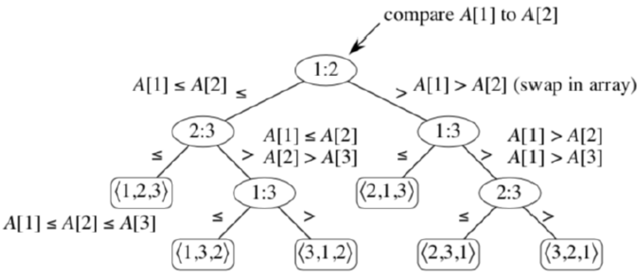
Sorting in Linear Time

- 모든 비교 정렬 알고리즘은 최악의 경우 $\Omega(n \lg n)$ 번의 비교 필요
- 정렬 알고리즘의 실행은 결정 트리의 루트에서 하나의 리프까지 경로를 따라가는 것

the decision-tree model for comparison sort



$$n! \le l \le 2^h$$

$$\begin{aligned} h &\ge \lg(n!) \\ &= \Omega(n \lg n) \end{aligned}$$

by eq.3.19 $\lg(n!) = \Theta(n \lg n)$

l : # of leaves
h : height of decision-tree = 최악의 경우 비교 횟수

Counting Sort

- 각 원소는 [0, k] 인 정수인 경우
- 각 원소에 대해 그보다 작은 원소의 갯수를 세면 정렬 후 원소의 위치를 알 수 있다

let $C[0..k]$ be a new array
for $i = 0$ **to** k
    $C[i] = 0$
for $j = 1$ **to** $A.length$
    $C[A[j]] = C[A[j]] + 1$

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

	0	1	2	3	4	5
C	2	0	2	3	0	1

for $i = 1$ **to** k
    $C[i] = C[i] + C[i-1]$

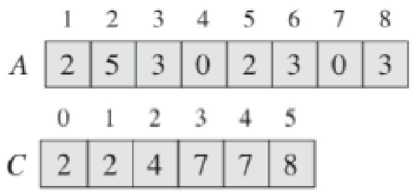
	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3

	0	1	2	3	4	5
C	2	0	2	3	0	1

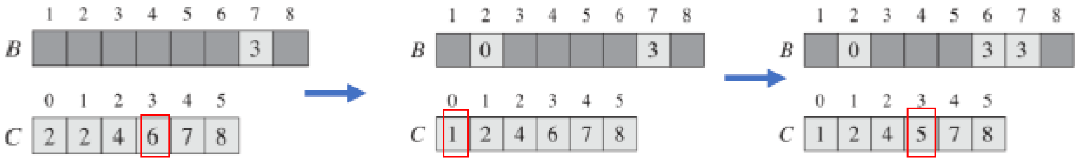
(a)

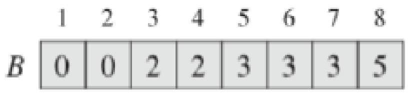
	0	1	2	3	4	5
C	2	2	4	7	7	8

(b)




```
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```





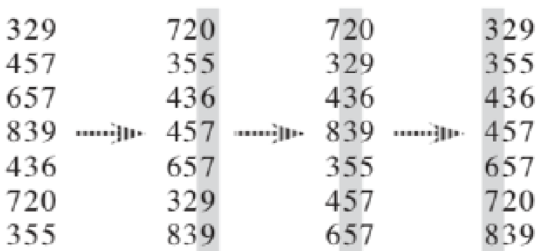
- Stable sort
  - 출력 배열에서 값이 같은 숫자가 입력 배열에 있던 것과 같은 순서를 유지하는 정렬
  - running time : $\Theta(k+n)$ if $k = O(n)$, $\Theta(n)$

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j = 1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
    C[i] = C[i] + C[i - 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

Radix Sort

- **가장 낮은 자리 숫자부터** 정렬하는 것을 자리수만큼 반복

```
RADIX-SORT(A, d)
for i = 1 to d
    use a stable sort to sort array A on digit i
```



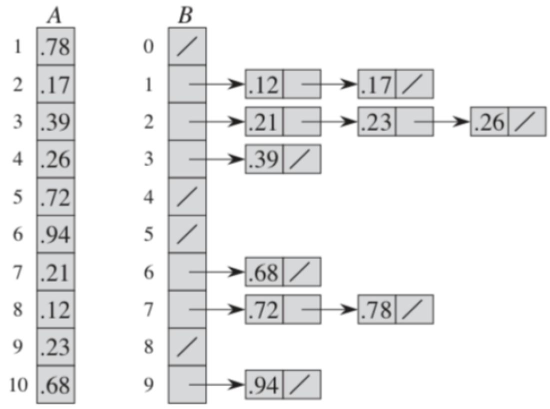
  - 각각의 정렬은 stable 해야 함, counting sort 사용
  - running time : $\Theta(d(n+k))$ total If $k = O(n)$, time $= \Theta(dn)$
- 높은 자리부터 정렬하면

34		17		31
17	⇨	25	⇨	34
25		34		25
31		31		17

Bucket Sort

- 입력이 uniformly distributed in (0,1)인 경우

A.length = B.length 라는 것이 복잡도 계산의 핵심




```
BUCKET-SORT(A)
let B[0..n-1] be a new array
n = A.length
for i = 0 to n - 1
    make B[i] an empty list
for i = 1 to n
    insert A[i] into list B[⌊nA[i]⌋]
for i = 0 to n - 1
    sort list B[i] with insertion sort
concatenate the lists B[0], B[1], ..., B[n-1] together in order
```

- Time Complexity
  - n_i : # of elements in B[i]
  - insertion sort on runs in $O(n^2)$

→ $$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2) .$$

  - average case running time

$$\begin{aligned} E[T(n)] &= E\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right] \\ &= \Theta(n) + \sum_{i=0}^{n-1} E\left[O(n_i^2)\right] \\ &= \Theta(n) + \sum_{i=0}^{n-1} O\left(E\left[n_i^2\right]\right) \end{aligned}$$

$$\begin{aligned} E[T(n)] &= \Theta(n) + \sum_{i=0}^{n-1} O(2 - 1/n) \\ &= \Theta(n) + O(n) \\ &= \Theta(n) \end{aligned}$$

  - $E\left[n_i^2\right] = 2 - 1/n$ 이므로

$E[n_i^2] = 2 - 1/n$ in bucket sort

$X_{ij} = I\{A[j] \text{ falls in bucket } i\}$

for $i = 0, 1, \ldots, n-1$ and $j = 1, 2, \ldots, n$. Thus,

$$n_i = \sum_{j=1}^{n} X_{ij} .$$

$$\begin{aligned}
E[n_i^2] &= E\left[\left(\sum_{j=1}^{n} X_{ij}\right)^2\right] \\
&= E\left[\sum_{j=1}^{n}\sum_{k=1}^{n} X_{ij}X_{ik}\right] \\
&= E\left[\sum_{j=1}^{n} X_{ij}^2 + \sum_{1\le j\le n}\ \sum_{\substack{1\le k\le n \\ k\ne j}} X_{ij}X_{ik}\right] \\
&= \sum_{j=1}^{n} E[X_{ij}^2] + \sum_{1\le j\le n}\ \sum_{\substack{1\le k\le n \\ k\ne j}} E[X_{ij}X_{ik}]
\end{aligned}$$

$$\begin{aligned}
E[X_{ij}^2] &= 1^2 \cdot \frac{1}{n} + 0^2 \cdot \left(1 - \frac{1}{n}\right) \\
&= \frac{1}{n} .
\end{aligned}$$

When $k \ne j$, the variables X_{ij} and X_{ik} are independent

$$\begin{aligned}
E[X_{ij}X_{ik}] &= E[X_{ij}]\,E[X_{ik}] \\
&= \frac{1}{n} \cdot \frac{1}{n} \\
&= \frac{1}{n^2} .
\end{aligned}$$

$$\begin{aligned}
E[n_i^2] &= \sum_{j=1}^{n} \frac{1}{n} + \sum_{1\le j\le n}\ \sum_{\substack{1\le k\le n \\ k\ne j}} \frac{1}{n^2} \\
&= n \cdot \frac{1}{n} + n(n-1) \cdot \frac{1}{n^2} \\
&= 1 + \frac{n-1}{n} \\
&= 2 - \frac{1}{n} ,
\end{aligned}$$

- Worst-case running time of bucket sort

n_i : # of elements in B[i]

insertion sort on runs in $O(n^2)$

→ $$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2) .$$

worst case running time $\Theta(n^2)$ when ?

```
BUCKET-SORT(A)
let B[0..n-1] be a new array
n = A.length
for i = 0 to n-1
    make B[i] an empty list
for i = 1 to n
    insert A[i] into list B[⌊nA[i]⌋]
for i = 0 to n-1
    sort list B[i] with insertion sort
concatenate the lists B[0], B[1], ..., B[n-1] together in order
```

- Running times of sorting algorithms

Algorithm	Worst-case running time	Average-case/expected running time
Insertion sort	$\Theta(n^2)$	$\Theta(n^2)$
Merge sort	$\Theta(n \lg n)$	$\Theta(n \lg n)$
Heapsort	$O(n \lg n)$	—
Quicksort	$\Theta(n^2)$	$\Theta(n \lg n)$ (expected)
Counting sort	$\Theta(k + n)$	$\Theta(k + n)$
Radix sort	$\Theta(d(n + k))$	$\Theta(d(n + k))$
Bucket sort	$\Theta(n^2)$	$\Theta(n)$ (average-case)