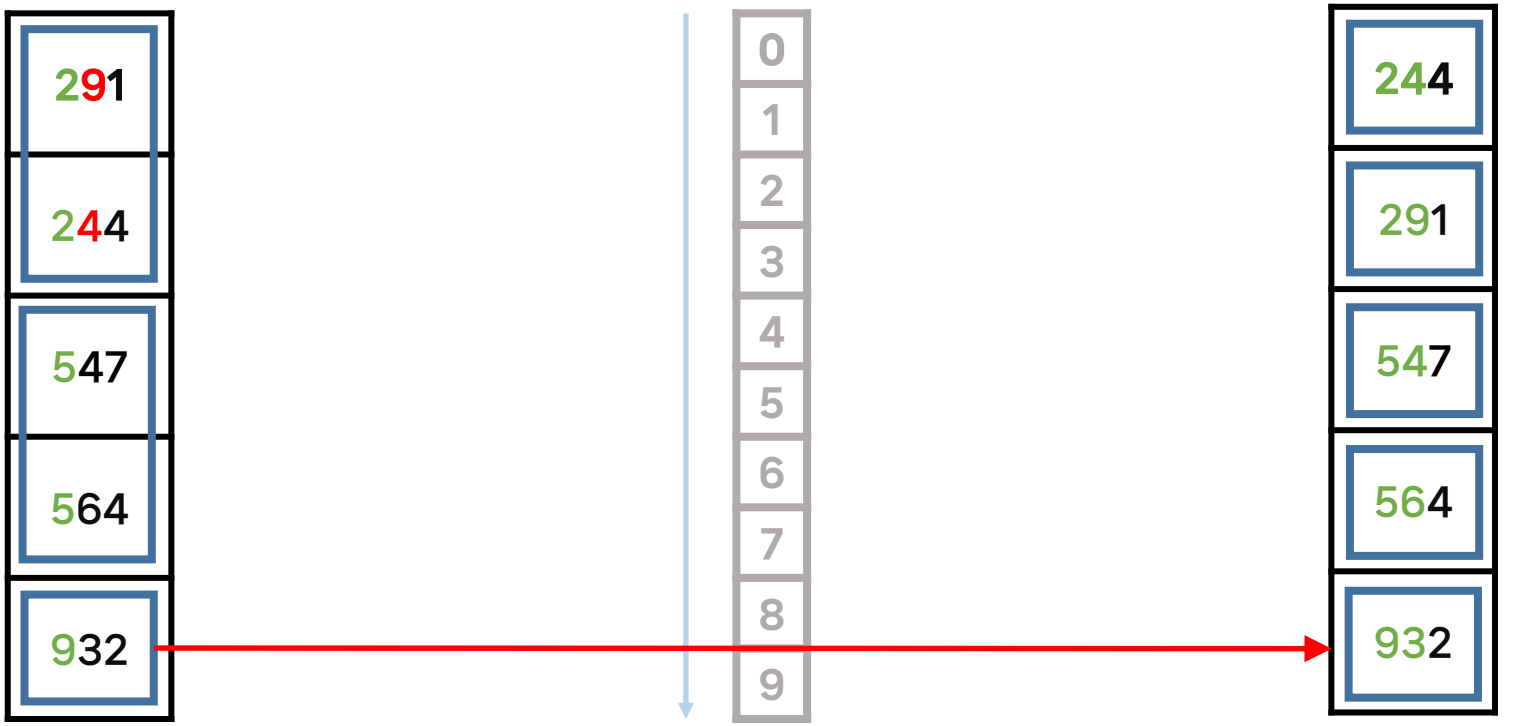
다음 시행 전에 같은 버킷에 들어간 원소 끼리 그룹으로 묶고,
그룹의 크기가 1이 될 때 까지, 재귀적으로 그룹에 대해 정렬을 수행한다.

MSD 기수 정렬


첫번째 시행
버킷
두번째 시행
291
244
547
564
932
0
1
2
3
4
5
6
7
8
9
547
564
244
291
547
564

MSD 기수 정렬

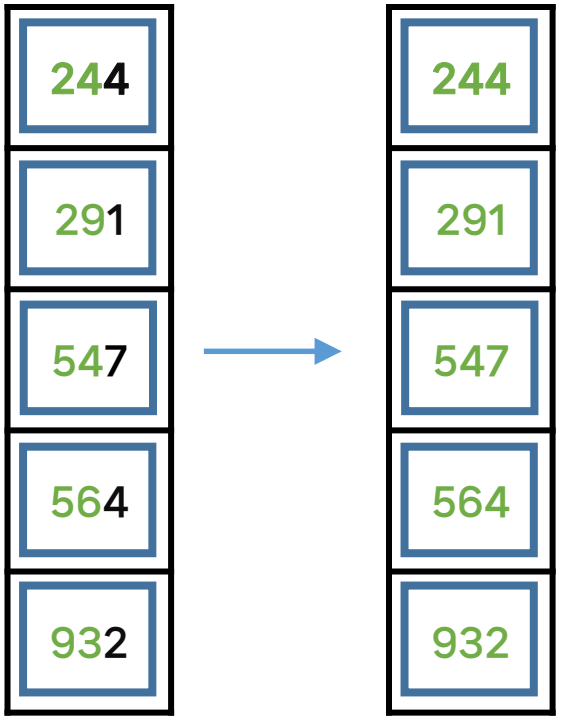
- 첫번째 시행
- 버킷
- 두번째 시행



(1개의 원소 뿐일 경우 이미 정렬되었다.)

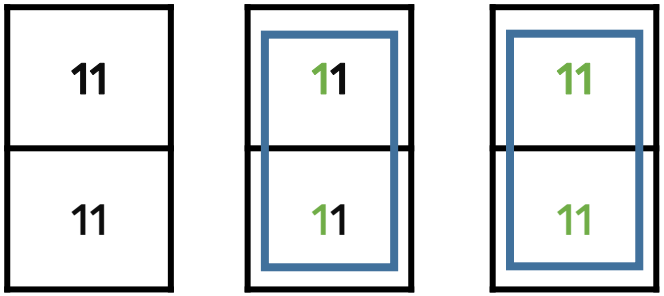
MSD 기수 정렬

- 두번째 시행



- 정렬 완료

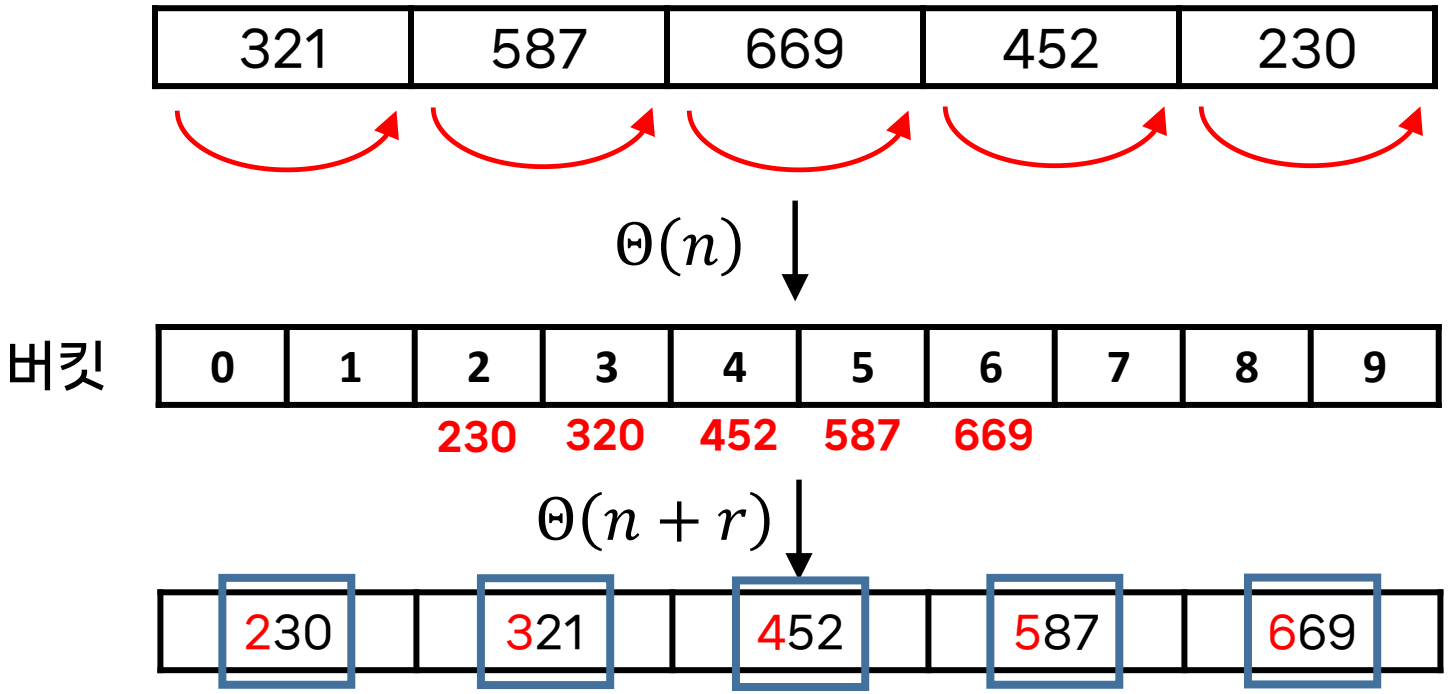
- 중복 원소가 존재하는 경우



다음 시행으로 넘어가기 전에
구분한 배열 내 모든 그룹이
1개의 원소만 가진다면 정렬이 완료된다.

모든 자릿수에 대해서 비교가 끝난 경우에도
그대로 정렬이 종료된다.

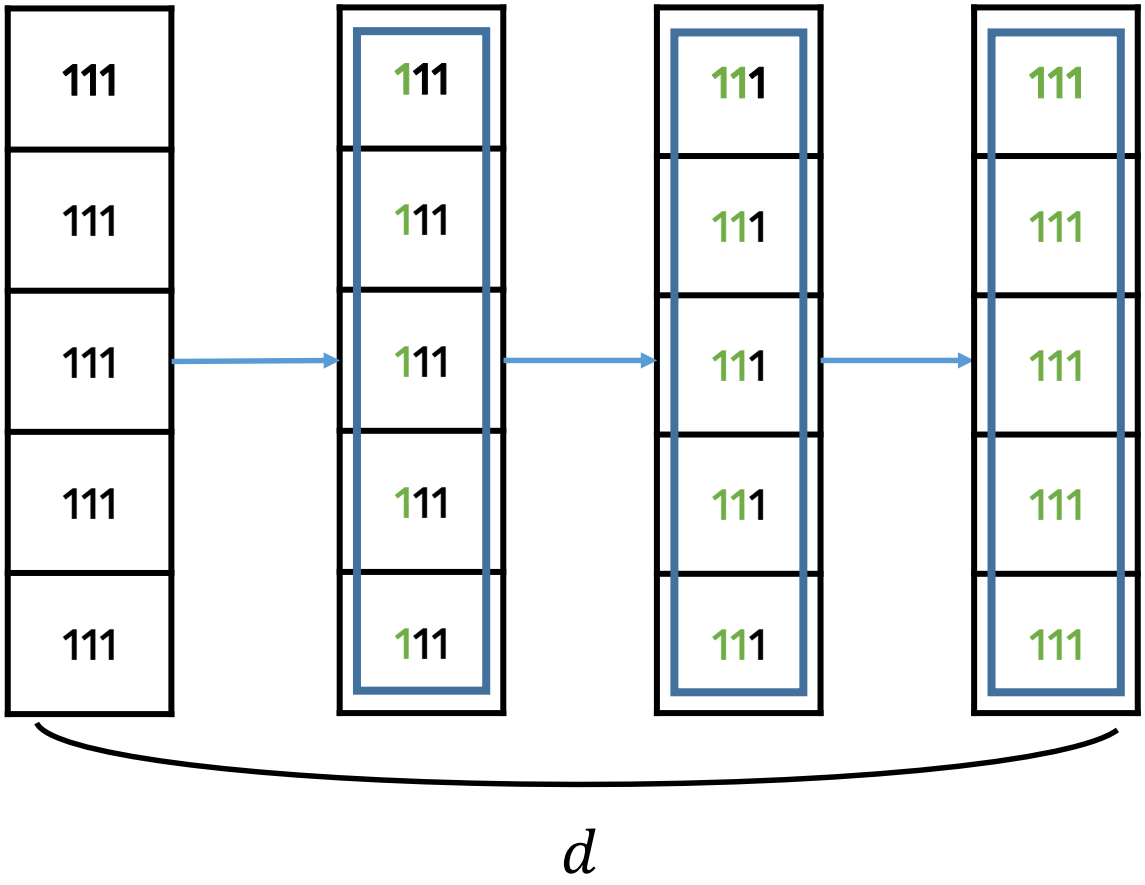
시간 복잡도 (최상의 경우)



최대 유효 숫자를 기준으로 정렬했을 때, 원소가 모두 정렬 되었을 경우

더 이상 정렬을 진행하지 않으므로 $\therefore \Theta(n+r)$

시간 복잡도 (최악의 경우)



1개 시행에 대해서
$\Theta(n+r)$

모든 자릿수에 대해
시행을 수행 d

$\therefore \Theta(\mathrm{d}(n+r))$

최대 유효 숫자 부터, **최소 유효 숫자**까지 기준으로 잡고 정렬하는 경우

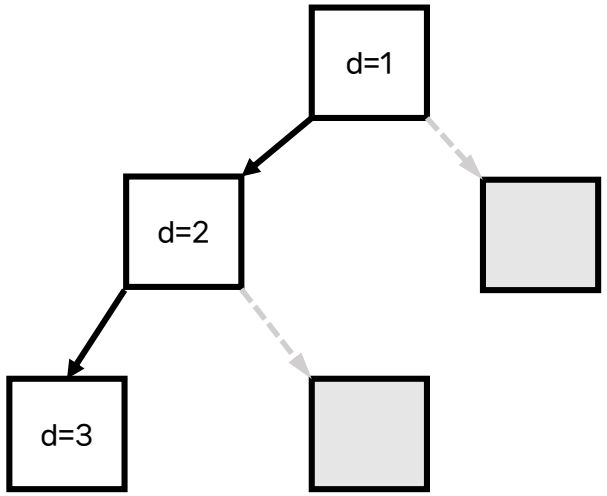
공간 복잡도

- MSD 기수 정렬은 재귀(Recursion)를 이용하는 알고리즘
- 빈 큐를 담은 배열의 공간 복잡도 : $\Theta(r)$
- 재귀적 동작으로 인해 동시에 존재하는 버킷의 수 : 최대 d 개 존재 할 수 있다.

예) $d = 3$일 경우

프로시저 시작 안됨

프로시저 실행 중



$\therefore O(n + dr)$

MSD 기수 정렬의 상대적 특성

- 특성

- 모든 자릿수에 대해 정렬을 하지 않더라도, 정렬이 완료되는 경우가 존재한다. (최상의 경우 포함)
- 재귀적인 동작으로 인해 공간 복잡도가 기수 정렬에 비해 높다.

Thank You