# Bucket Sort

Adapted from the CLRS book slides

# BUCKET SORT

- Assumes that the input is generated by a random process that distributes elements uniformly and independently over [0, 1).
- ***Idea***
  - Divide [0, 1) into $n$ equal-sized *buckets*.
  - Distribute the $n$ input values into the buckets. *[Can implement the buckets with linked lists]*
  - Sort each bucket.
  - Then go through buckets in order, listing elements in each one.

PSEUDOCODE

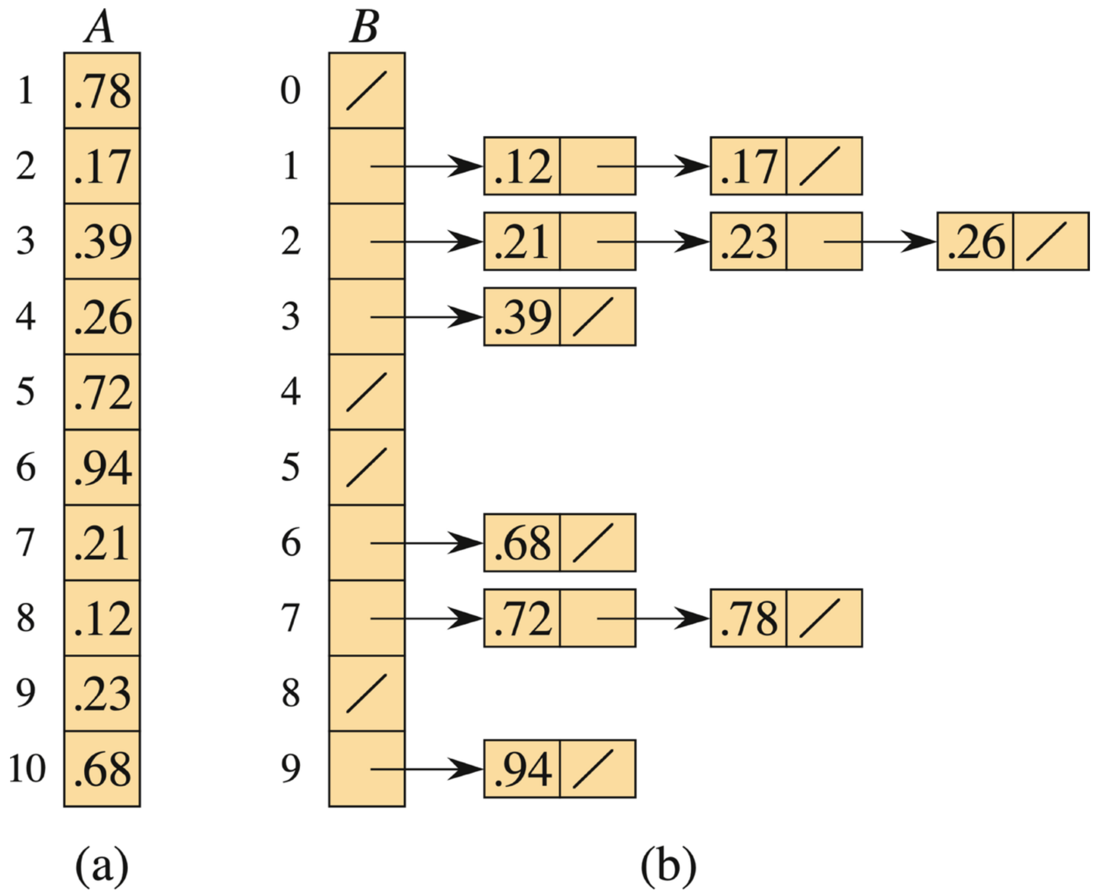
**Input:** $A[1:n]$, where $0 \leq A[i] < 1$ for all $i$.

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

```
BUCKET-SORT(A, n)
let B[0 : 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[⌊n · A[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
return the concatenated lists
```

# EXAMPLE

The buckets are shown after each has been sorted.



# 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 (continued)

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

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

Take expectations of both sides:

$$\begin{aligned}
\mathrm{E}[T(n)] &= \mathrm{E}\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right] \\
&= \Theta(n) + \sum_{i=0}^{n-1} \mathrm{E}\left[O(n_i^2)\right] \quad \text{(linearity of expectation)} \\
&= \Theta(n) + \sum_{i=0}^{n-1} O(\mathrm{E}\left[n_i^2\right]) \quad (\mathrm{E}[aX] = a\mathrm{E}[X])
\end{aligned}$$

# CLAIM

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

***Proof of claim***

View each $n_i$ as number of successes in $n$ Bernoulli trials
Success occurs when an element goes into bucket $B[i]$.

- Probability $p$ of success: $p = 1/n$.
- Probability $q$ of failure: $q = 1 - 1/n$.

Binomial distribution counts number of successes in $n$ trials: $\mathrm{E}[n_i] = np = n(1/n) = 1$ and $\mathrm{Var}[n_i] = npq = 1 - 1/n$

$$\begin{aligned}
\mathrm{E}\left[n_i^2\right] &= \mathrm{Var}[n_i] + \mathrm{E}^2[n_i] \\
&= (1 - 1/n) + 1^2 \\
&= 2 - 1/n
\end{aligned}$$

Therefore:

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

# CLAIM (continued)

Again, not a comparison sort. Used a function of key values to index into an array.

This is a ***probabilistic analysis***—we used probability to analyze an algorithm whose running time depends on the distribution of inputs.

Different from a ***randomized algorithm***, where we use randomization to *impose* a distribution.

With bucket sort, if the input isn't drawn from a uniform distribution on [0, 1), the algorithm is still correct, but might not run in $\Theta(n)$ time. It runs in linear time as long as the sum of squares of bucket sizes is $\Theta(n)$.