Radix Sort

Prepared by Suk Jin Lee

Radix Sort

- *How did IBM get rich originally?*
  - Answer: punched card readers for census tabulation in early 1900's.
  - In particular, a *card sorter* that could sort cards into different bins
    - Each column can be punched in 12 places
    - 10 places for decimal digits and 2 places for nonnumeric char
  - Problem: only one column can be sorted on at a time

Radix Sort

- Based on examining digits in some base-*b* numeric representation of items (or keys)
- *Key idea*: Sort *least* signifiant digit first
  - Processes digits from right to left
  - Used in early punched-card sorting machines
- RADIX-SORT(A, d)

    for i = 1 to d

        use a stable sort to sort array A on digit i

Operation of Radix sort

64, 8, 216, 11, 512, 27, 729, 199, 550, 343, 125, 93, 666

Write them all with three digits, padding with 0s

064, 008, 216, 011, 512, 027, 729, 199, 550, 343, 125, 093, 666

Distribute them into 10 bits labeled 0, 1,…, 9

			093			666			199
550	011	512	343	064	125	216	027	008	729
0	1	2	3	4	5	6	7	8	9

Collect them together from left to right (→), bottom to top (↑)

550, 011, 512, 343, 093, 064, 125, 216, 666, 027, 008, 729, 199

Operation of Radix sort

550, 011, 512, 343, 093, 064, 125, 216, 666, 027, 008, 729, 199

Distribute them again, using the second digit:

	216	729							
	512	027				666			199
008	011	125		343	550	064			093
0	1	2	3	4	5	6	7	8	9

Collect them together (→, ↑)

008, 011, 512, 216, 125, 027, 729, 343, 550, 064, 666, 093, 199

Distribute them, using the leftmost digit:

093									
064									
027									
011	199				550				
008	125	216	343		512	666	729		
0	1	2	3	4	5	6	7	8	9

Collecting them produces the sorted list:

008, 011, 027, 064, 093, 125, 199, 216, 343, 512, 550, 666, 729

Correctness of Radix sort

- Induction on digit position
  - Assume that lower-order digits $1, 2, \ldots, i - 1$ are sorted
  - Show that sorting next digit i leaves array correctly sorted
    - If two digits at position i are different, ordering numbers by that digit is correct (lower-order digits irrelevant)
    - If two digits are the same, numbers are already sorted on the lower-order digits. The numbers stay in the right order

Analysis

- Counting sort
  - Sort n numbers on digits that range from $0,\ldots, k$
  - Time: $O(n + k)$
- Assume that we use counting sort as the intermediate sort
  - $\Theta(n + k)$ per pass (digits in range $0,\ldots, k$)
  - d passes
  - $\Theta(d(n + k))$ total
  - If $k = O(n)$, time $= \Theta(dn)$
  - When d is constant and $k = O(n)$, , takes $\Theta(n)$

Radix Sort

- In general, radix sort based on counting sort is
  - Fast
  - Asymptotically fast (i.e., $O(n)$)
  - Simple to code
  - A good choice
  - Doesn't sort in place

Bucket Sort

Prepared by Suk Jin Lee

Bucket Sort

- Assumes the input is generated by a random process that distributes elements uniformly over [0, 1).
- ***Idea***
  - Divide [0, 1) into n equal-size *buckets*
  - Distribute the n input values into the buckets
  - Sort each bucket
  - Then go through buckets in order, listing elements in each one

Input: $A[1,\ldots, n]$, where $0 \le A[i] < 1$ for all i

Auxiliary array: $B[0,\ldots, n-1]$ of linked lists, each list initially empty

Bucket Sort

- BUCKET-SORT(A)

$n = A.length$
Let $B[0, \ldots, n-1]$ be a new array
for $i = 0$ to $n - 1$
  make $B[i]$ an empty list
for $i = 1$ to n
  insert $A[i]$ into list $B[\lfloor [n \cdot A[i] \rfloor]$
for $i = 0$ to $n - 1$
  sort list $B[i]$ with insertion sort
Concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

Bucket Sort

- BUCKET-SORT(A)

1. $n = A.length$
2. Let $B[0,\ldots, n-1]$ be a new array
3. **for** $i = 0$ to $n - 1$
4.   make $B[i]$ an empty list
5. **for** $i = 1$ to n
6.   insert $A[i]$ into list $B[\lfloor [n \cdot A[i] \rfloor]$
7. **for** $i = 0$ to $n - 1$
8.   sort list $B[i]$ with insertion sort
9. Concatenate the lists $B[0]$, $B[1]$,…, $B[n - 1]$ together in order

If $A[i]$ and $A[j]$ go into the same bucket, then the **for** loop of lines 7 – 8 puts them into the proper order.

If $A[i]$ and $A[j]$ go into different buckets, then line 9 puts them into the proper order.

Operation of Bucket-Sort

Input array: A

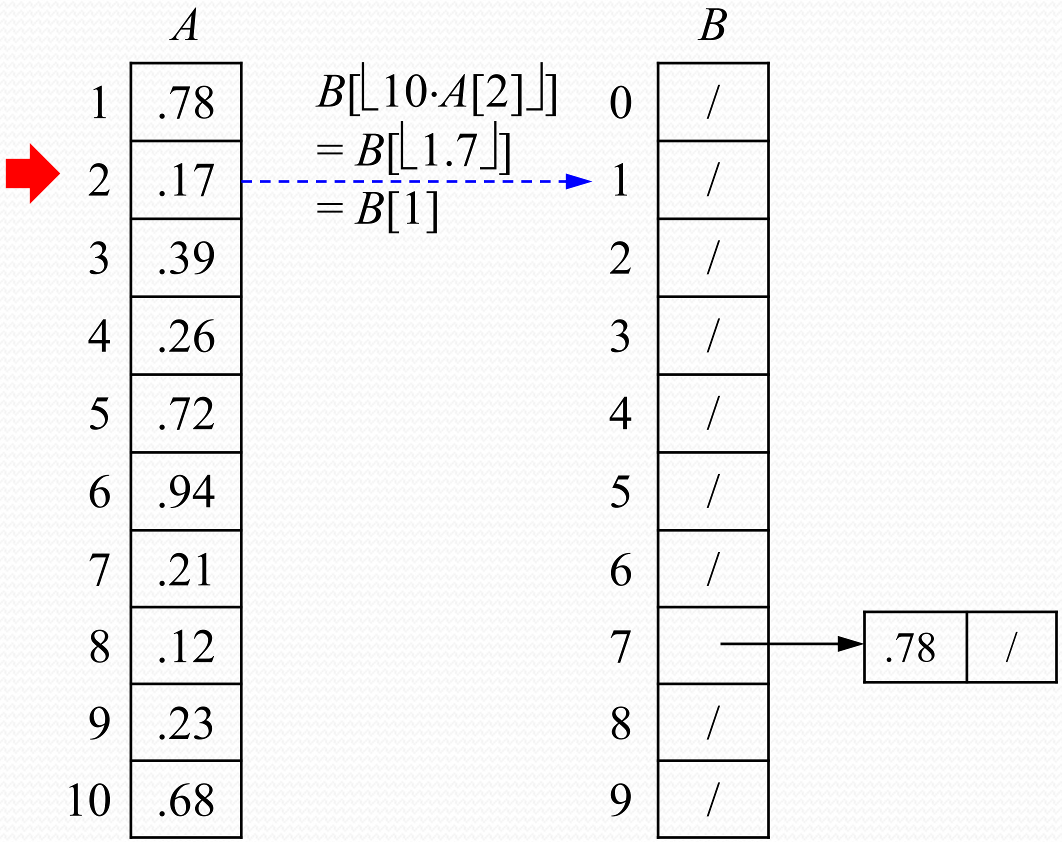
	A
1	.78
2	.17
3	.39
4	.26
5	.72
6	.94
7	.21
8	.12
9	.23
10	.68

$$B[\lfloor 10 \cdot A[1] \rfloor] = B[\lfloor 7.8 \rfloor] = B[7]$$

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

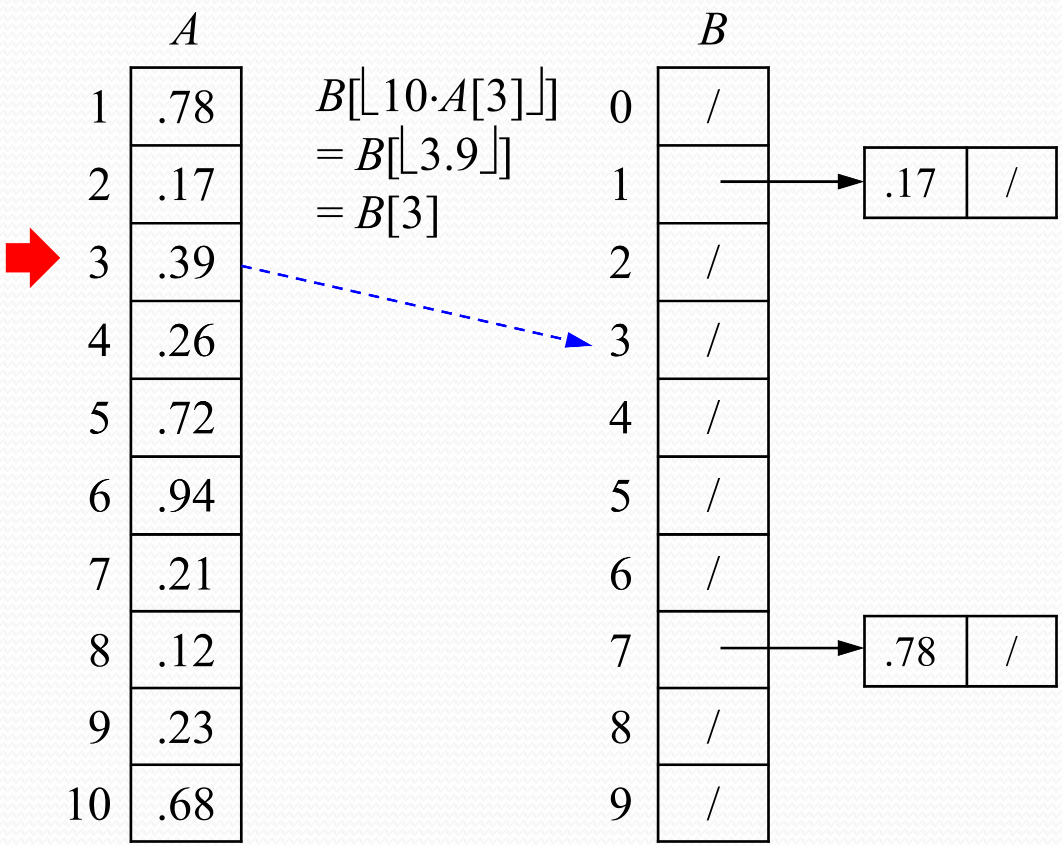
Operation of Bucket-Sort

Input array: *A*



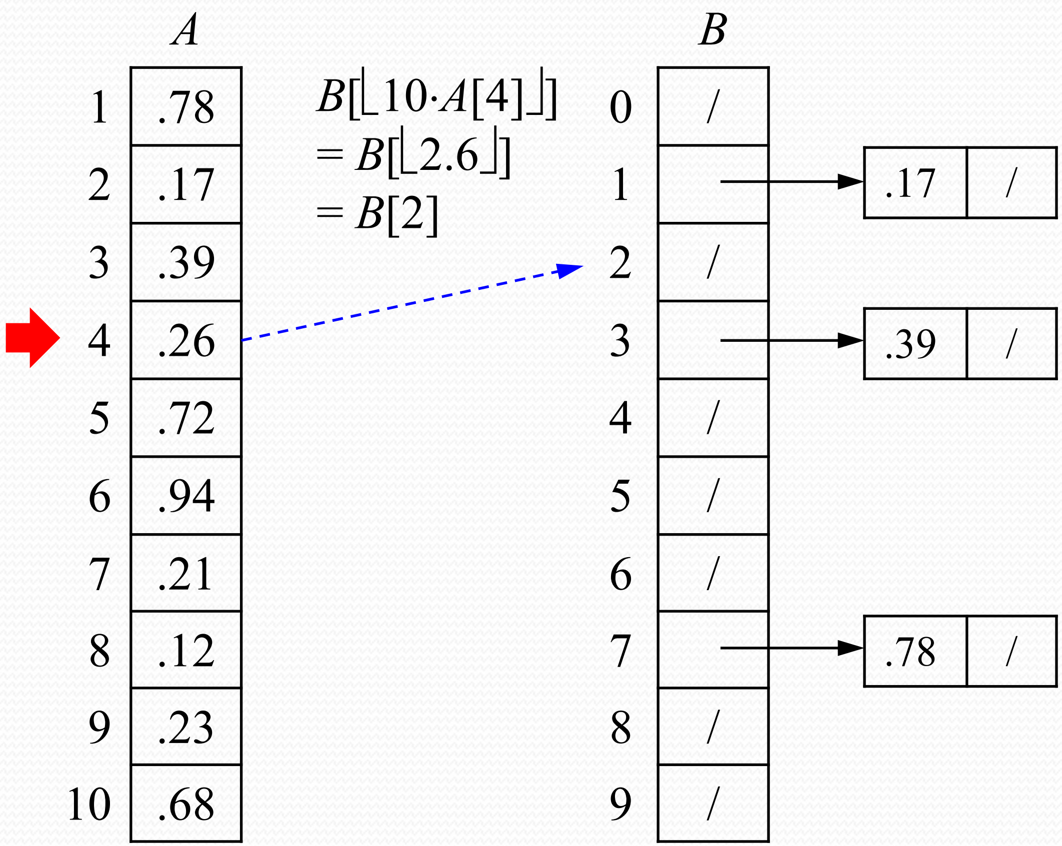
Operation of Bucket-Sort

Input array: *A*



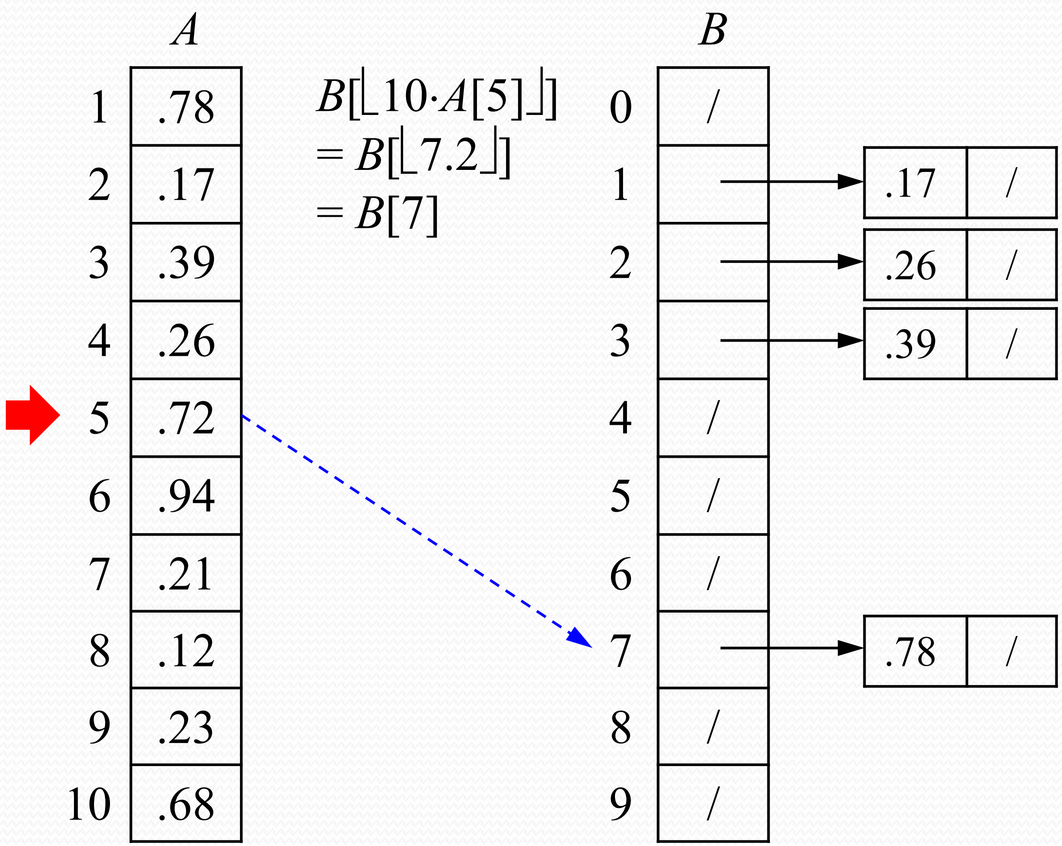
Operation of Bucket-Sort

Input array: A



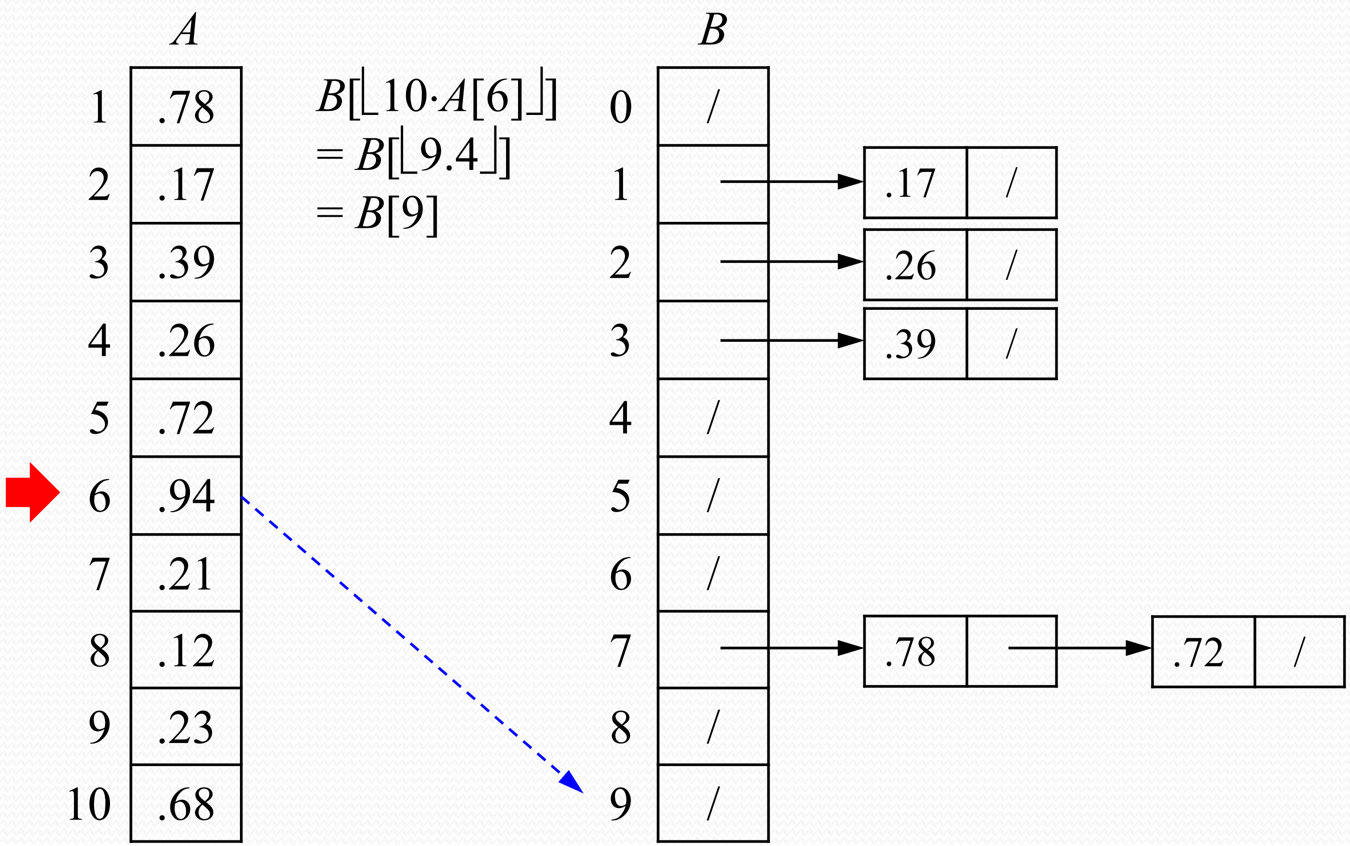
Operation of Bucket-Sort

Input array: A



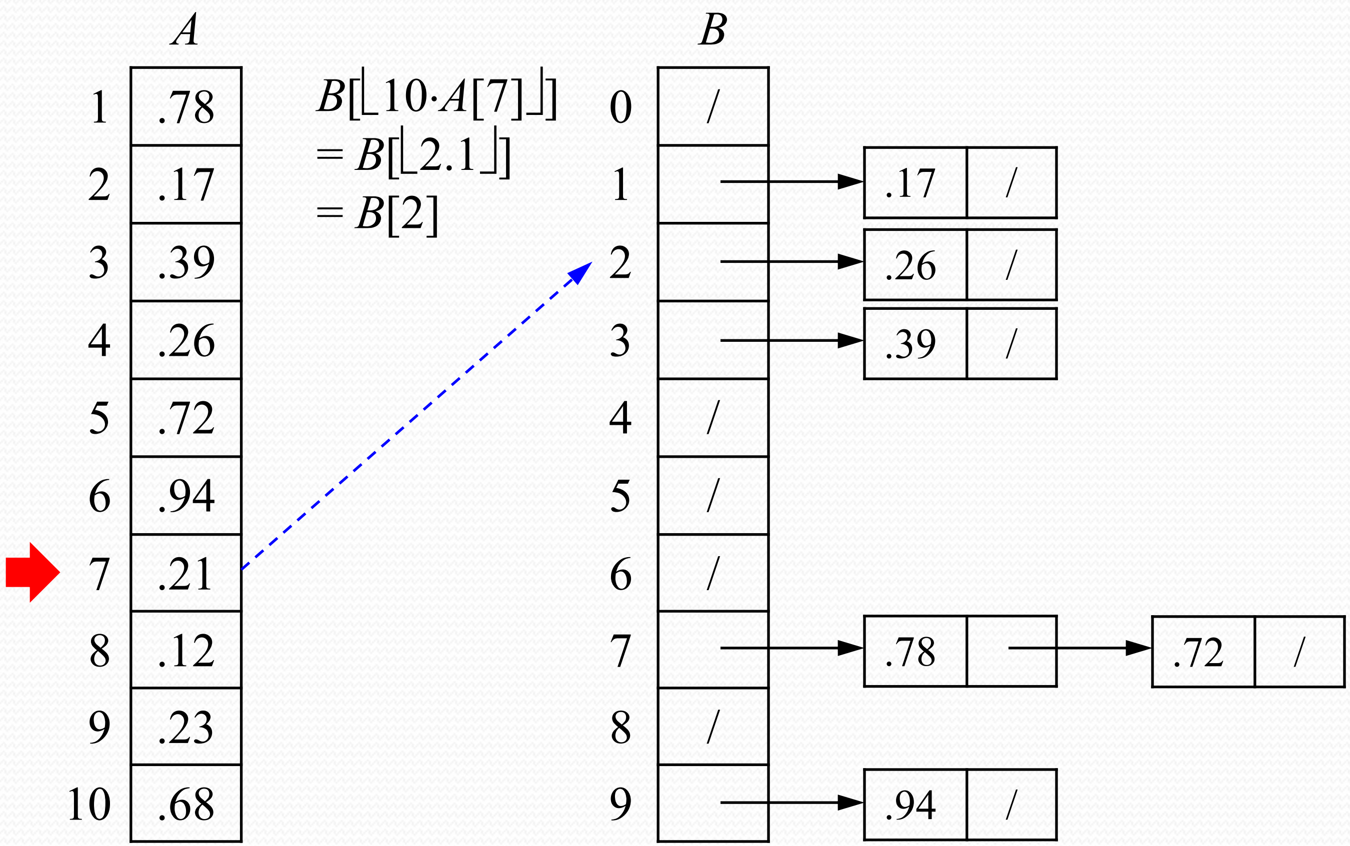
Operation of Bucket-Sort

Input array: A



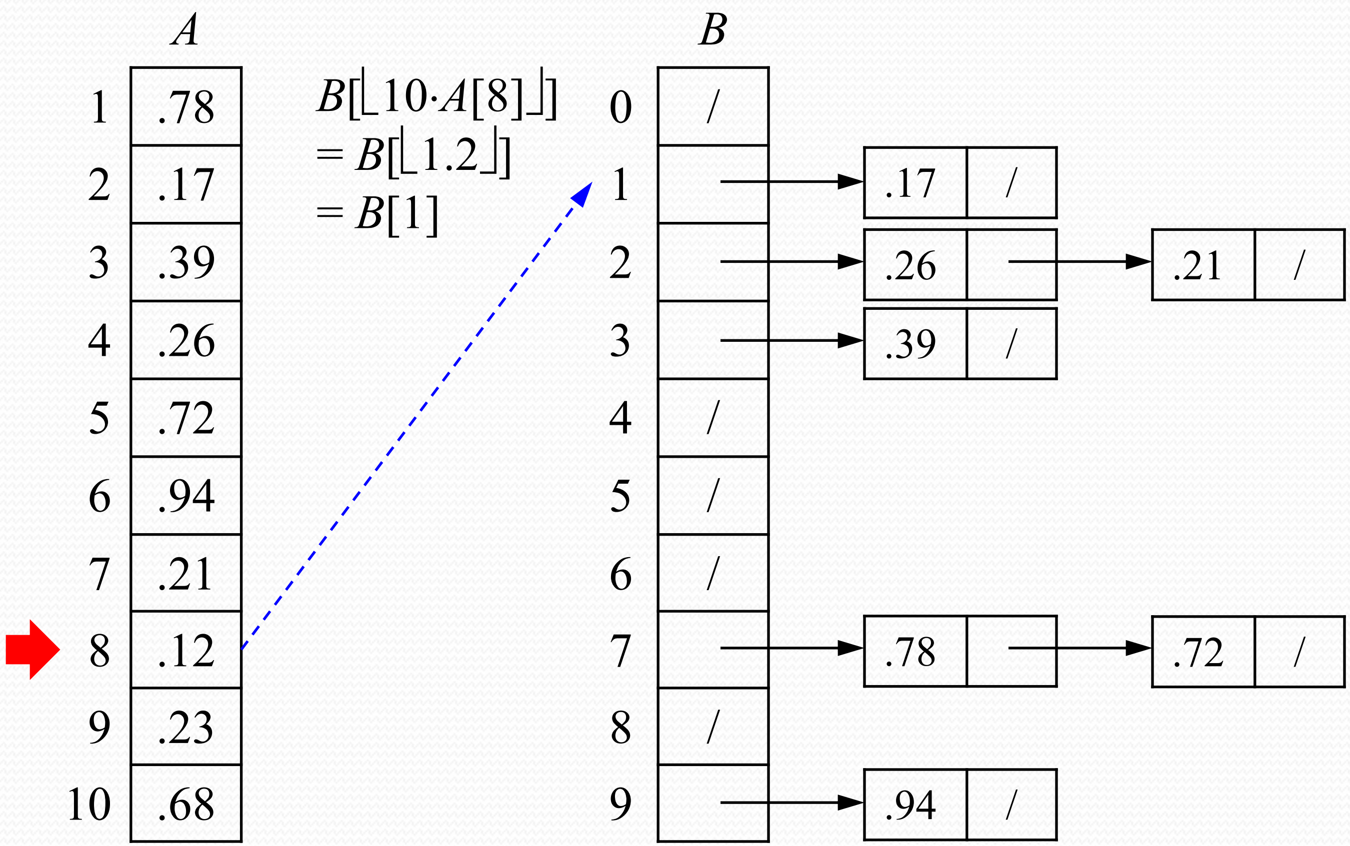
Operation of Bucket-Sort

Input array: *A*



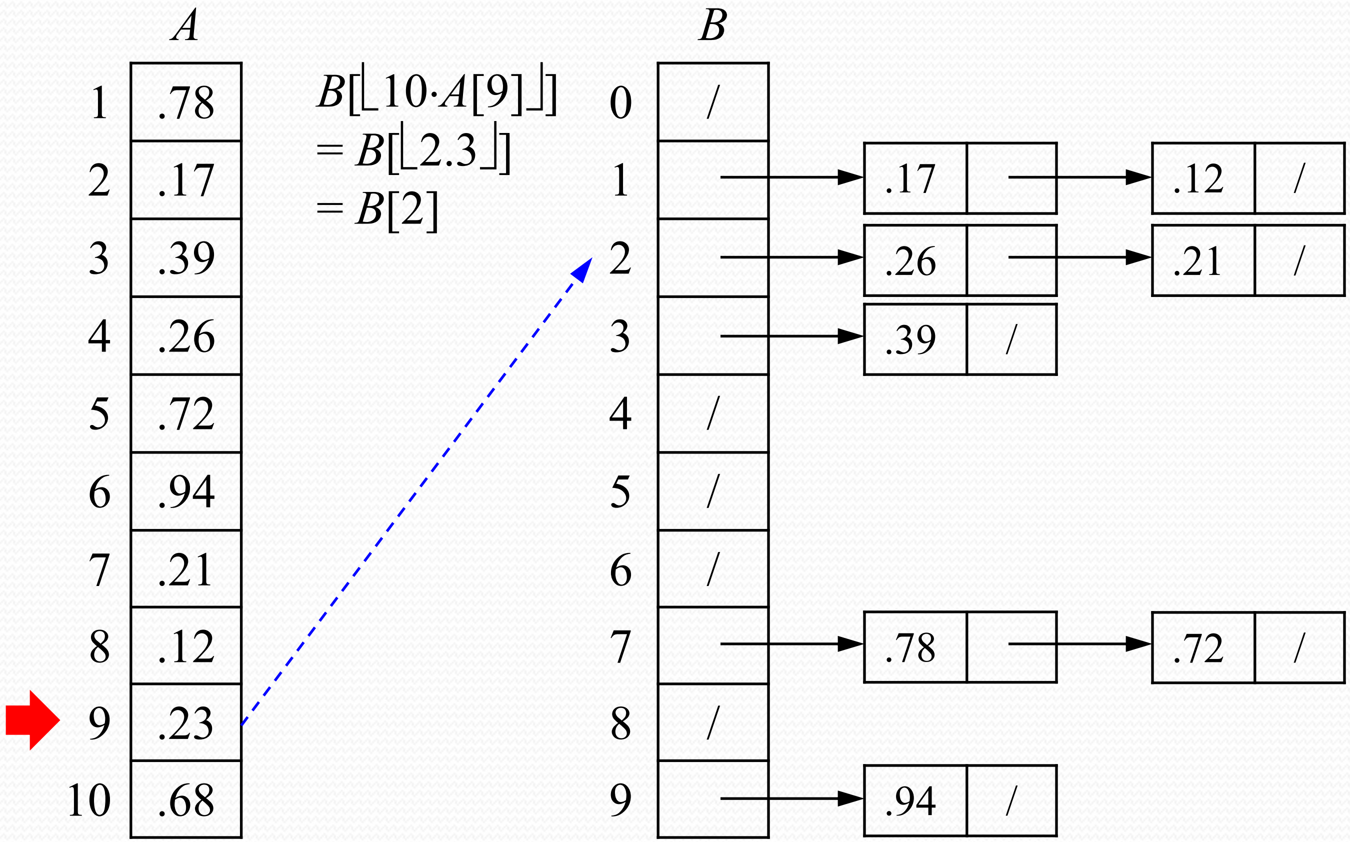
Operation of Bucket-Sort

Input array: A



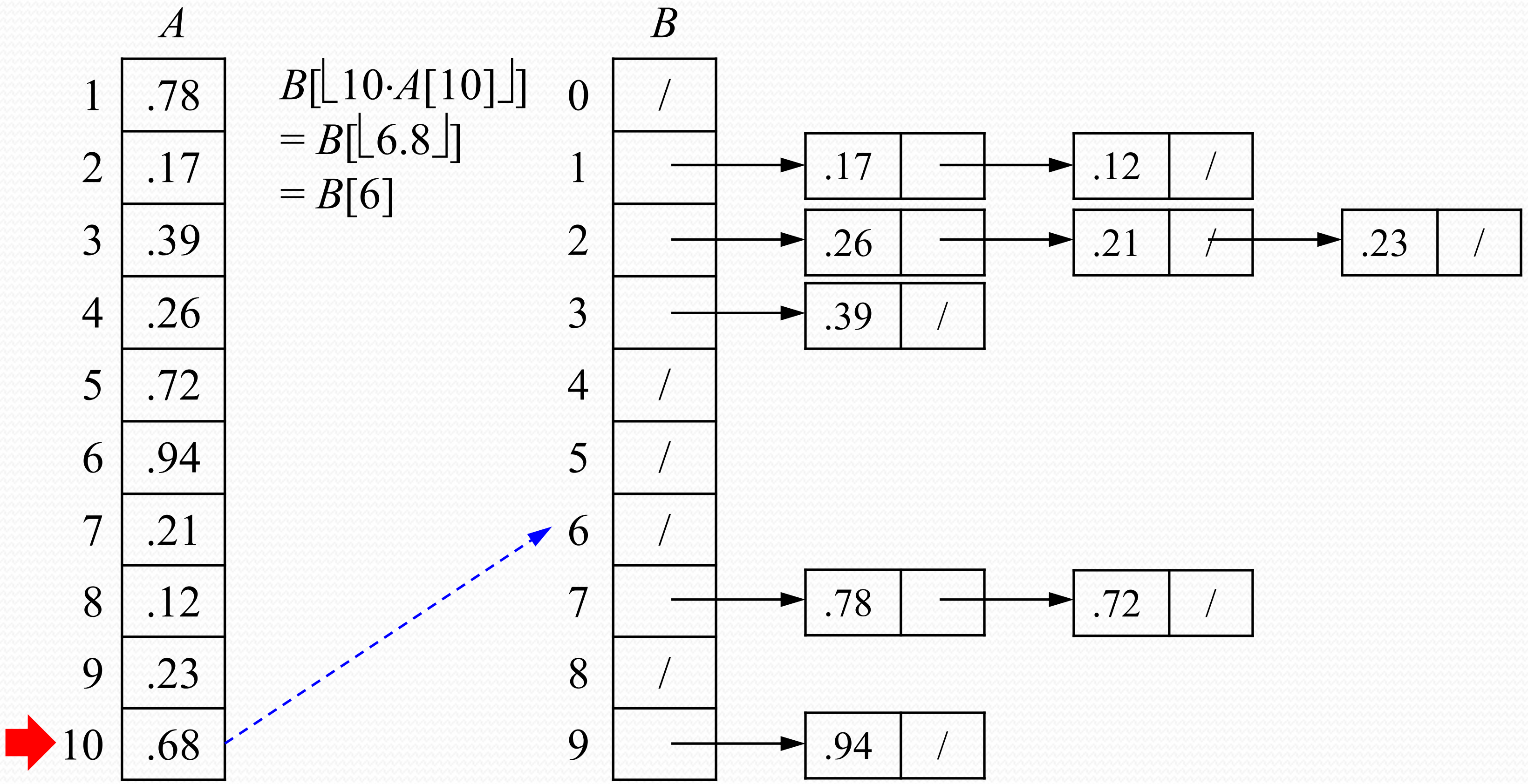
Operation of Bucket-Sort

Input array: *A*



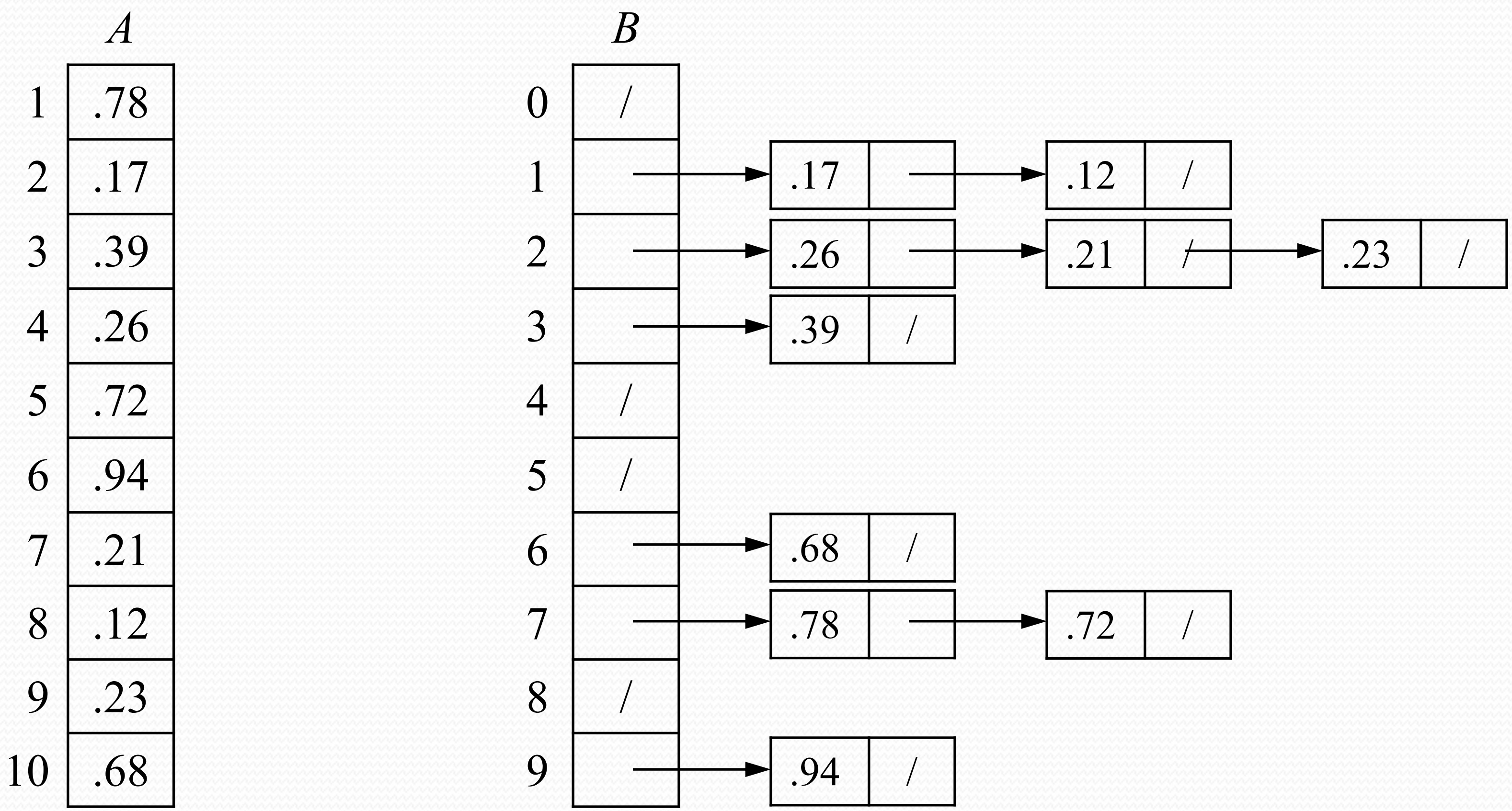
Operation of Bucket-Sort

Input array: A



Operation of Bucket-Sort

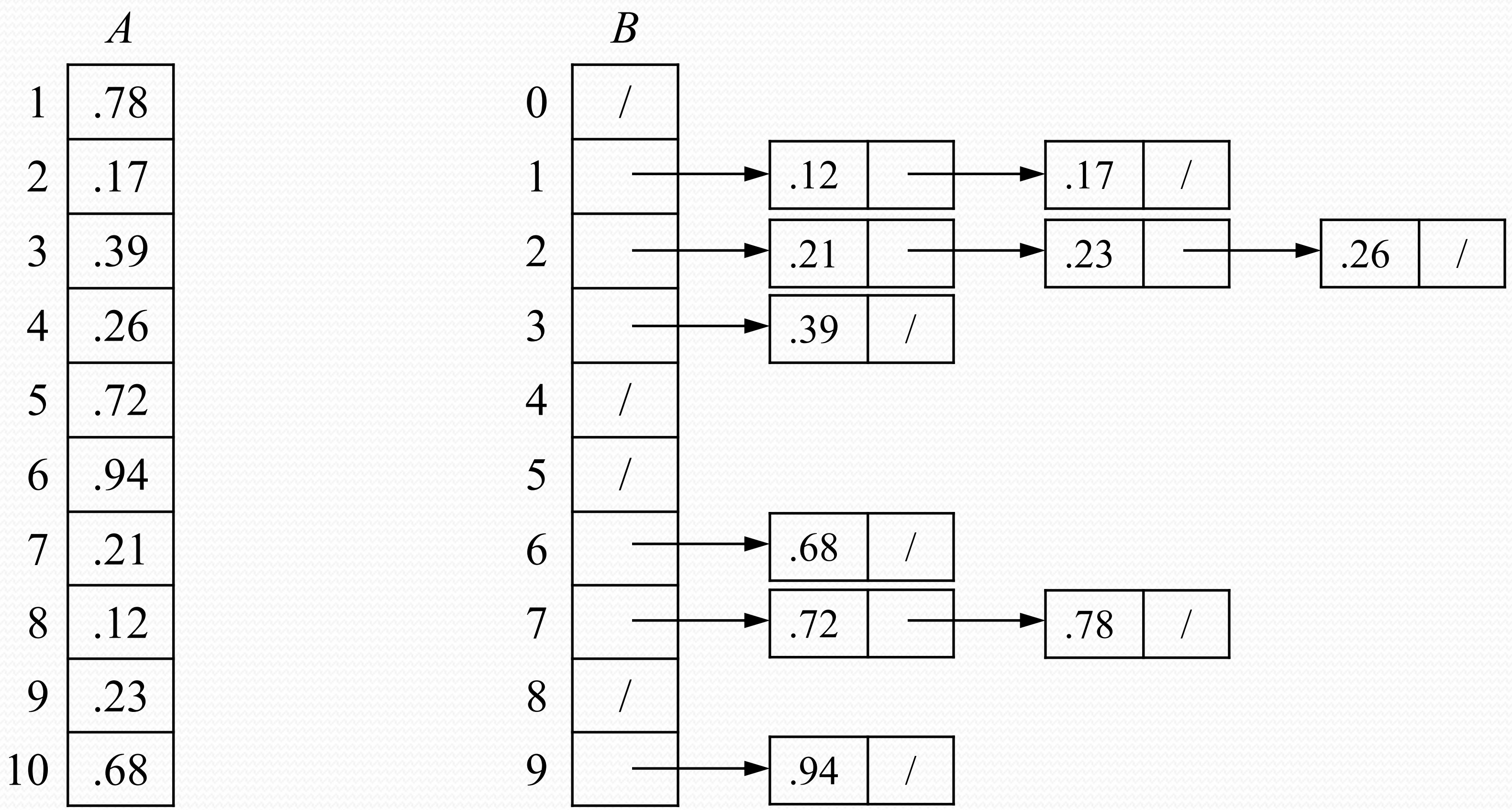
Input array: *A*



Operation of Bucket-Sort

Input array: *A*

Sort list *B*[*i*] with insertion sort



Correctness

- Consider $A[i]$, $A[j]$.
  - Assume without loss of generality that $A[i] \leq A[j]$.
  - Then $\lfloor n \cdot A[i] \rfloor \leq \lfloor n \cdot A[j] \rfloor$
  - So $A[i]$ is placed into the same bucket as $A[j]$ or into a bucket with a lower index
    - If same bucket, insertion sort fixes up.
    - If earlier bucket, concatenation of lists fixes up

Analysis

- Analysis
  - Relies on no bucket getting too many values
  - All lines of algorithm except insertion sorting take $\Theta(n)$ altogether
  - Intuitively, if each bucket gets a constant number of elements, it takes O(1) time to sort each bucket $\Rightarrow$ $O(n)$ sort time for all buckets
  - We “expect” each bucket to have few elements, since the average is 1 element per bucket
  - But we need to do a careful analysis

Analysis

- Define a random variable:

  n_i = the number of elements placed in bucket $B[i]$
- Because insertion sort runs in quadratic time, bucket sort time is

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

Analysis

- Take expectation of both sides:

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

$$= \Theta(n) + \sum_{i=0}^{n-1} E\left[\mathrm{O}(n_i^2)\right] \qquad \text{(linearity of expectation)}$$

$$= \Theta(n) + \sum_{i=0}^{n-1} \mathrm{O}\left(E\left[n_i^2\right]\right) \qquad (E[aX] = aE[X])$$

Analysis

- Claim

  $E[n_i^2] = 2 - (1/n)$ for $i = 0,\ldots, n - 1$

- Proof of claim

  Define indicator random variables:

  - $X_{ij} = I\,\{A[j]$ falls in bucket $i\}$
  - $\Pr\{A[j]$ falls in bucket $i\} = 1/n$
  - $n_i = \sum_{j=0}^{n} X_{ij}$

Analysis

- Claim

  $E[n_i^2] = 2 - (1/n)$ for $i = 0, \ldots, n-1$

- Proof of claim

  To compute $E[n_i^2]$, we expand the square and regroup term

- $$E\left[n_i^2\right] = E\left[\left(\sum_{j=1}^{n} X_{ij}\right)^2\right] = E\left[\sum_{j=1}^{n} X_{ij}^2 + 2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n} X_{ij}X_{ik}\right]$$

$$= \sum_{j=1}^{n} E\left[X_{ij}^2\right] + 2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n} E\left[X_{ij}X_{ik}\right]$$

Analysis

- Claim

  $E[n_i^2] = 2 - (1/n)$ for $i = 0,\ldots, n - 1$

- Proof of claim

$$E\left[X_{ij}^2\right] = 0^2 \cdot \Pr\{A[j] \text{ doesn't fall in bucket } i\} + 1^2 \cdot \Pr\{A[j] \text{ fall in bucket } i\}$$

$$= 0 \cdot \left(1 - \frac{1}{n}\right) + 1 \cdot \frac{1}{n}$$

$$= \frac{1}{n}$$

Analysis

- Claim

$$E[n_i^2] = 2 - (1/n) \text{ for } i = 0,\ldots, n-1$$

- Proof of claim

$E\lfloor X_{ij} X_{ik} \rfloor$ for $j \neq k$: since $j \neq k$, X_{ij} and X_{ik} are independent random variables

$$\Rightarrow E\lfloor X_{ij} X_{ik} \rfloor = E\lfloor X_{ij} \rfloor E[X_{ik}]$$

$$= \frac{1}{n} \cdot \frac{1}{n} = \frac{1}{n^2}$$

Analysis

- Claim

  $E[n_i^2] = 2 - (1/n)$ for $i = 0, \ldots, n-1$

- Proof of claim

Substituting these two expected values in $\sum_{j=1}^{n} E\left[X_{ij}^2\right] + 2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n} E\left[X_{ij}X_{ik}\right]$

$$E\left[n_i^2\right] = \sum_{j=1}^{n}\frac{1}{n} + \sum_{1\le j\le n}\ \sum_{\substack{1\le k\le n \\ k\ne j}}\frac{1}{n^2}$$

$$= n\cdot\frac{1}{n} + n(n-1)\cdot\frac{1}{n^2}$$

$$= 1 + \frac{(n-1)}{n} = 2 - \frac{1}{n}$$

Analysis

- Therefore:

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

  - With bucket sort, if the input isn't drawn from a uniform distribution on [0, 1), all bets are off (performance wise, but the algorithm is still correct)

Quiz

- Using the previous figure model, illustrate the operation of BUCKET-SORT on the array $A = \langle .79, .13, .16, .64, .39, .20, .89, .53, .71, .42 \rangle$

Quiz

- Explain why the worst-case running time for bucket sort is $\Theta(n^2)$

Quiz

- What is the worst-case running time if the algorithm use merge sort, instead of insertion sort when sorting the buckets?