자료구조

L09 Sorting (3)

2022년 1학기

국민대학교 소프트웨어학부

Overview

- **Heapsort**
- Binsort
- Radix Sort

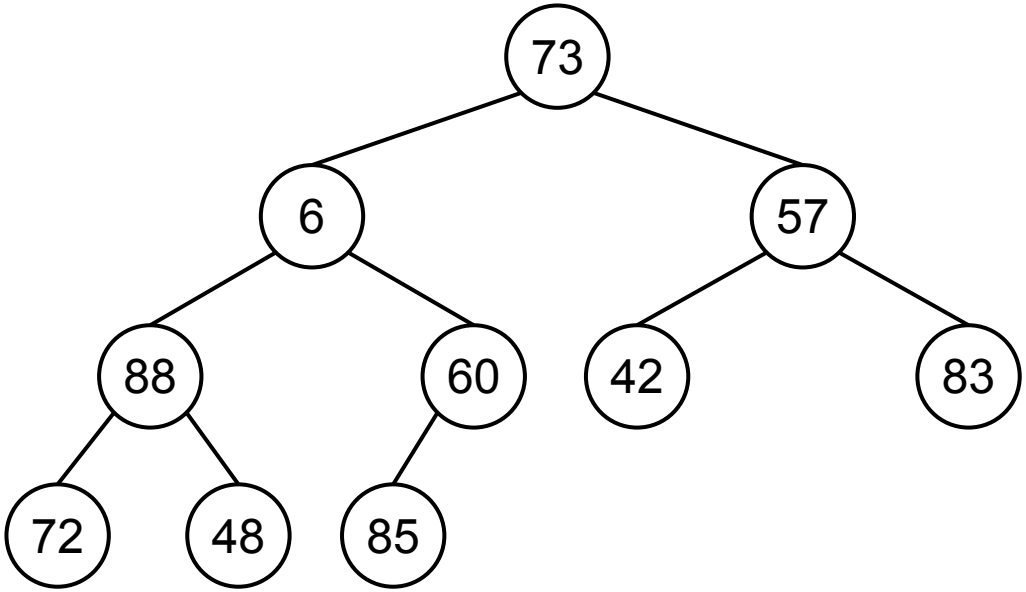
Heapsort

- Heap 자료구조 활용

- 크기 n의 배열이 입력으로 주어지면
  - max-heap 을 만들고
  - max값을 n번 꺼낸다

Heapsort example

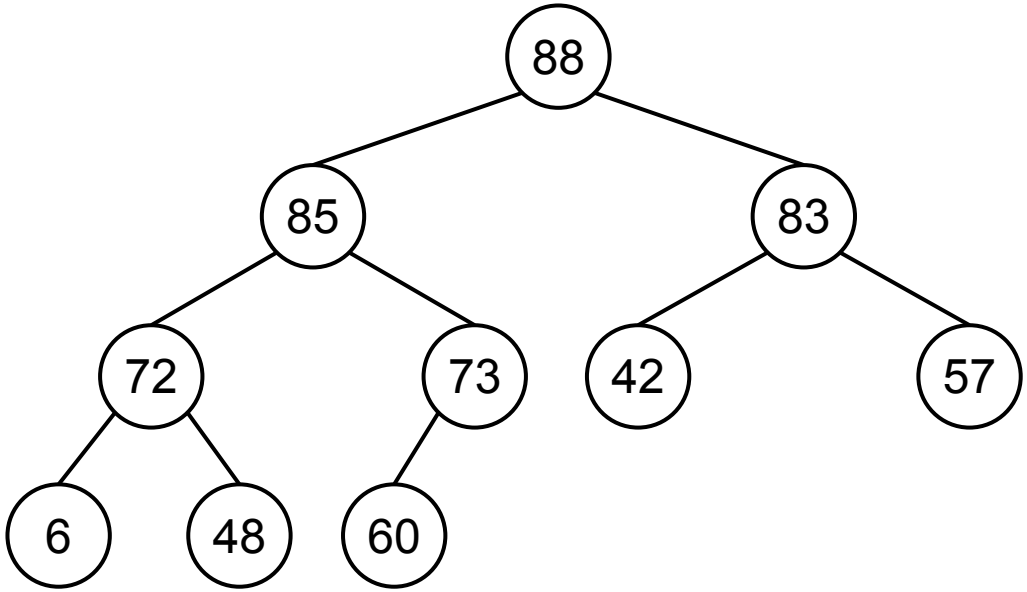
- Original numbers



73	6	57	88	60	42	83	72	48	85

Heapsort example

- Build heap
  - Time complexity?

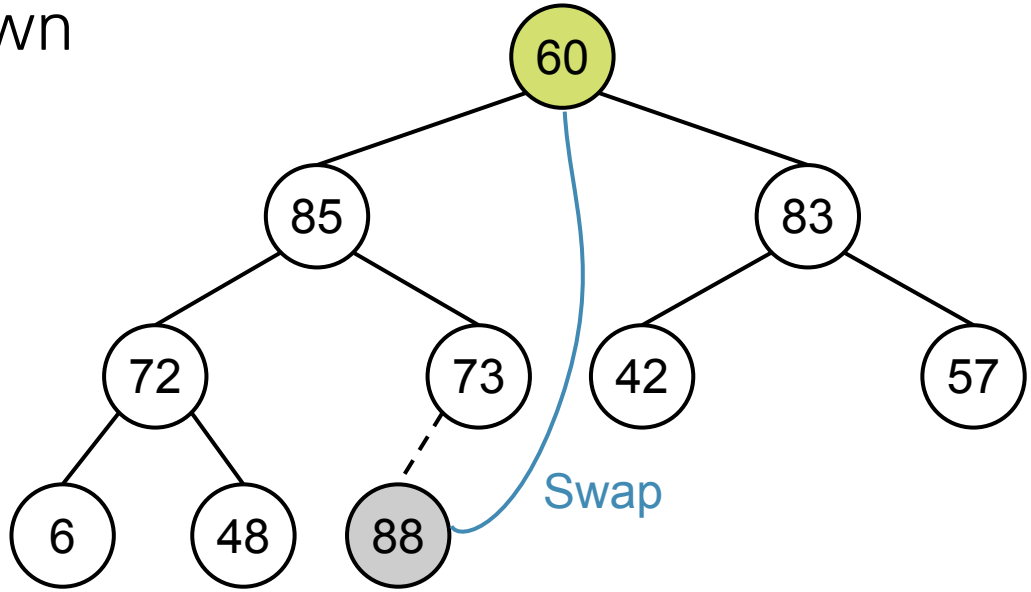


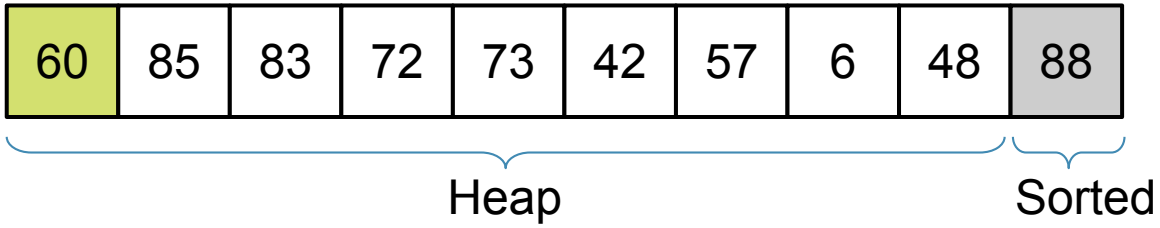
88	85	83	72	73	42	57	6	48	60

Heapsort example

```
public E removemax() {
    swap(0, --n);
    if (n != 0) siftdown(0);
    return Heap[n];
}
```

- Remove 88
  - **Swap the root and the last node**
  - Siftdown

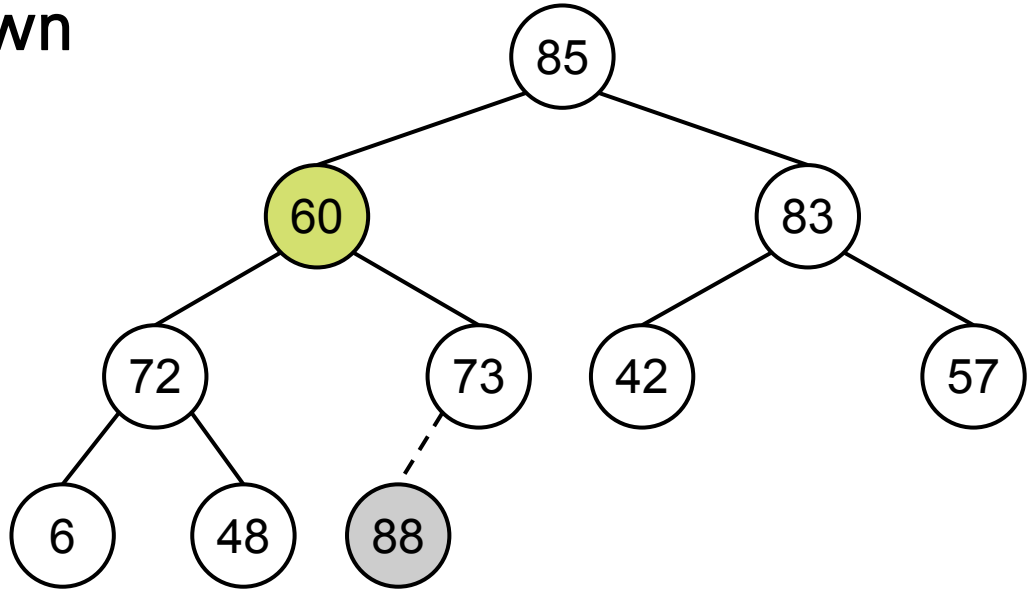


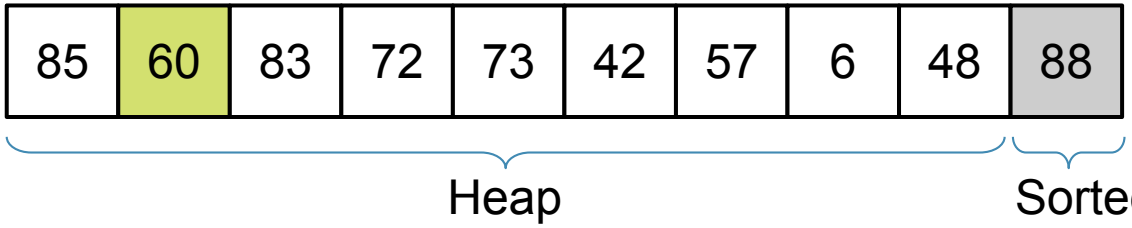


Heapsort example

```
public E removemax() {
    swap(0, --n);
    if (n != 0) siftdown(0);
    return Heap[n];
}
```

- Remove 88
  - Swap the root and the last node
  - **Siftdown**

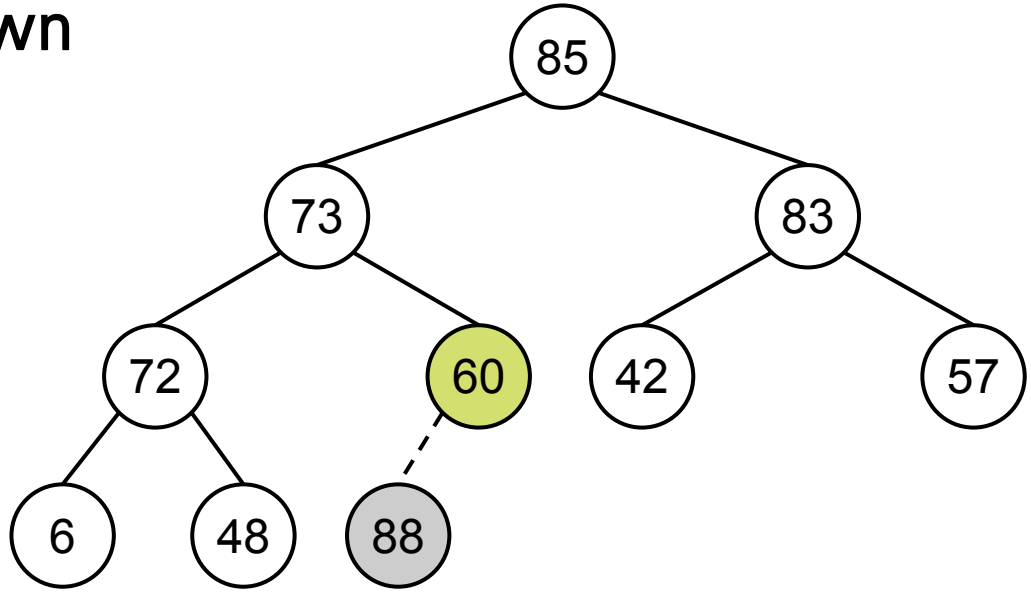


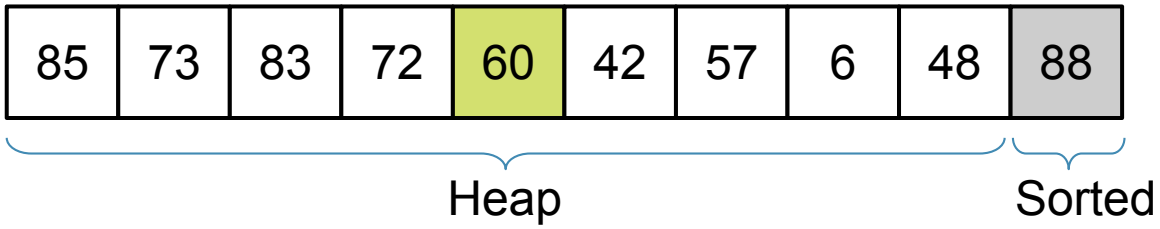


Heapsort example

```
public E removemax() {
    swap(0, --n);
    if (n != 0) siftdown(0);
    return Heap[n];
}
```

- Remove 88
  - Swap the root and the last node
  - **Siftdown**

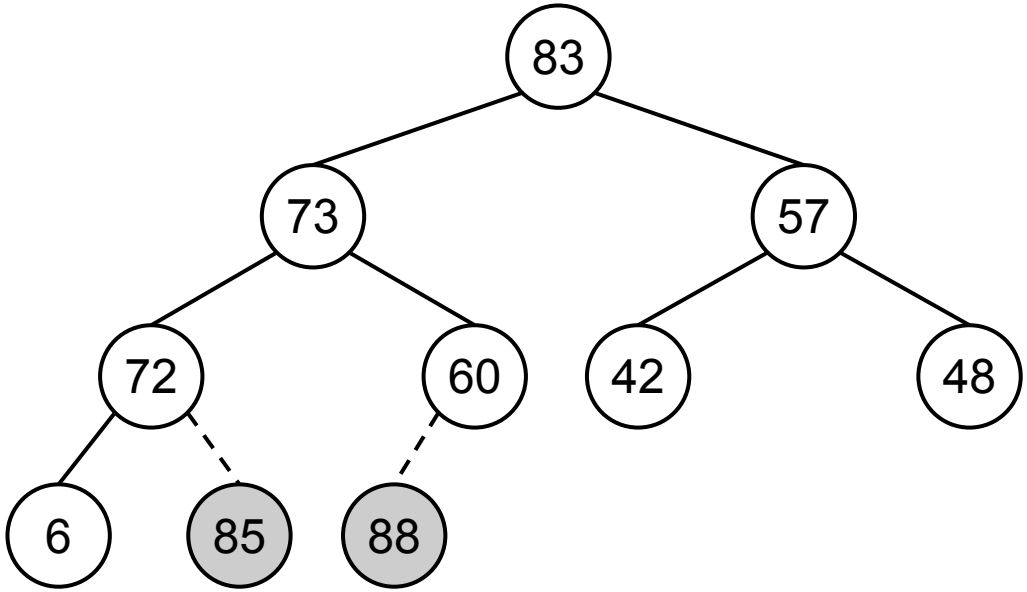


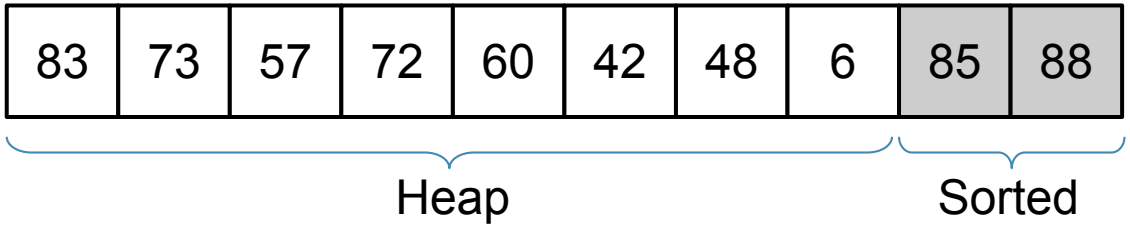


Heapsort example

```
public E removemax() {
    swap(0, --n);
    if (n != 0) siftdown(0);
    return Heap[n];
}
```

- Remove 85

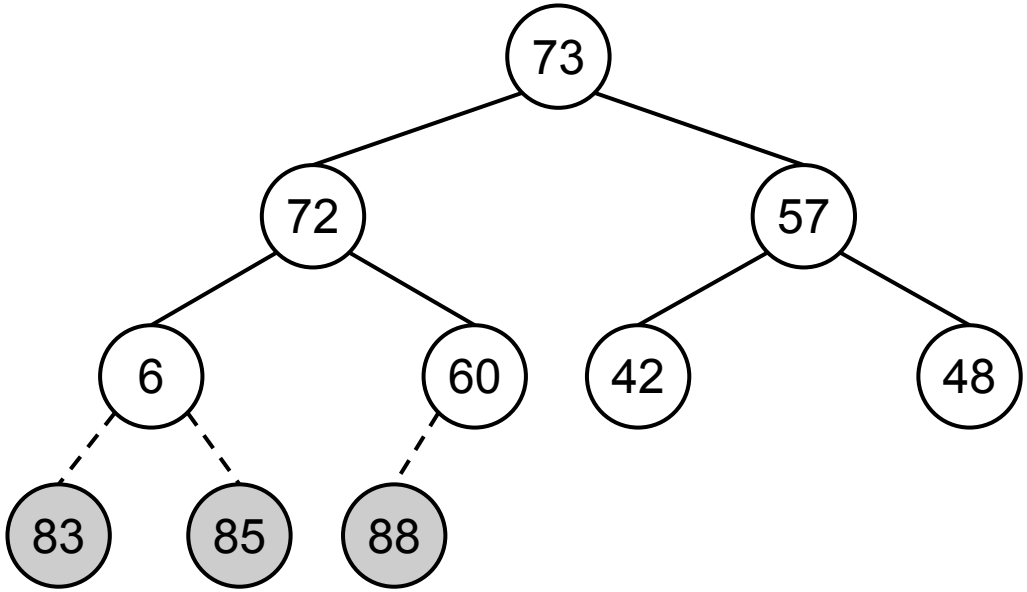


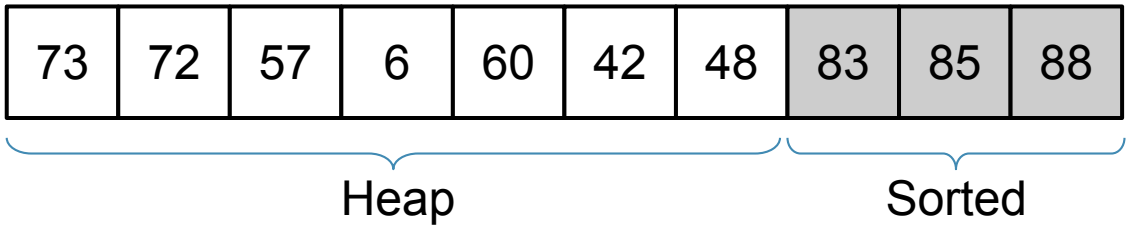


Heapsort example

```
public E removemax() {
    swap(0, --n);
    if (n != 0) siftdown(0);
    return Heap[n];
}
```

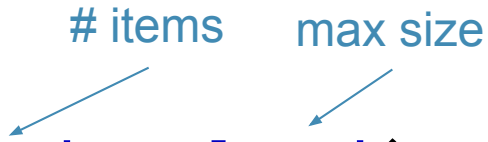
- Remove 83





Heapsort

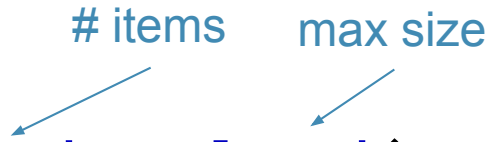
```
static <E extends Comparable<? super E>>
void heapsort(E[] A) {
    MaxHeap<E> H = new MaxHeap<E>(A, A.length, A.length);
    for (int i = 0; i < A.length; i++)
        H.removemax();
}
```



- heapsort는 **in-place** sort이다
  - In-place sort: 추가 메모리 사용 없이 정렬함
- Cost of heapsort:
- Cost of finding k largest elements:

Heapsort

```
static <E extends Comparable<? super E>>
void heapsort(E[] A) {
    MaxHeap<E> H = new MaxHeap<E>(A, A.length, A.length);
    for (int i = 0; i < A.length; i++)
        H.removemax();
}
```



- heapsort는 **in-place** sort이다
  - In-place sort: 추가 메모리 사용 없이 정렬함
- Cost of heapsort: $\Theta(n + n \log n) = \Theta(n \log n)$
- Cost of finding k largest elements: $\Theta(n + k \log n)$

If k is small, Heapsort is very fast!

Overview

- Heapsort
- **Binsort**
- Radix Sort

Binsort Motivation

- 1부터 n까지의 key를 갖는 개체가 랜덤하게 섞여있는 배열이 있다.
- 가장 빠르게 정렬하는 방법은?

Binsort

- 단순하지만 효율적인 알고리즘:
  - 하지만, key가 0부터 n-1까지 순서대로 있을 때만 정렬가능

```
for (i=0; i<n; i++)
   B[A[i].key()] = A[i];
```

- 중복된 키가 있거나, key의 범위가 n보다 큰 경우에도 처리 가능하게 하려면?
- Main Idea:
  - 중복된 키 허용: linked list를 담은 array 사용하기.
    - 각각의 bin이 linked list를 담는다
  - 배열 B를 더 크게 만든다 (크기가 **MaxKeyValue**+1이 되도록)

The max key value from the input

Binsort

```
static void binsort(Integer A[]) {
   List<Integer>[] B =
             (LList<Integer>[]) new LList[MaxKey];
   Integer item;
   for (int i = 0; i < MaxKey; i++)
      B[i] = new LList<Integer>();
   for (int i = 0; i < A.length; i++)
      B[A[i]].append(A[i]);
   for (int i = 0; i < MaxKey; i++)
      for (B[i].moveToStart();
         (item = B[i].getValue()) != null; B[i].next())
         output(item);
}
```

Cost: $\Theta(n + MaxKeyValue)$

Binsort

- Binsort의 강점
  - MaxKeyValue 가 작으면 빠르고, 공간 효율적이다.

- Binsort의 약점
  - MaxKeyValue가 크면, 느리고 공간 비효율적이다.

Overview

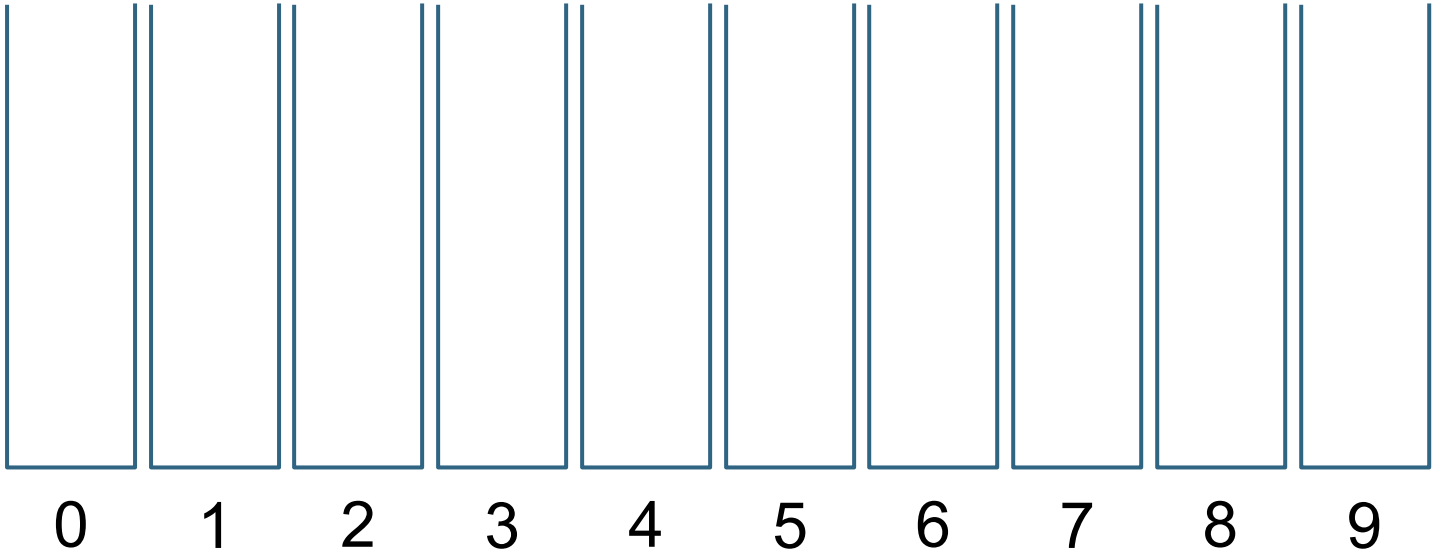
- Heapsort
- Binsort
- **Radix Sort**

Radix Sort

- Binsort를 확장한 버전
  - radix sort는 binsort보다 더 공간 효율적
- key가 숫자나 짧은 문자열인 경우에만 사용 가능
- k자리 숫자들을 정렬하려면 k번 반복하여 연산
  - i번째 연산은 i번째 자릿수에 대해서 binsort를 수행
    - 첫번째 연산: 가장 낮은 자릿수 (LSD Least Significant Digit) 정렬
  - 각 반복 연산마다 bin에 있는 아이템들을 순서대로 조회

Radix Sort Example

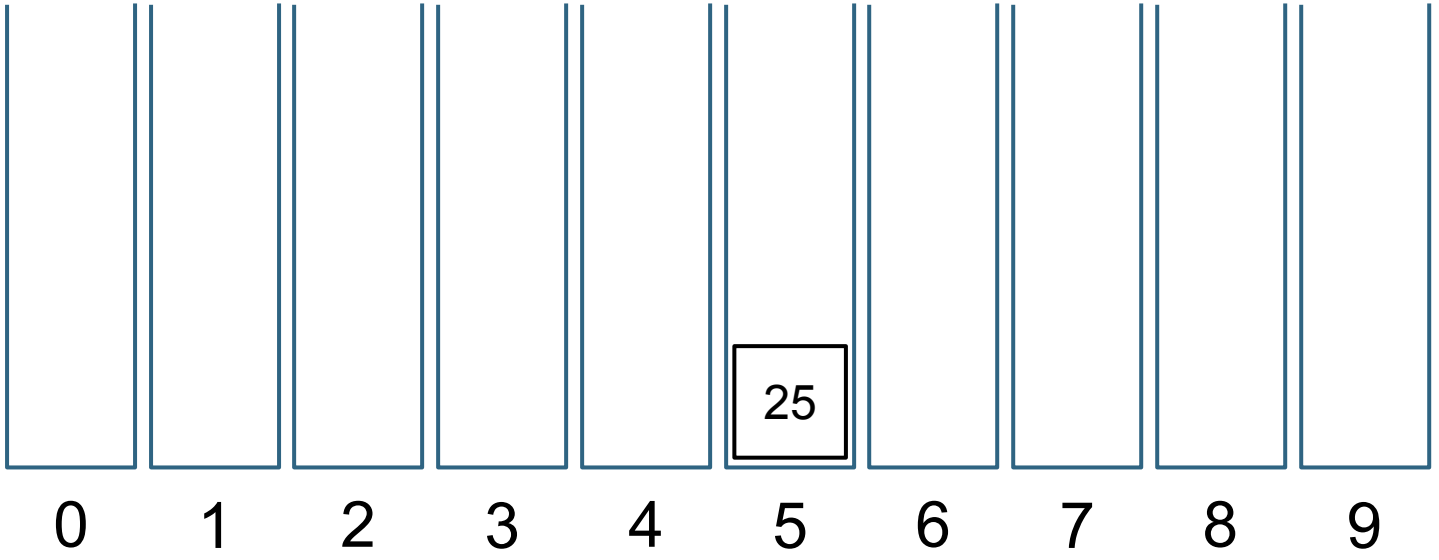
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84	28	72	5	67	25

Radix Sort Example

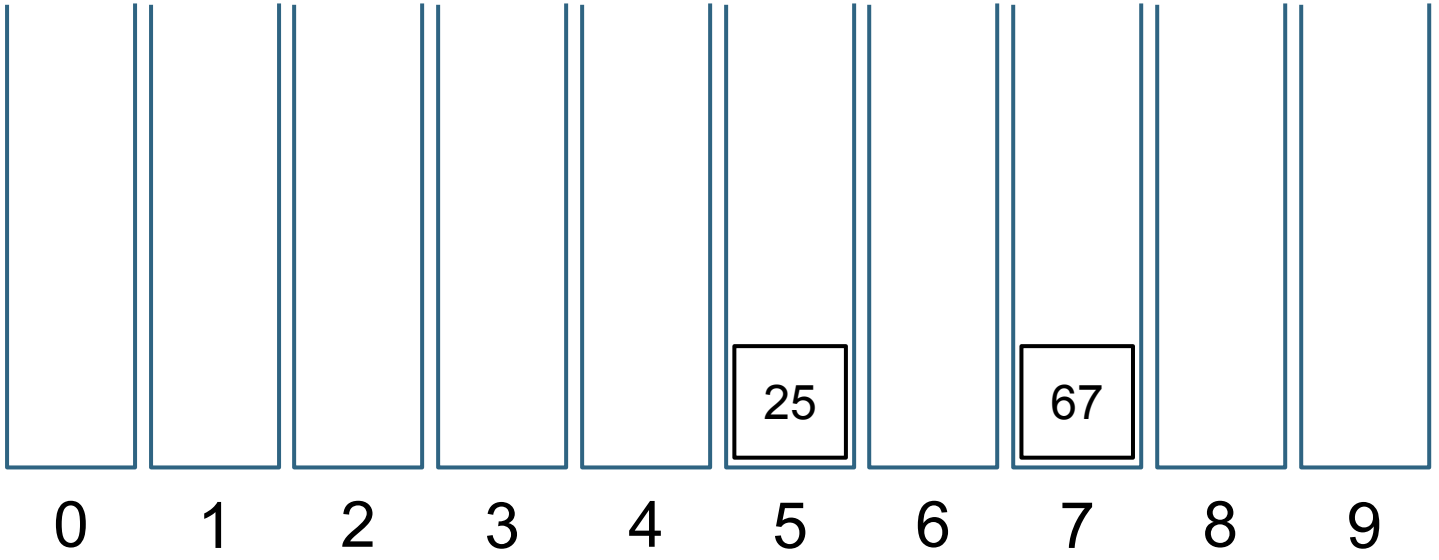
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84	28	72	5	67	

Radix Sort Example

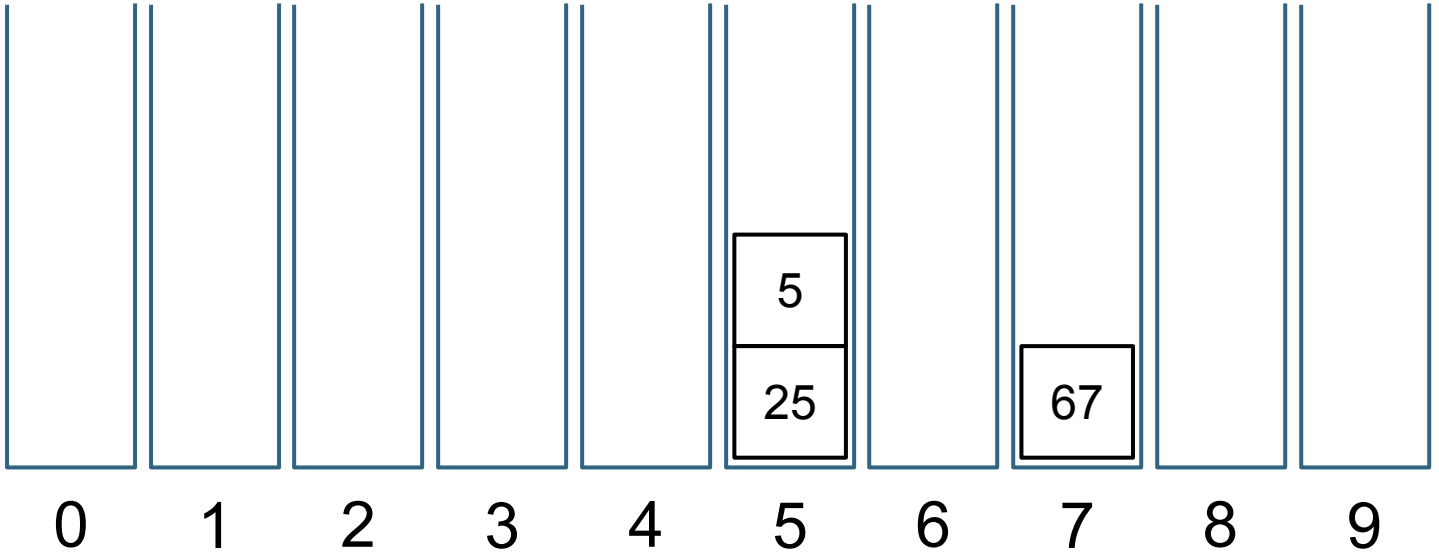
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84	28	72	5		

Radix Sort Example

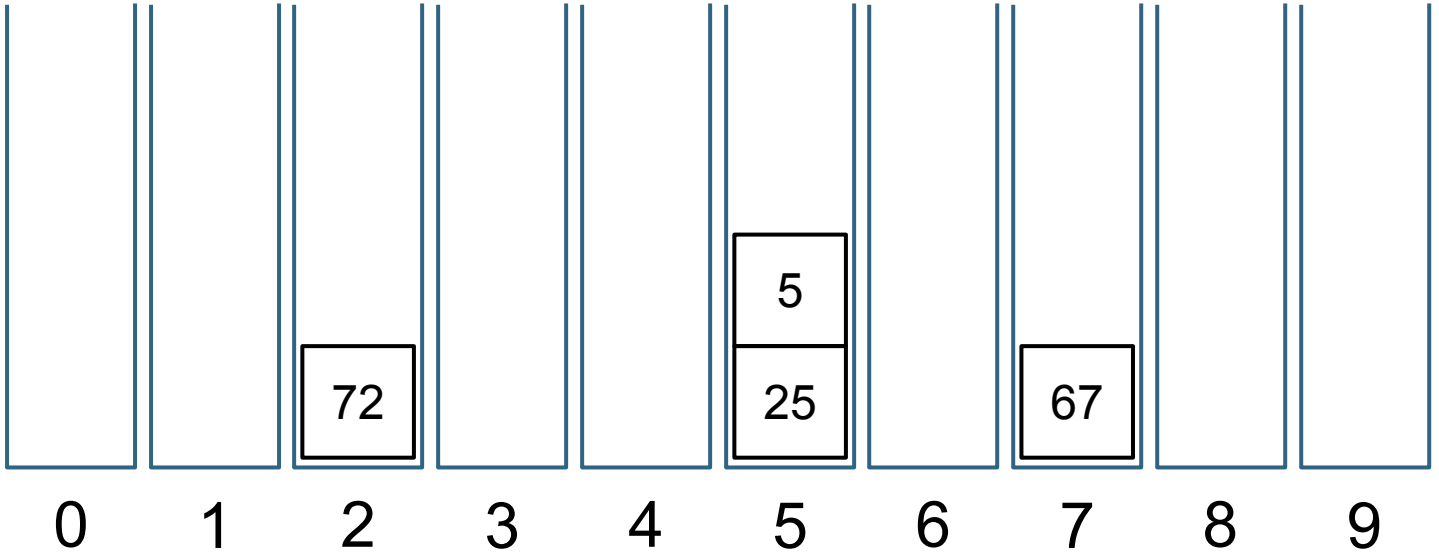
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84	28	72			

Radix Sort Example

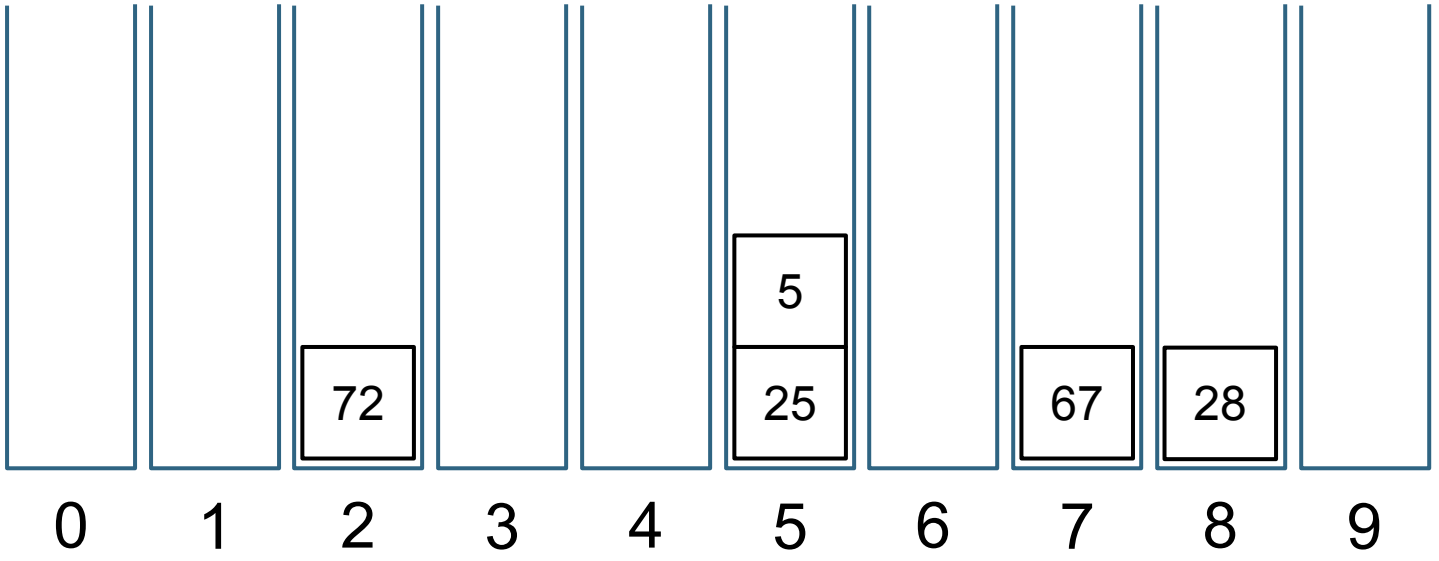
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84	28				

Radix Sort Example

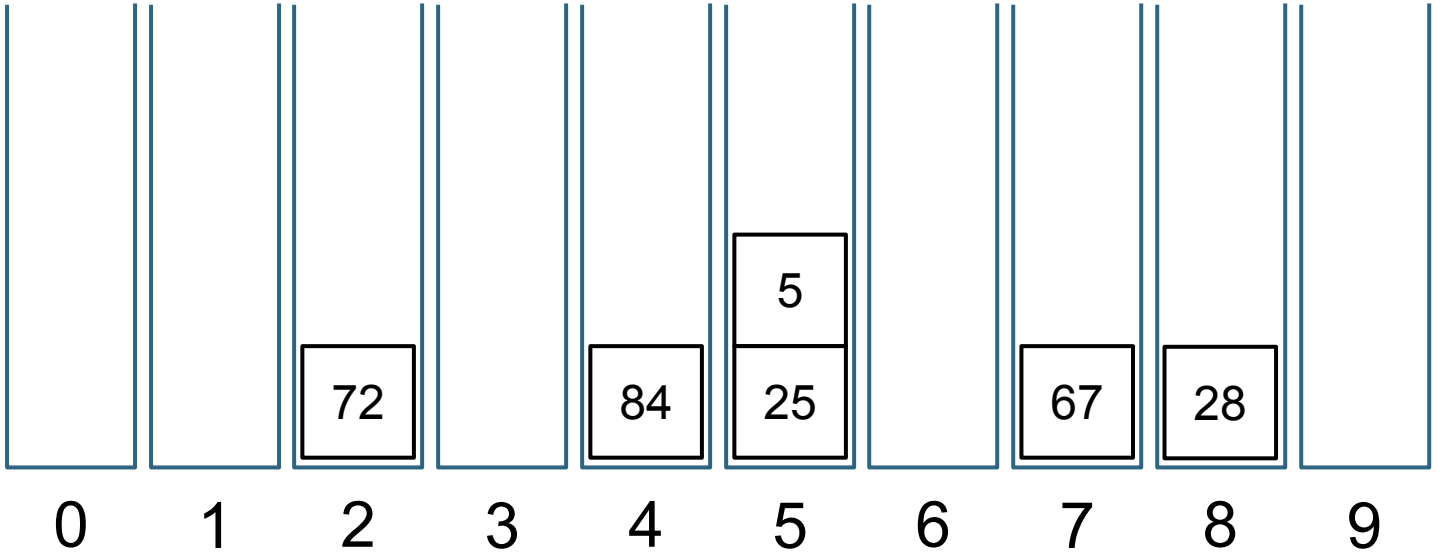
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23	84					

Radix Sort Example

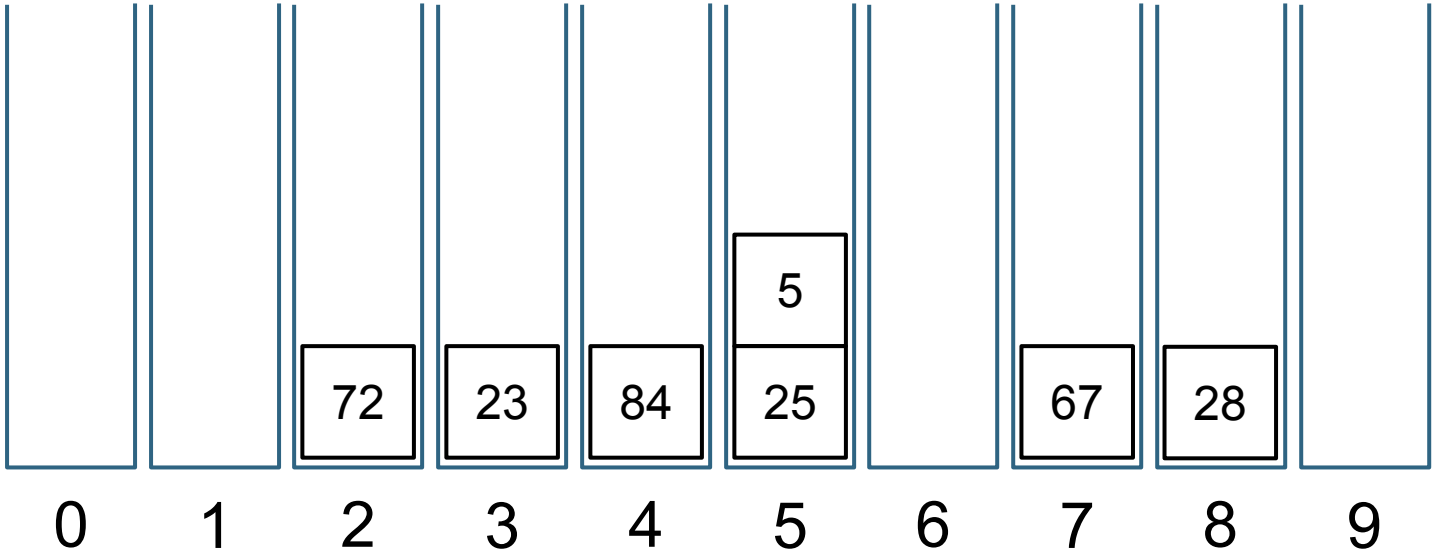
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17	23						

Radix Sort Example

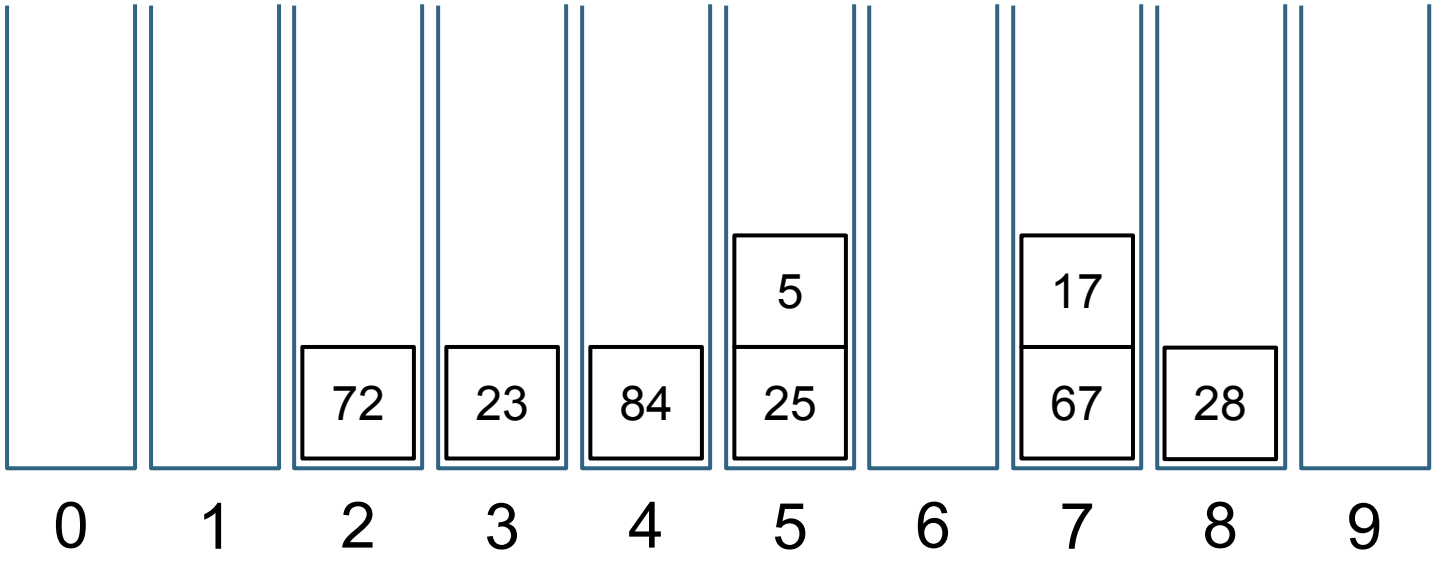
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97	17							

Radix Sort Example

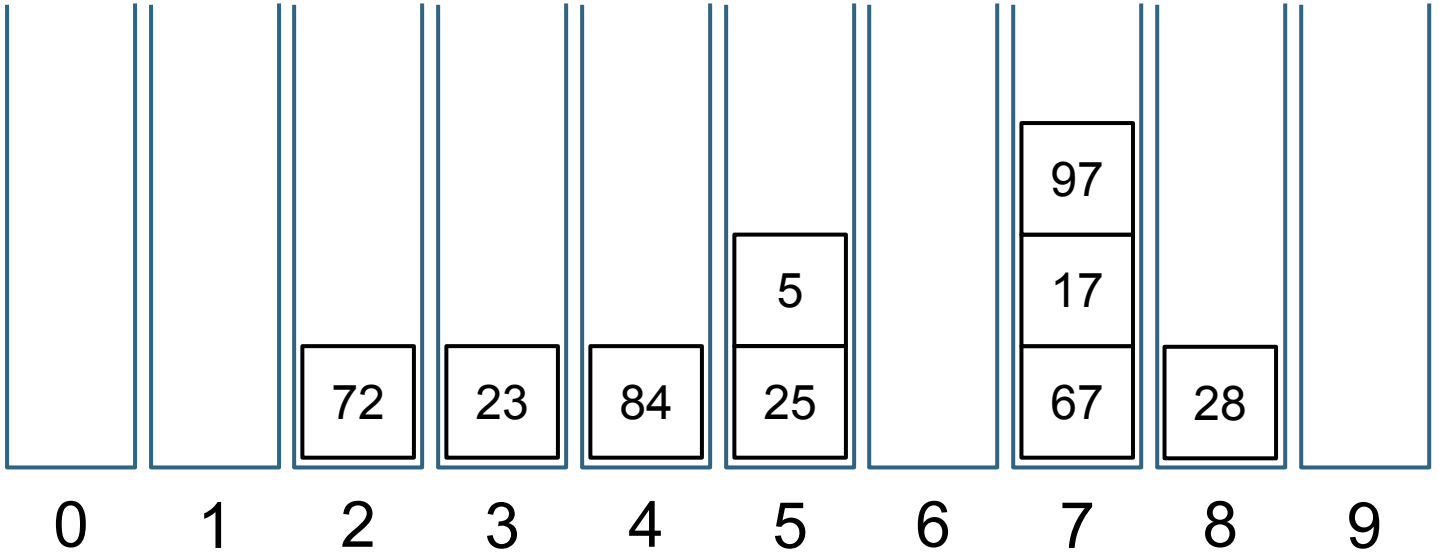
- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



27	91	1	97								

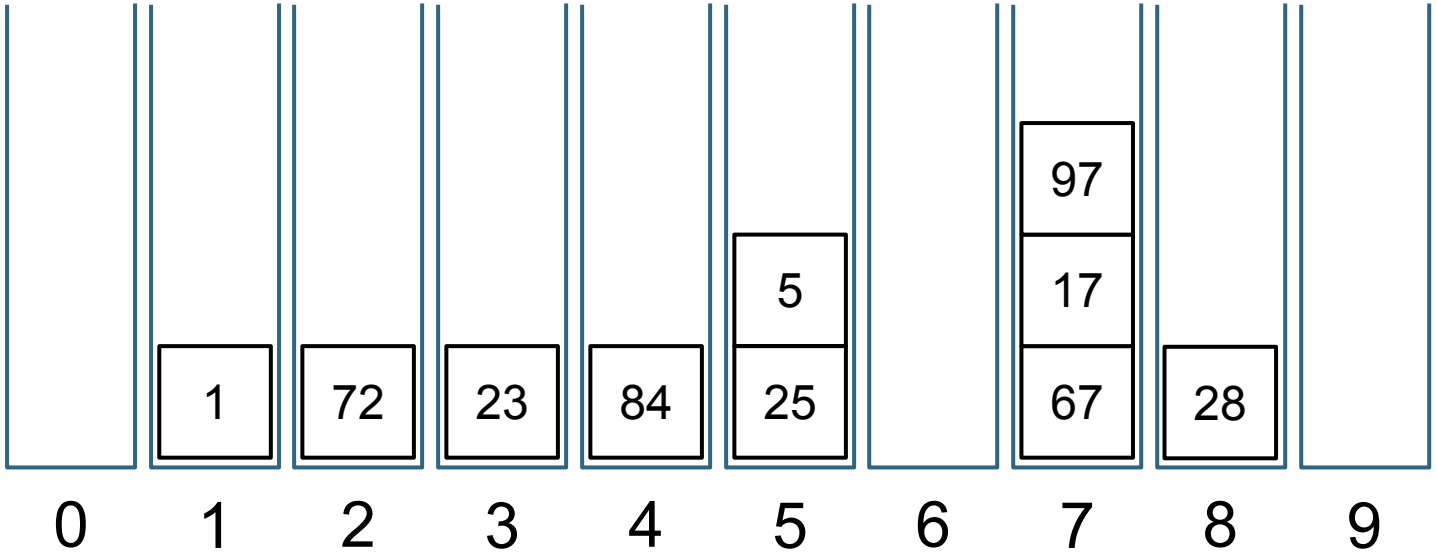
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



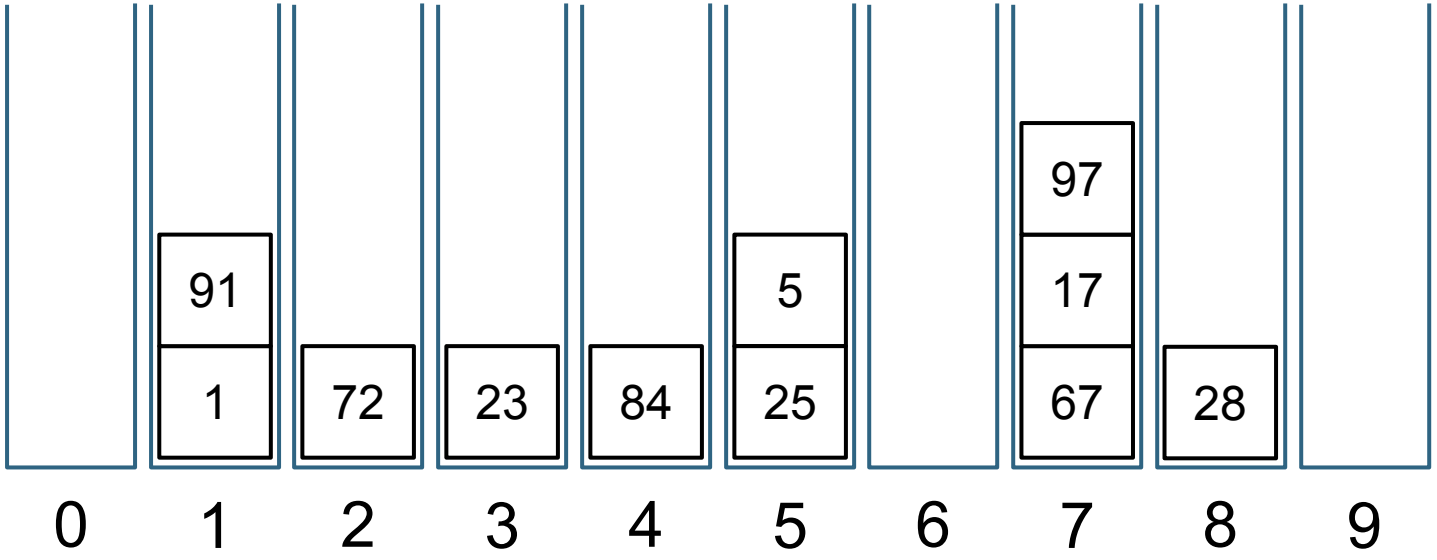
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



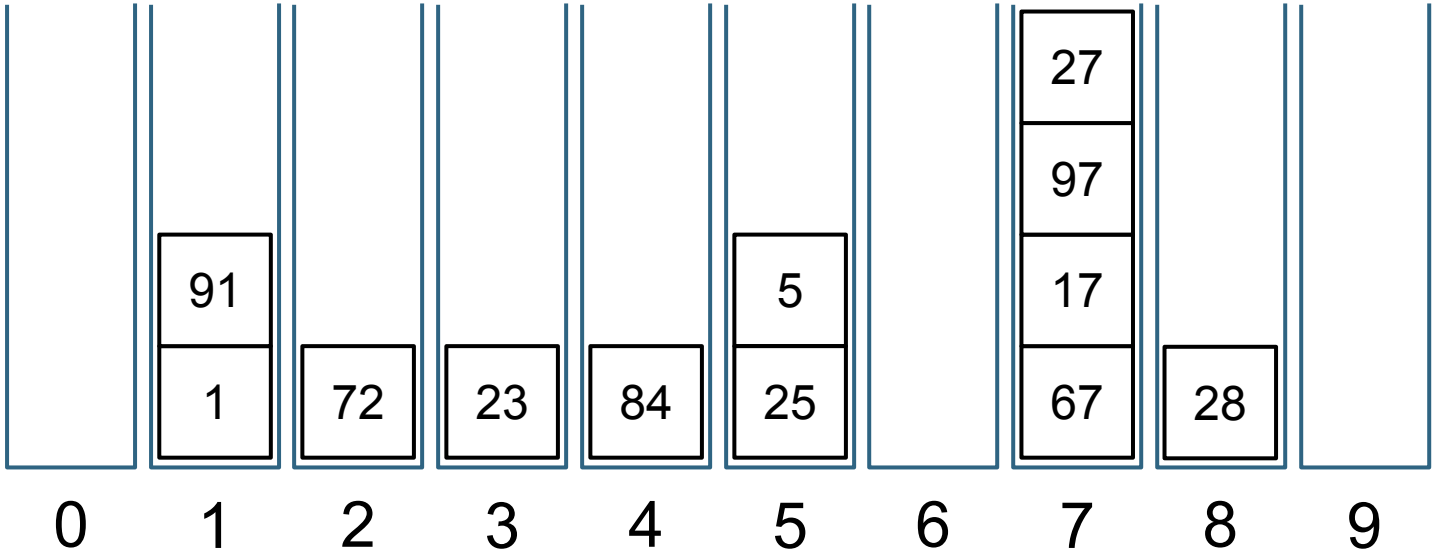
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



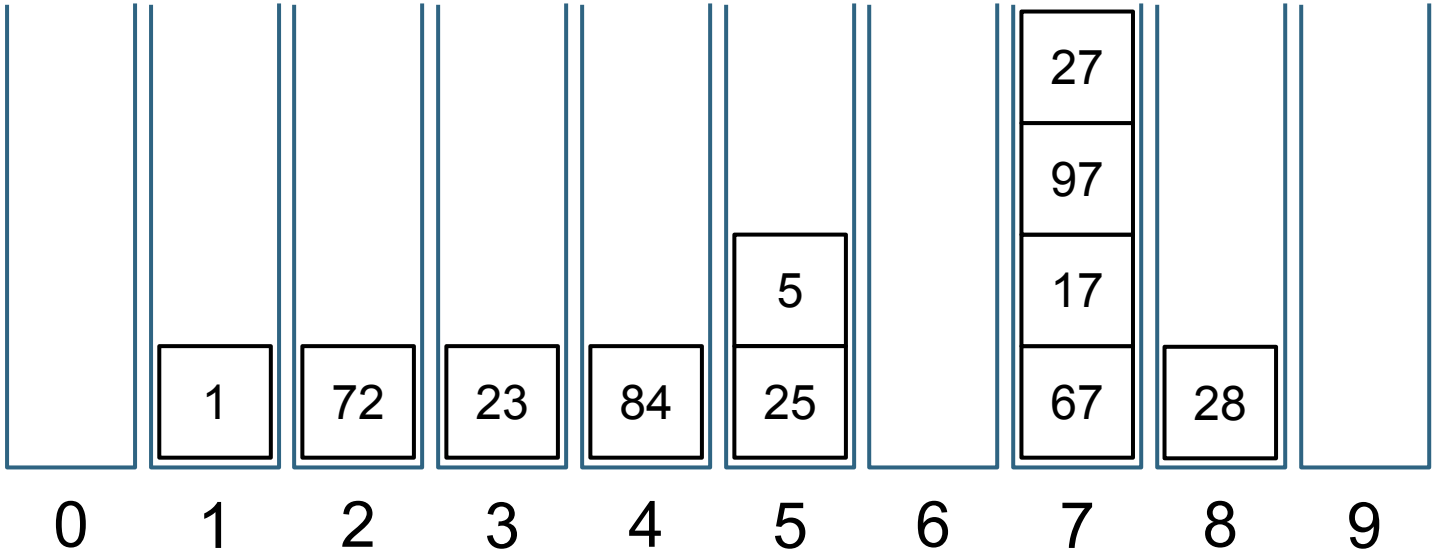
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements to bins*



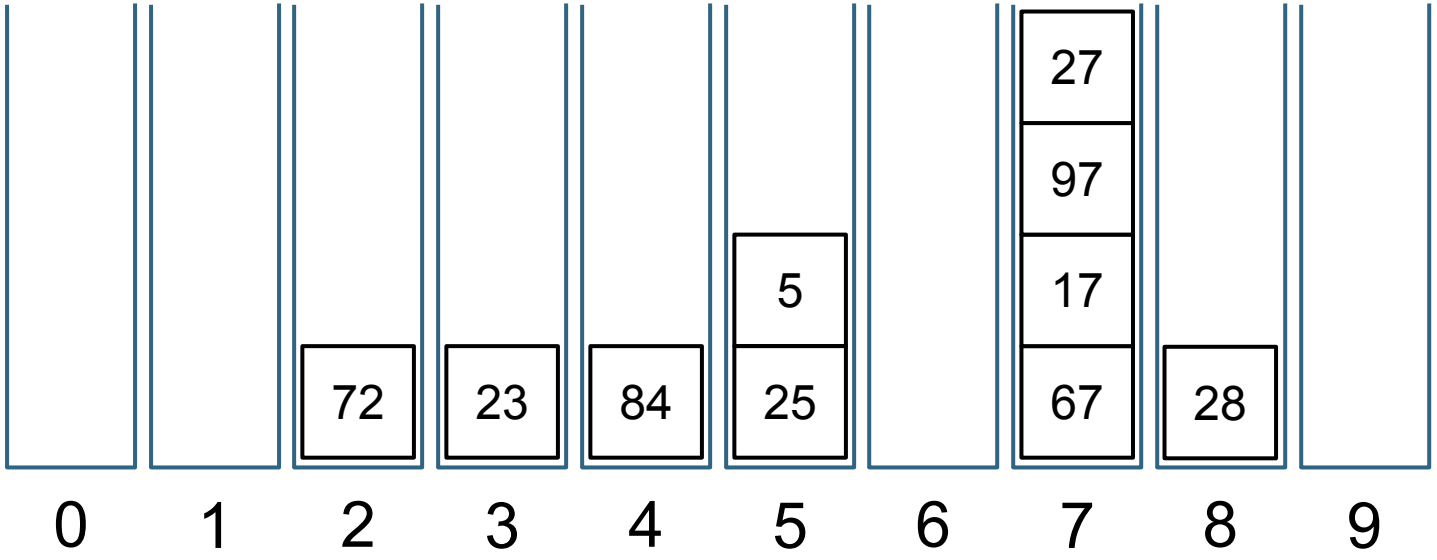
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



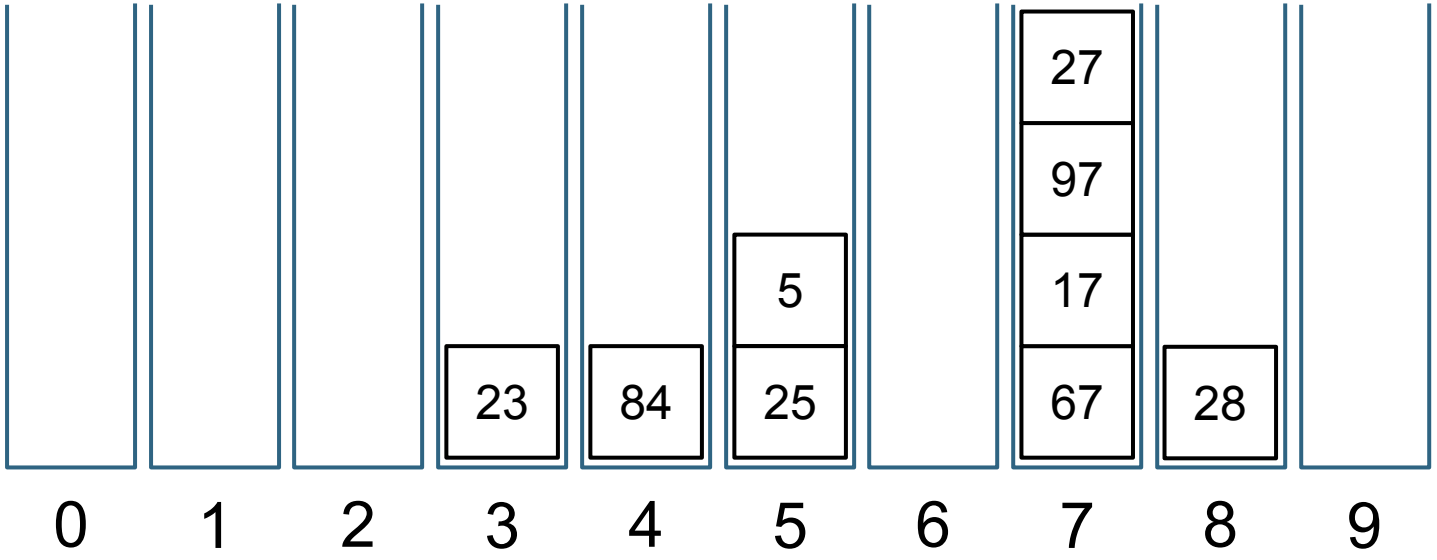
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



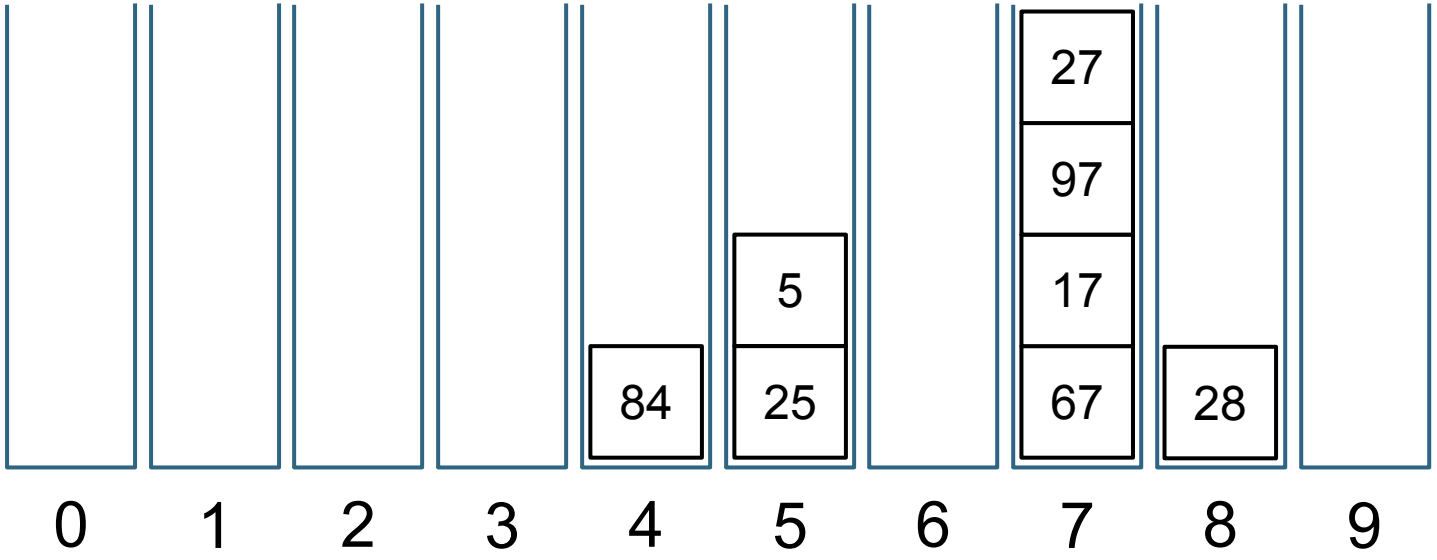
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



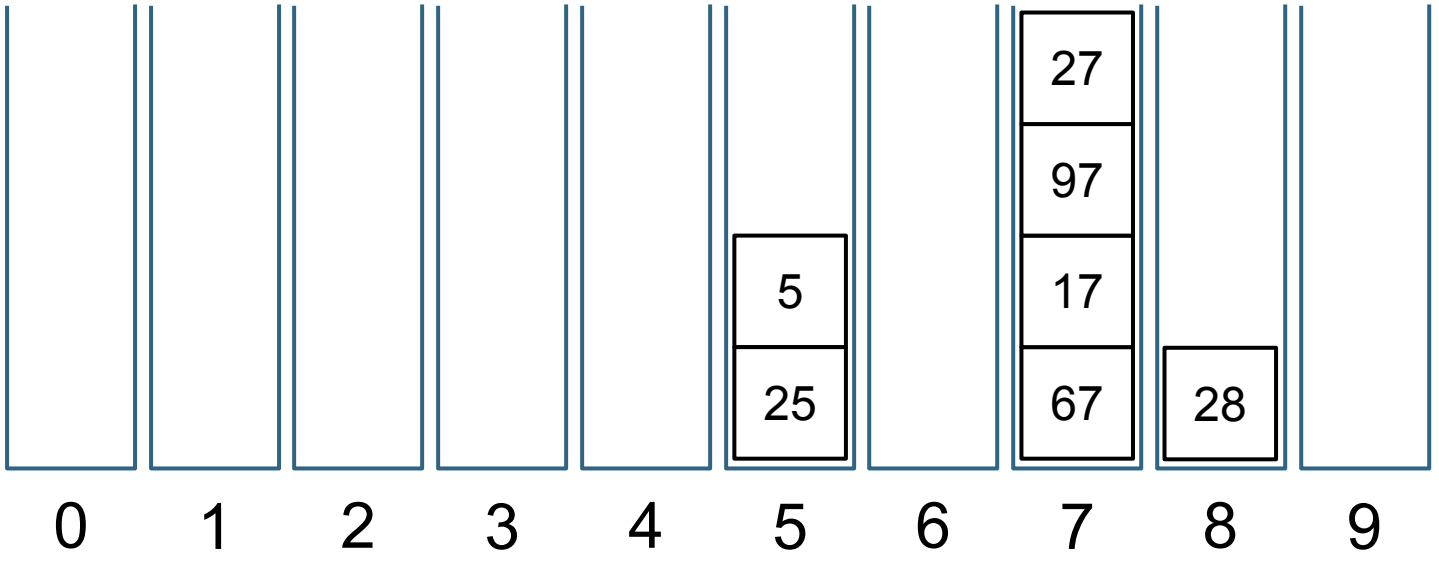
Radix Sort Example

- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



Radix Sort Example

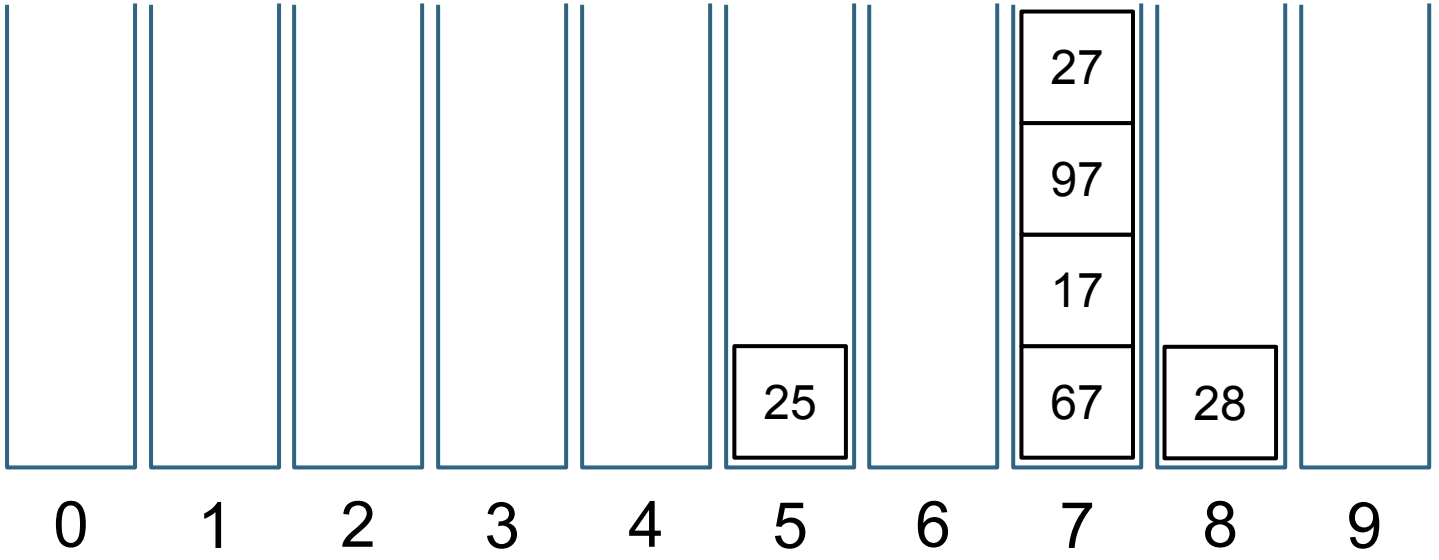
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84							

Radix Sort Example

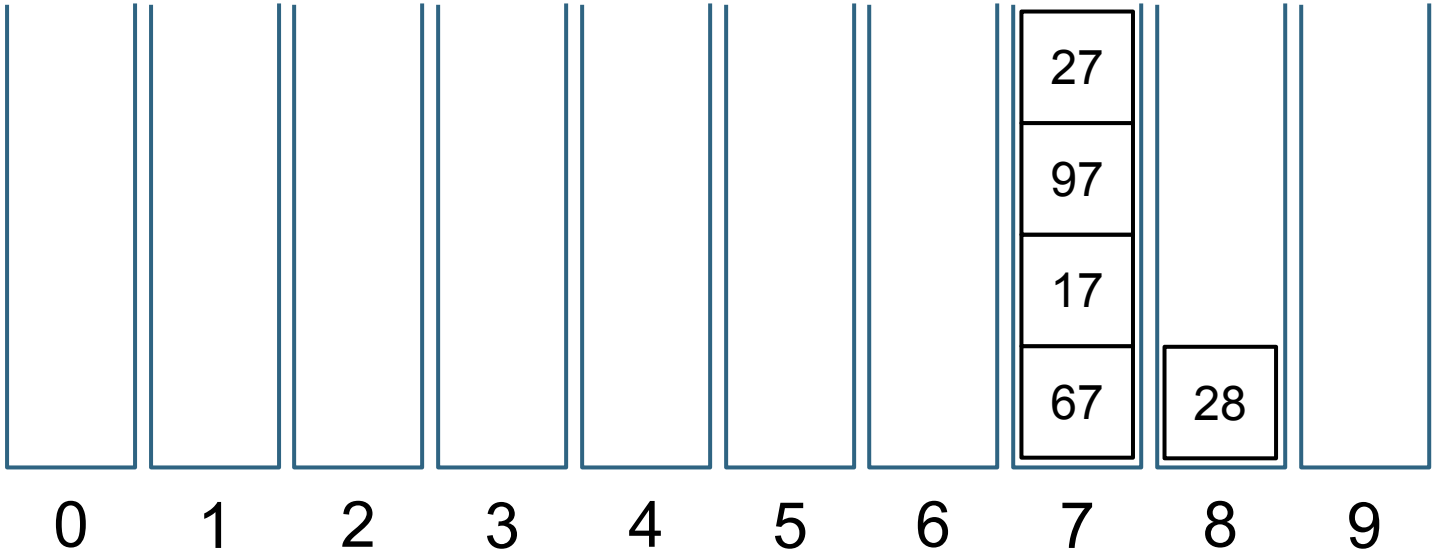
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5						

Radix Sort Example

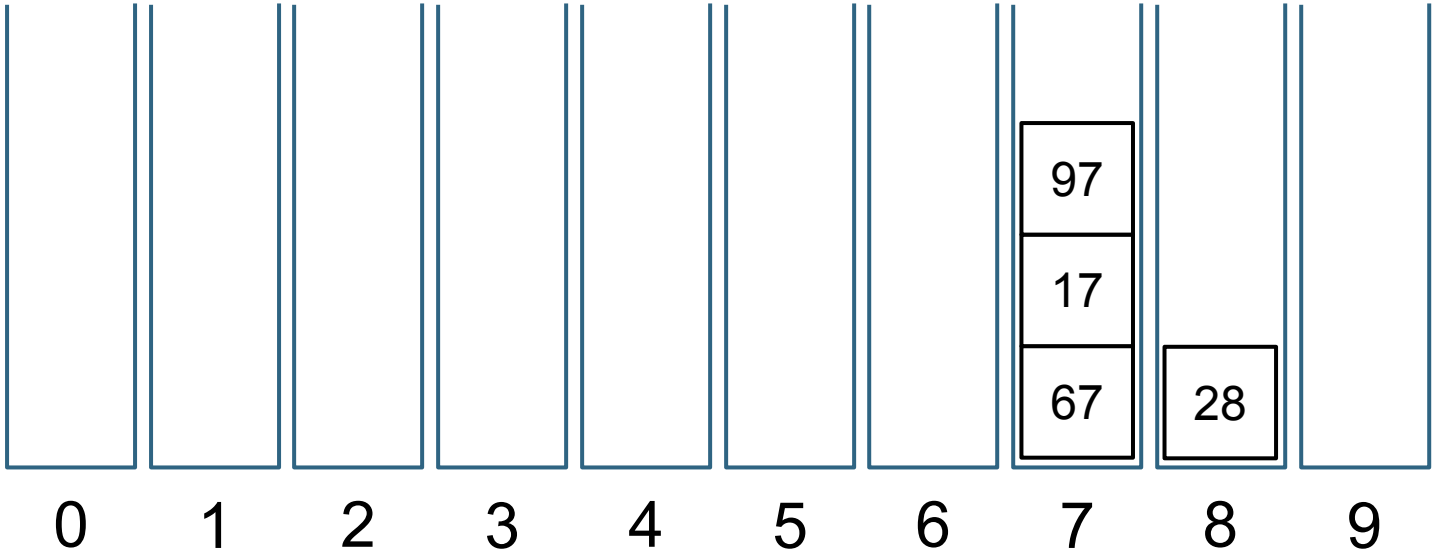
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25					

Radix Sort Example

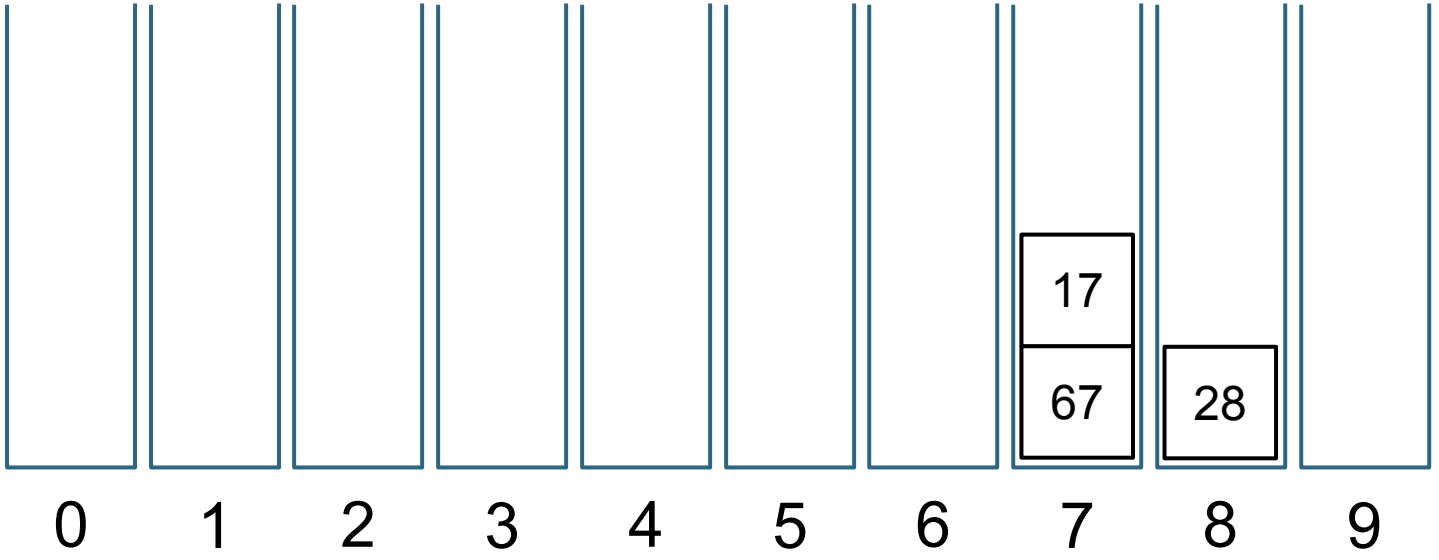
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25	27				

Radix Sort Example

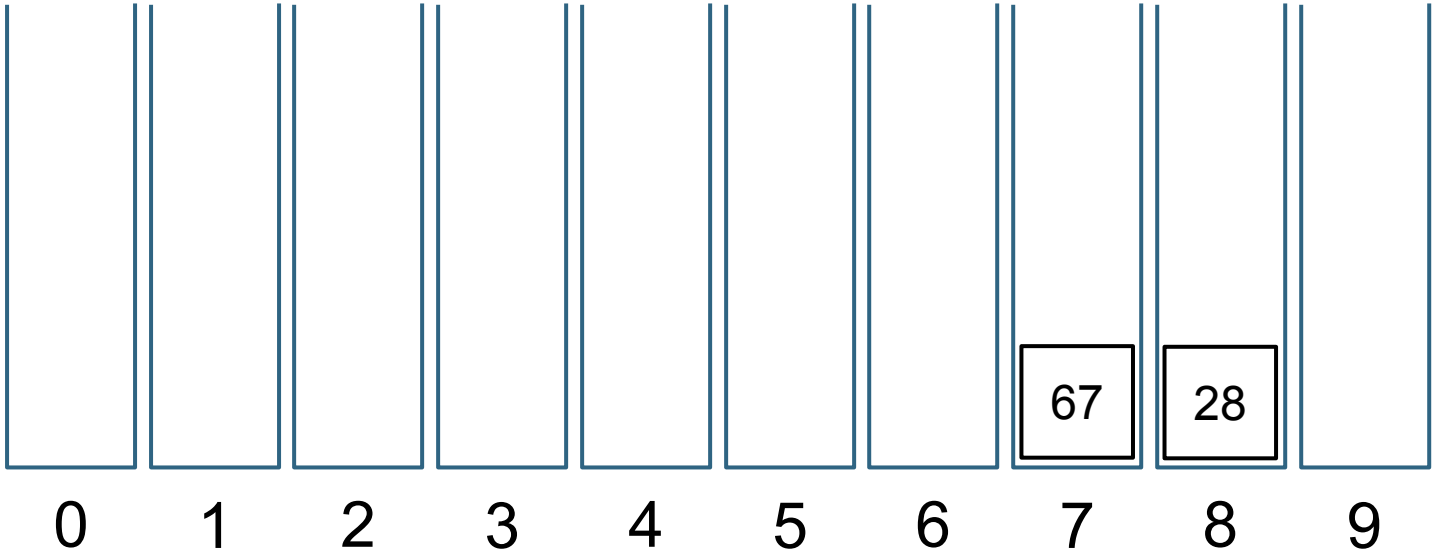
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25	27	97			

Radix Sort Example

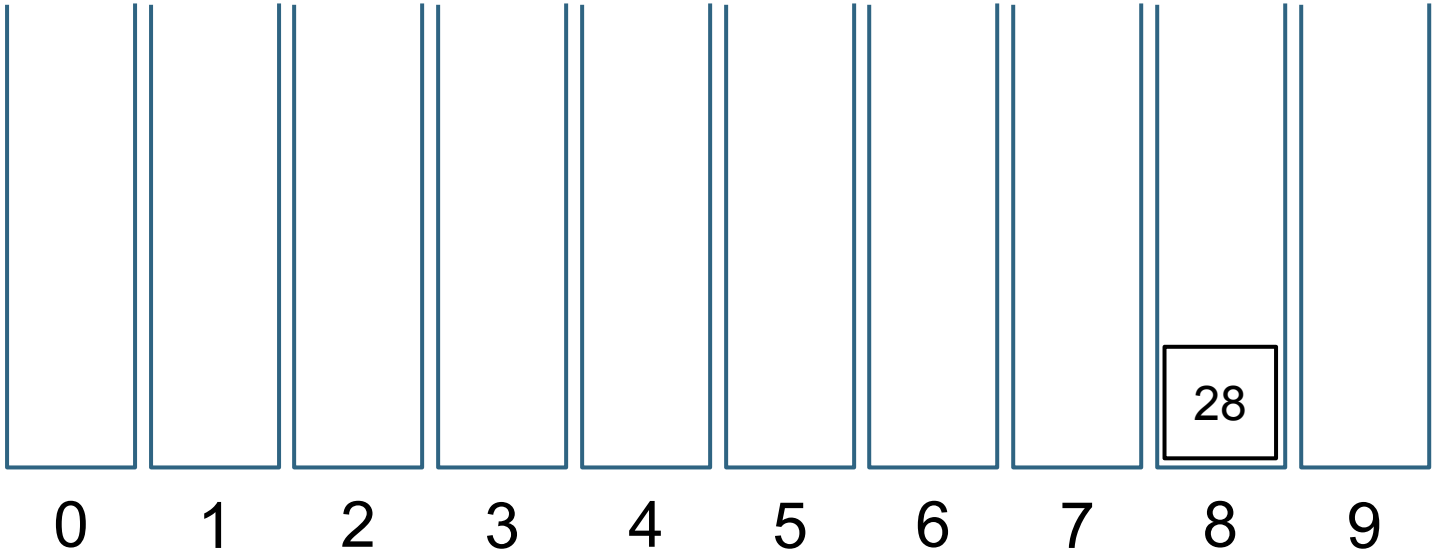
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25	27	97	17		

Radix Sort Example

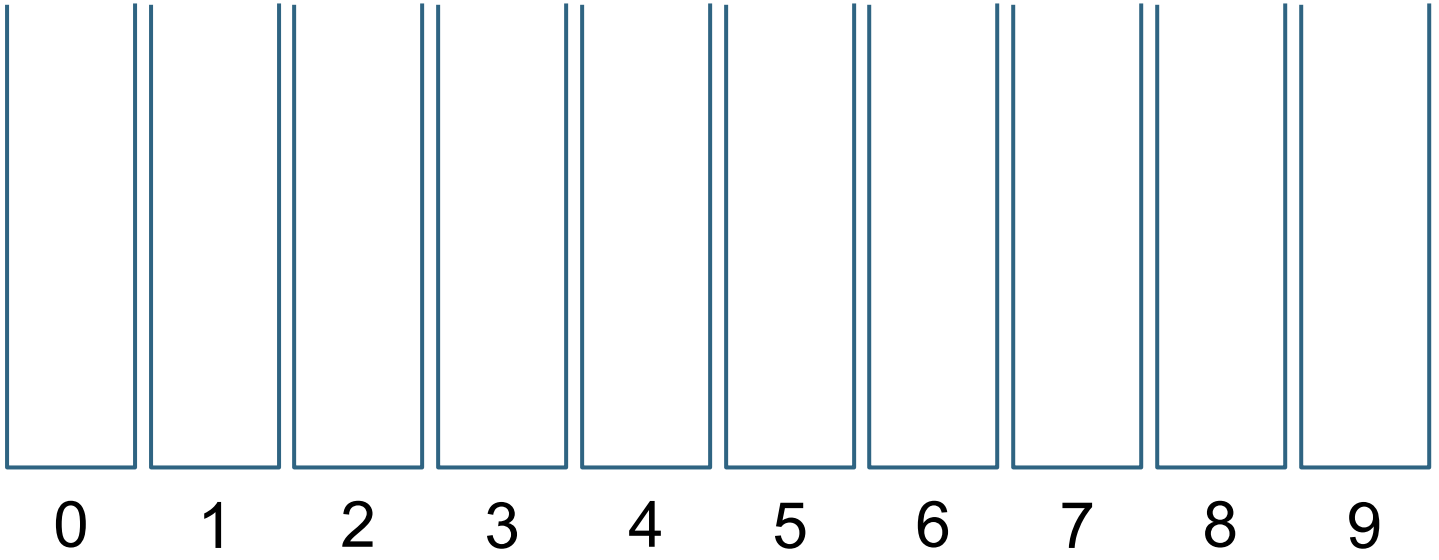
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25	27	97	17	67	

Radix Sort Example

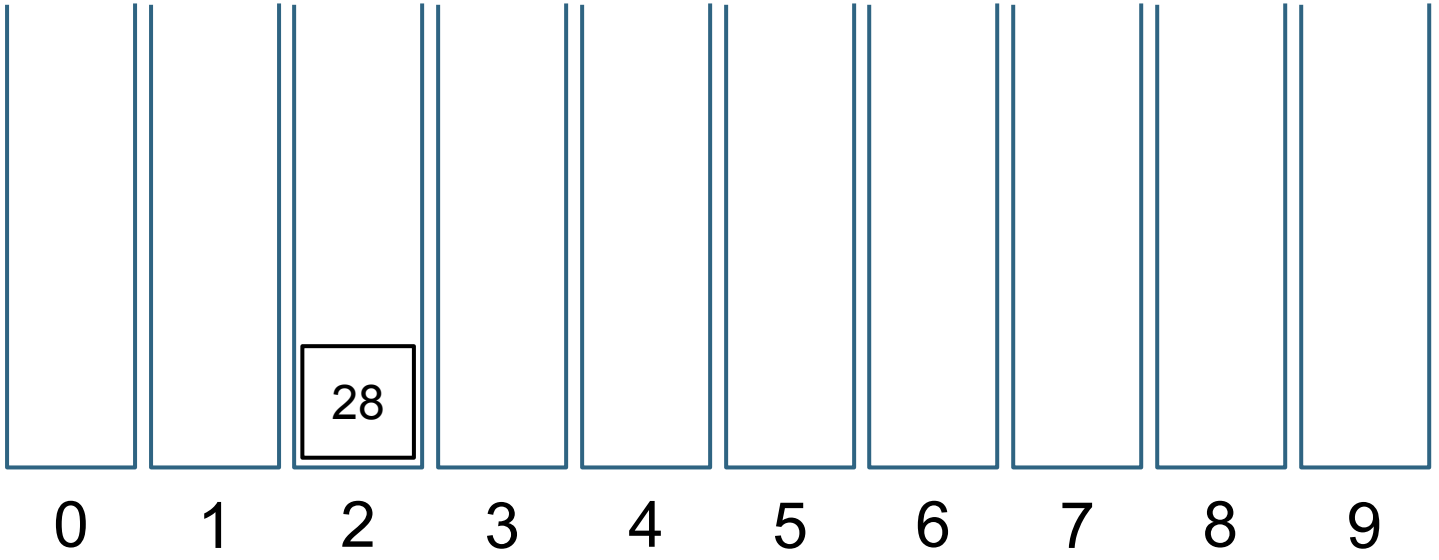
- *First pass (on the 1st digit from right)*
  - *Put the elements back into the array*



91	1	72	23	84	5	25	27	97	17	67	28

Radix Sort Example

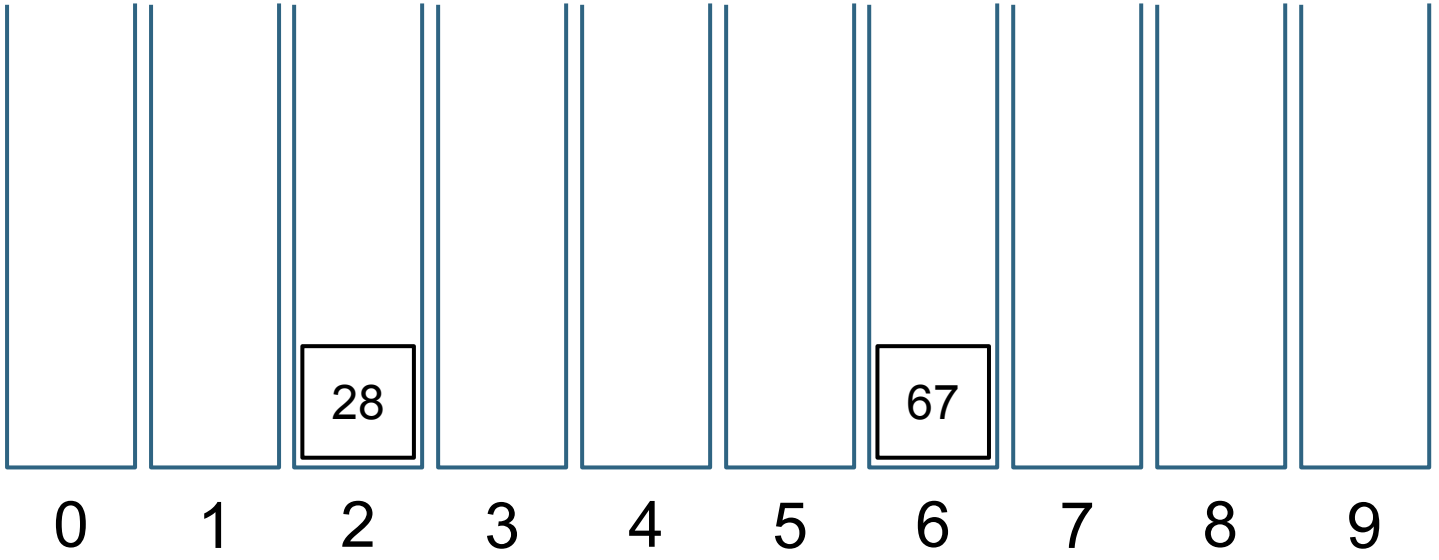
- *Second pass (on the 2nd digit from right)*
  - *Put the elements to bins*



91	1	72	23	84	5	25	27	97	17	67	

Radix Sort Example

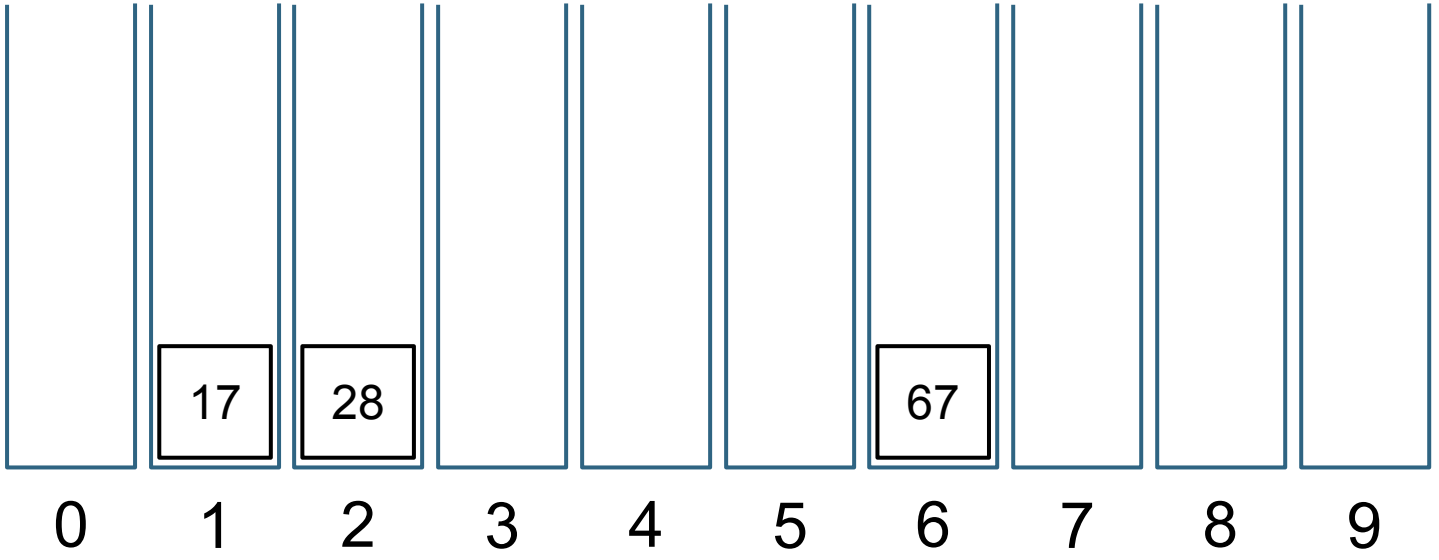
- *Second pass (on the 2nd digit from right)*
  - *Put the elements to bins*



91	1	72	23	84	5	25	27	97	17		

Radix Sort Example

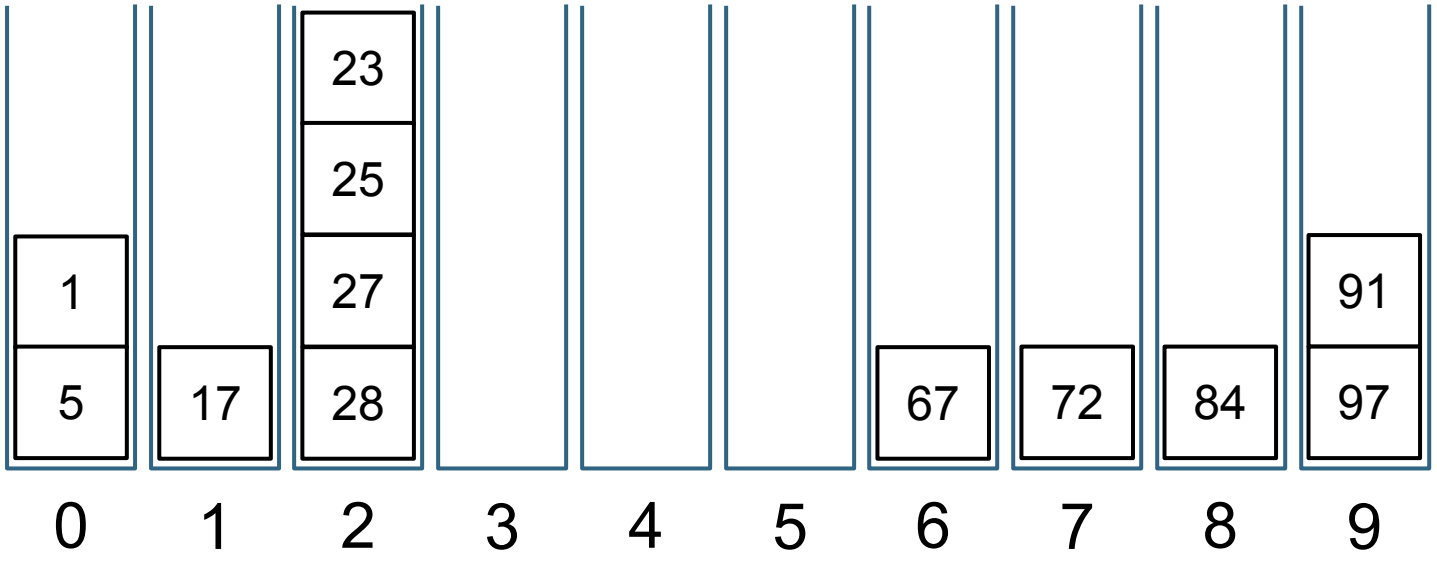
- *Second pass (on the 2nd digit from right)*
  - *Put the elements to bins*



91	1	72	23	84	5	25	27	97			

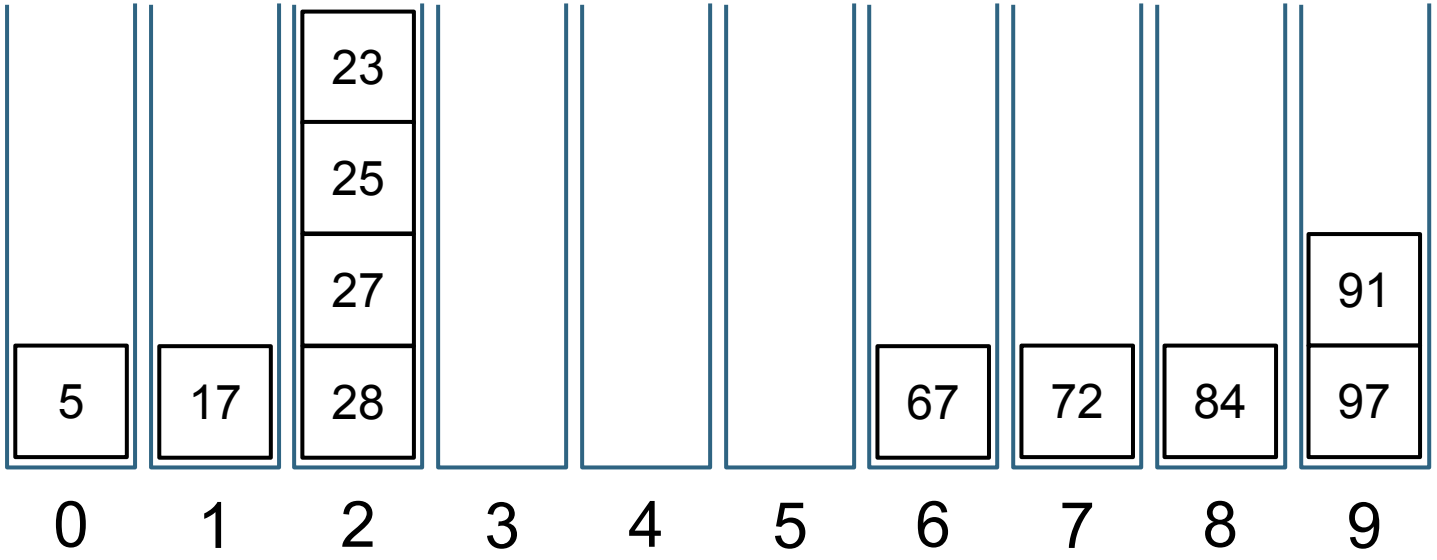
Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements to bins (some steps are skipped)*



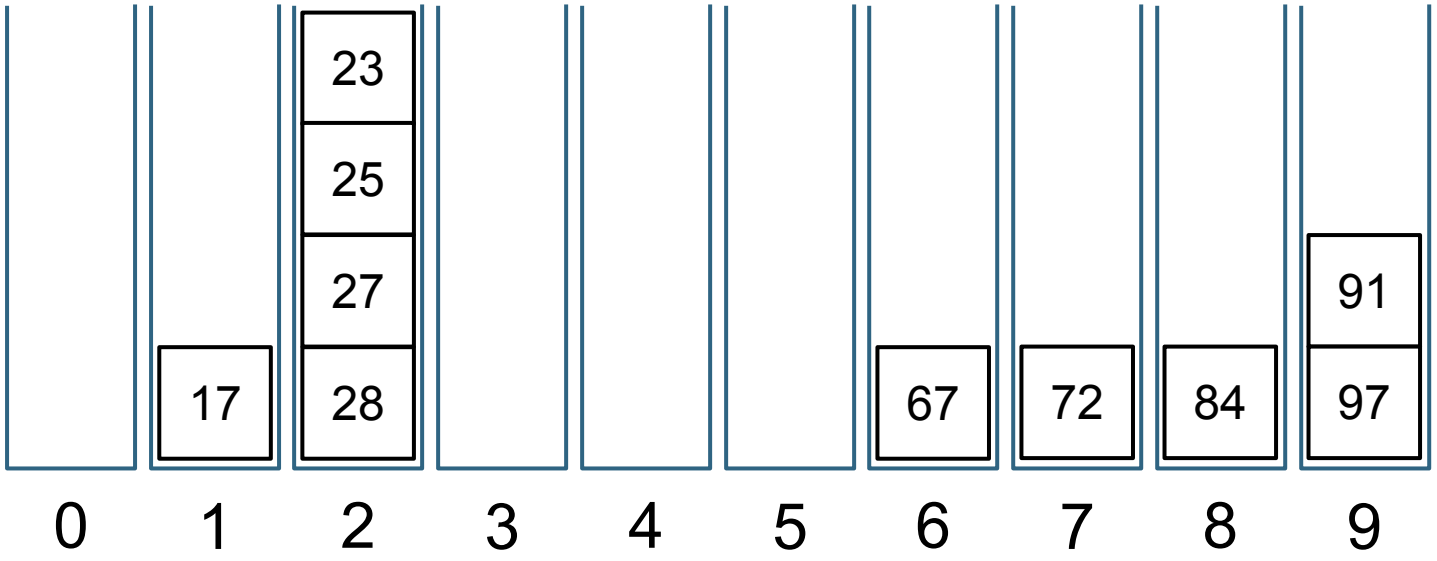
Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements back into the array*



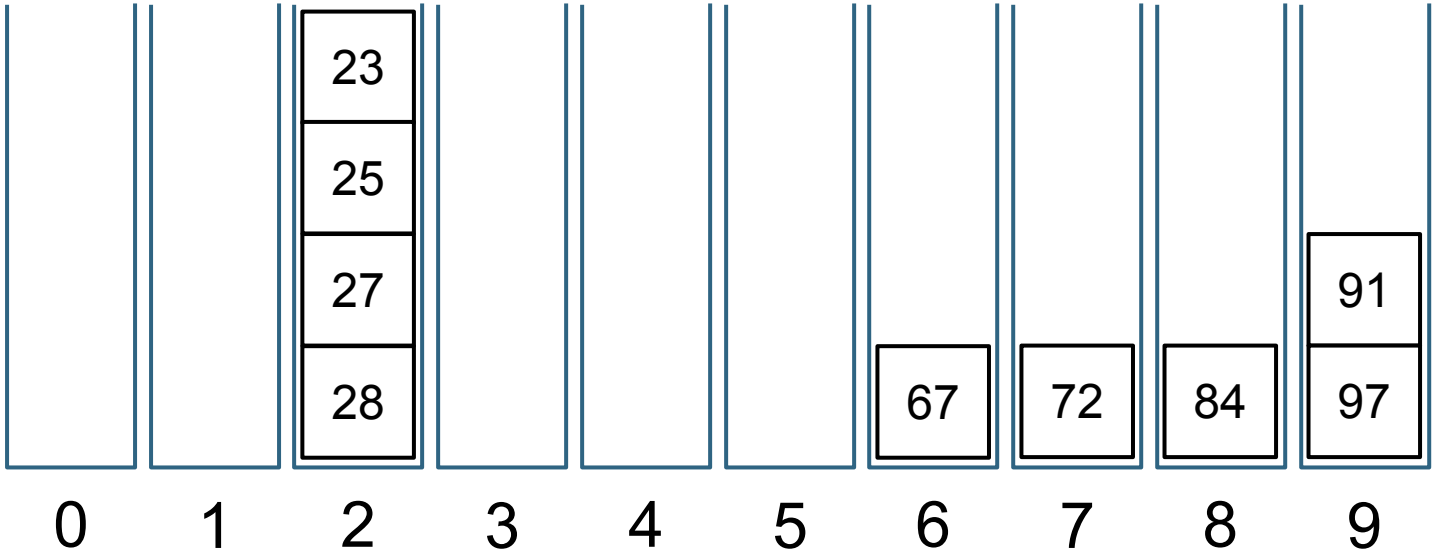
Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements back into the array*



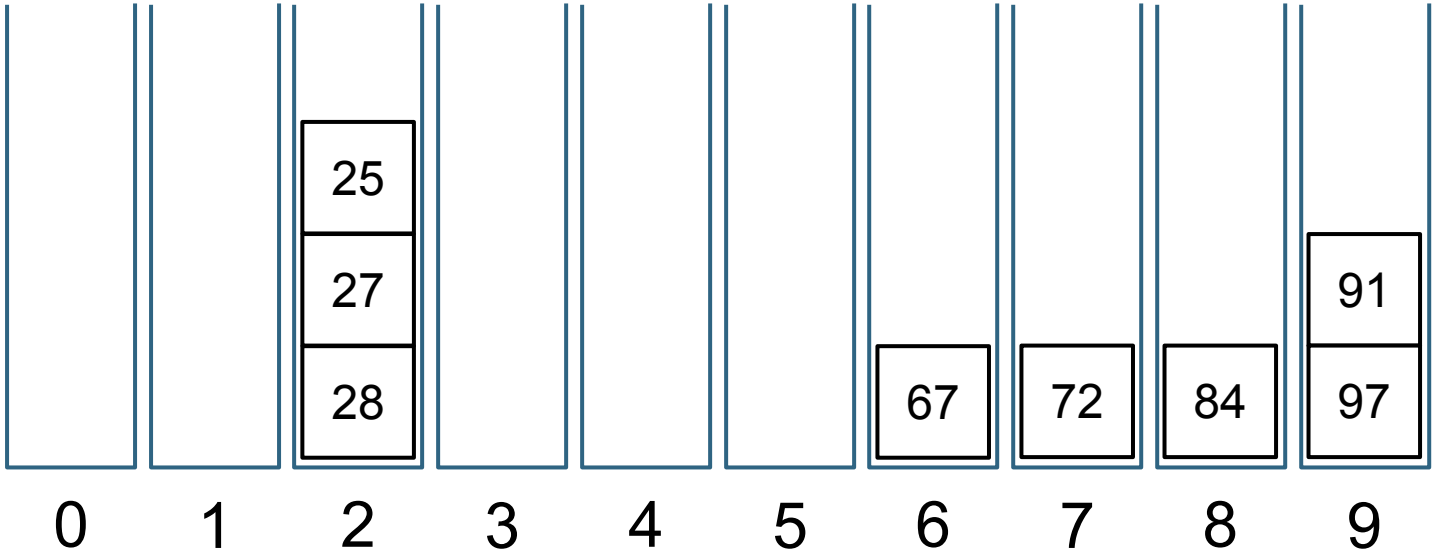
Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements back into the array*



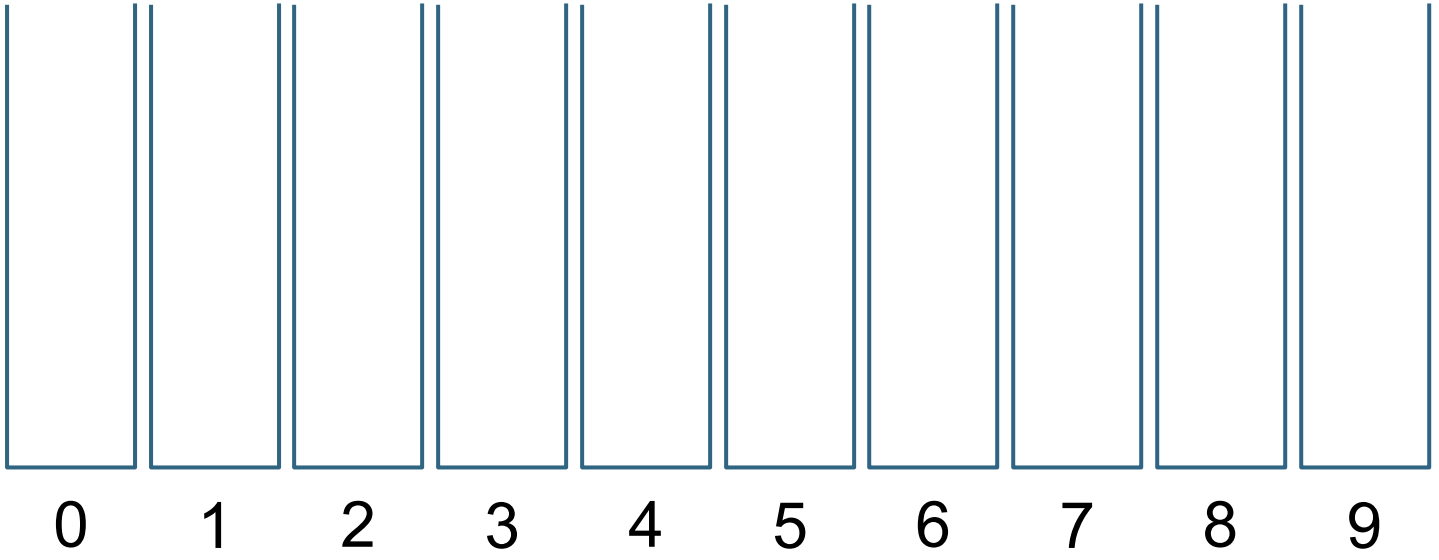
Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements back into the array*



Radix Sort Example

- *Second pass (on the 2nd digit from right)*
  - *Put the elements back into the array (some steps are skipped)*



1	5	17	23	25	27	28	67	72	84	91	97

Radix Sort Implementation

- 이 방법은 배열과 (linked) stack을 필요로 함
- radix sort를 크기가 n인 배열을 이용해서 구현 가능?
  - 추가 배열을 사용해도 된다고 하면…
- Yes!
  - Main idea: 2-step algorithm on the input data
    - Step 1: 각 bin에 몇 개의 item이 들어가는지 개수를 셈
    - Step 2: step 1 에서 센 것을 사용하여 각 item을 적절한 위치에 놓음

Radix Sort Implementation

Initial Input: Array A

27	91	1	97	17	23	84	28	72	5	67	25

First pass values for Count.
rtoi = 1.

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

Count array:
Index positions for Array B.

0	1	2	3	4	5	6	7	8	9
0	2	3	4	5	7	7	11	12	12

End of Pass 1: Array A.

91	1	72	23	84	5	25	27	97	17	67	28
0	1	2	3	4	5	6	7	8	9	10	11

Radix Sort Implementation

End of Pass 1: Array A.

91	1	72	23	84	5	25	27	97	17	67	28
0	1	2	3	4	5	6	7	8	9	10	11

Second pass values for Count.
rtoi = 10.

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

Count array:
Index positions for Array B.

0	1	2	3	4	5	6	7	8	9
2	3	7	7	7	7	8	9	10	12

End of Pass 2: Array A.

1	5	17	23	25	27	28	67	72	84	91	97
0	1	2	3	4	5	6	7	8	9	10	11

Radix Sort Implementation

```
static void radix(Integer[] A, Integer[] B,
                  int k, int r, int[] count) {
    int i, j, rtok;
    for (i = 0, rtok = 1; i < k; i++, rtok *= r) {
        for (j = 0; j < r; j++) count[j] = 0;

        // Count # of elements for each bin on this pass
        for (j = 0; j < A.length; j++) count[(A[j] / rtok) % r]++;

        // count[j] is index in B for last slot of j
        for (j = 1; j < r; j++) count[j] = count[j - 1] + count[j];

        for (j = A.length - 1; j >= 0; j--)
            B[--count[(A[j] / rtok) % r]] = A[j];

        for (j = 0; j < A.length; j++) A[j] = B[j];
    }
}
```

Radix Sort Cost

n: input size
k: length of each input key
r: radix

- Radix Sort의 시간복잡도?

- 모든 key들이 서로 다르다면, 키 길이 k의 최소값은?

Radix Sort Cost

n: input size
k: length of each input key
r: radix

- Radix Sort의 시간복잡도? $\Theta(nk + rk)$

- 모든 key들이 서로 다르다면, 키 길이 k의 최소값은?
  - 최소 $\log_r n$
  - 따라서, 일반적인 경우 radix sort는 $O(n \log n)$ 알고리즘임

Overview

- Heapsort
- Binsort
- Radix Sort

Questions?