Bucket Sort

- Overview
- Algorithm
- Examples
- Analysis

Overview

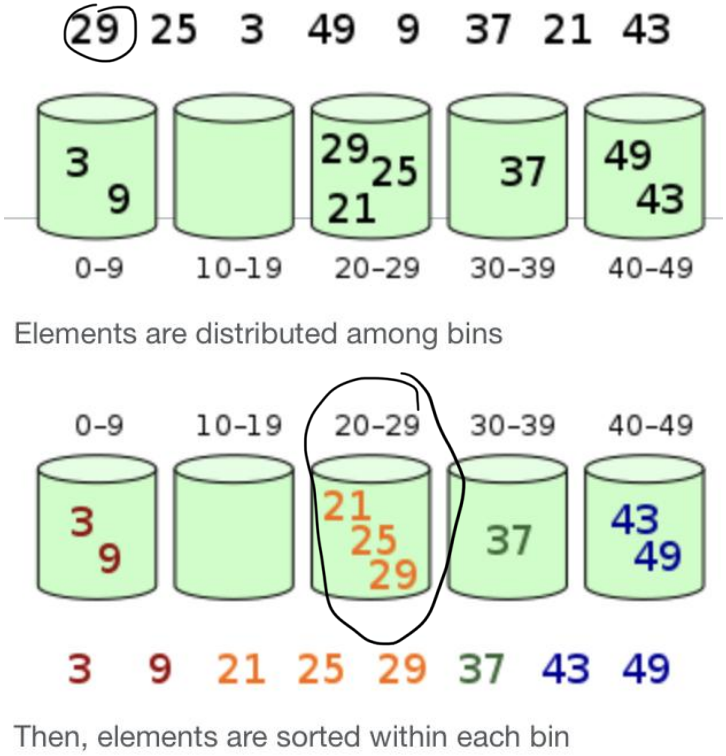
- Sort **N** items into **k** buckets
- Sort each bucket
- Concatenate the items in each bucket to form the sorted array of **N** elements

Function used in this case to sort items into buckets:

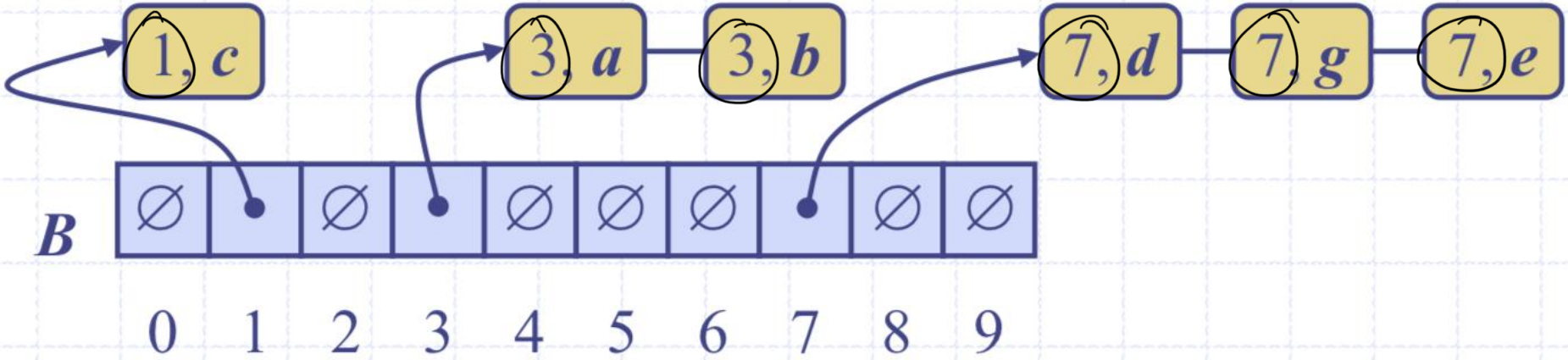
bucket index = floor(*key*/*M****k**)

$$\left\lfloor \frac{29}{49} \cdot 5 \right\rfloor = 2$$

where *M* = maximum key possible and *k* = number of buckets

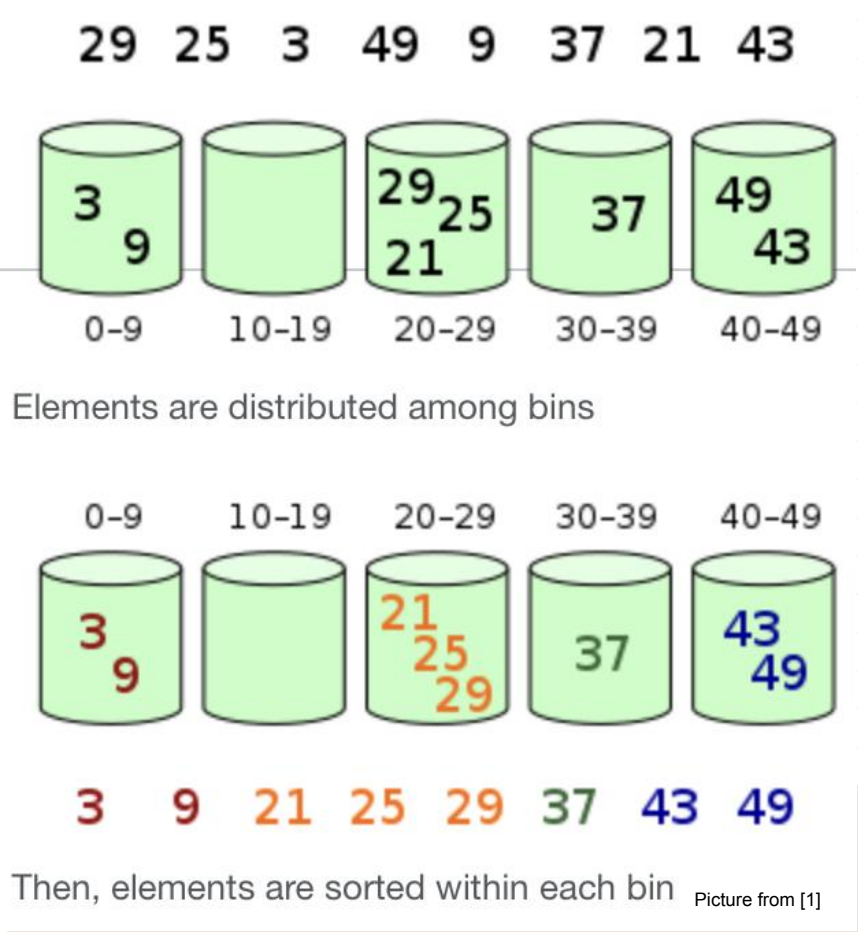


Picture from [1]



Picture from [2]

- How are buckets represented? **An array of linked lists (or possibly another structure).**
- How are items in each bucket sorted? **Typically, insertion sort.**

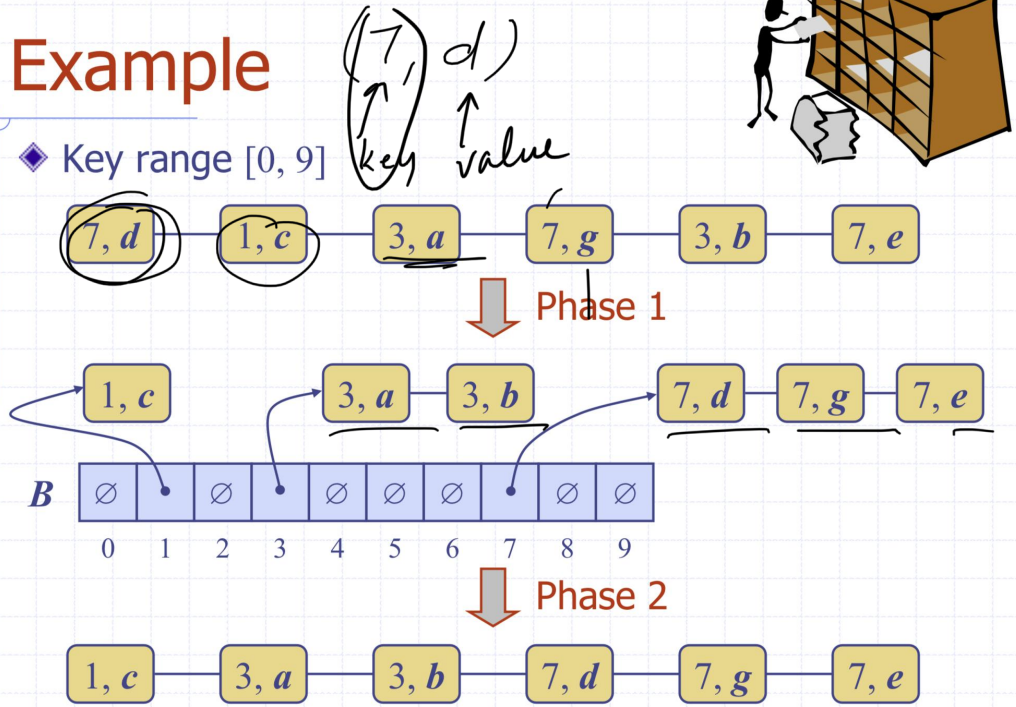


Picture from [1]

Picture from [2]

Example

- Key range [0, 9]



Compare/contrast these examples of bucket sort...

```
Algorithm bucketSort(S, k)
    Input sequence S of (key,
    element) items with keys in the
    range [0, k-1]
    Output sequence S sorted by
    non-decreasing keys
    B := array of k empty sequences
    while !S.isEmpty()
        add S.first() to the sequence
        at B[k] and remove it from S
    for i from 0 to k - 1
        sort the list at  B[i]
    Concatenate the lists from each
    bucket into a single list
```

Phase 1

Put items into buckets (Since the keys are indexes in the array, this can be done in O(N) time.)

(Phase 2)

Sort the items in each bucket (time depends on sort method and distribution of items, but generally $O(N^2/k)$ on average)

Phase 3

Concatenate the lists from each bucket into a single list (Since two lists can be concatenated in O(1) time and there are k buckets, this can be done in O(k) time.)

Analysis: The runtime depends on...

- The number of elements *N*
- The number of buckets *k*
- How the elements are distributed among the buckets
- The method for sorting the buckets (assume insertion sort)

- Worst case?

$O(N^2)$

- Best case?

$O(N)$

- Average case?

$O(N + N^2/k + k)$

Algorithm *bucketSort(S, k)*

Input sequence **S** of (key, element) items with keys in the range [0, **k**-1]

Output sequence **S** sorted by non-decreasing keys

B := array of **k** empty sequences

while !**S**.isEmpty()

add **S**.first() to the sequence at **B**[**k**] and remove it from **S**

for i from 0 to **k** - 1

sort the list at **B**[**i**]

Concatenate the lists from each bucket into a single list

Assuming 1 key per Bucket instead of a range of keys…(*N* items, *k* buckets)...

Phase 1

Put items into buckets (Since the keys are indexes in the array, this can be done in O(N) time.)

~~**(Phase 2)**~~

~~Sort the items in each bucket (time depends on sort method and distribution of items, but generally $O(N^2/k)$ on average)~~

Phase 3

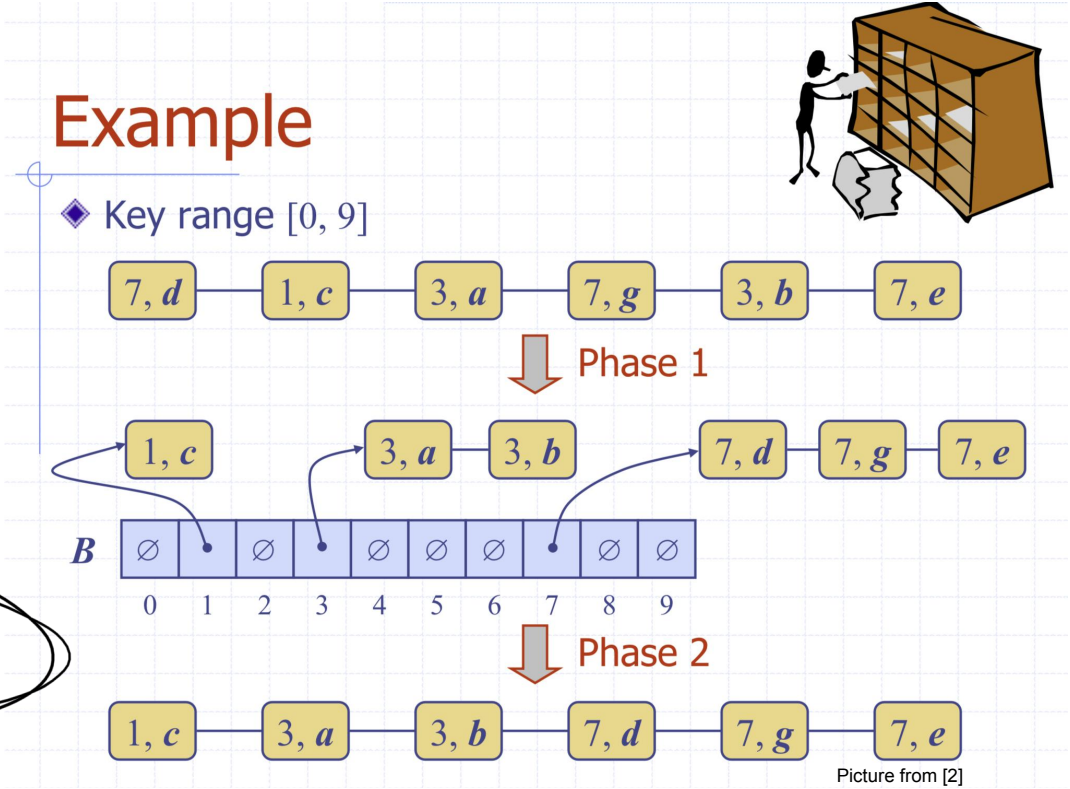
Concatenate the lists from each bucket into a single list (Since two lists can be concatenated in O(1) time and there are k buckets, this can be done in O(k) time.)

Analysis: Assuming 1 Key per bucket...

- number of elements *N*
- number of buckets *k*

Time is: O(N + k)—which is linear.

This is the version we will come back to in Radix Sort.



References

[1] https://en.wikipedia.org/wiki/Bucket_sort
[2] Tamassia and Goodrich