

VE281

Data Structures and Algorithms

Non-comparison Sort

Learning Objective:

- Understand three non-comparison sorts, counting sort, bucket sort, and radix sort

Outline

- ❖ Non-comparison Sort
 - ❖ Counting Sort
 - ❖ Bucket Sort
 - ❖ Radix Sort

Counting Sort

A Simple Version

- ❖ Sort an array **A** of **integers** in the range $[0, k]$, where k is known.
 1. Allocate an array **count**[**k+1**].
 2. Scan array **A**. For **i=1** to **N**, increment **count**[**A[i]**].
 3. Scan array **count**. For **i=0** to **k**, print **i** for **count**[**i**] times.
- ❖ Time complexity: $O(N + k)$.
- ❖ The algorithm can be converted to sort integers in some other known range $[a, b]$.
 - ❖ Minus each number by a , converting the range to $[0, b - a]$.

Example

2	5	3	0	2	3	0	3
---	---	---	---	---	---	---	---

Counting Sort

A General Version

- ❖ In the previous version, we print **i** for **count[i]** times
 - ❖ Simple but only works when sorting integer keys alone
 - ❖ How to sort items when there is “*additional*” information with each key?
Furthermore, how to guarantee the stability?
 - ❖ A general version:
 1. Allocate an array **C[k+1]**
 2. Scan array **A**. For **i=1** to **N**, increment **C[A[i]]**
 3. For **i=1** to **k**, **C[i]=C[i-1]+C[i]**
 - ❖ **C[i]** now contains number of items less than or equal to **i**
 4. For **i=N** downto **1**, put **A[i]** in new position **C[A[i]]** and decrement **C[A[i]]**

Counting Sort

Example

1. Allocate an array $C[k+1]$.

$k=5$

2. Scan array A . For $i=1$ to N ,
increment $C[A[i]]$.

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

3. For $i=1$ to k , $C[i]=C[i-1]+C[i]$

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

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

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

Counting Sort Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N ,
increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

$k=5$

	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	2	4	7	7	8

1	2	3	4	5	6	7	8
						3	

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

Why putting 3 at location 7 is correct?

Counting Sort

Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N , increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in new position $C[A[i]]$ and decrement $C[A[i]]$.

$k=5$

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N ,
increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

$k=5$

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

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

	1	2	3	4	5	6	7	8
		0				3	3	

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N ,
increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

$k=5$

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

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

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

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N ,
increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

$k=5$

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

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

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

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.
2. Scan array A. For $i=1$ to N , increment $C[A[i]]$.
3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$
4. For $i=N$ down to 1, put $A[i]$ in new position $C[A[i]]$ and decrement $C[A[i]]$.

$k=5$

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

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

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

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.

2. Scan array A. For $i=1$ to N ,
increment $C[A[i]]$.

3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$

4. For $i=N$ down to 1, put $A[i]$ in
new position $C[A[i]]$ and
decrement $C[A[i]]$.

$k=5$	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3

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

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

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

Counting Sort

Example

1. Allocate an array $C[k+1]$.
2. Scan array A. For $i=1$ to N , increment $C[A[i]]$.
3. For $i=1$ to k , $C[i] = C[i-1] + C[i]$
4. For $i=N$ down to 1, put $A[i]$ in new position $C[A[i]]$ and decrement $C[A[i]]$.

$k=5$

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

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

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

Done!

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

Is counting sort stable?

Yes!

Short Break

❖ Question time!

Outline

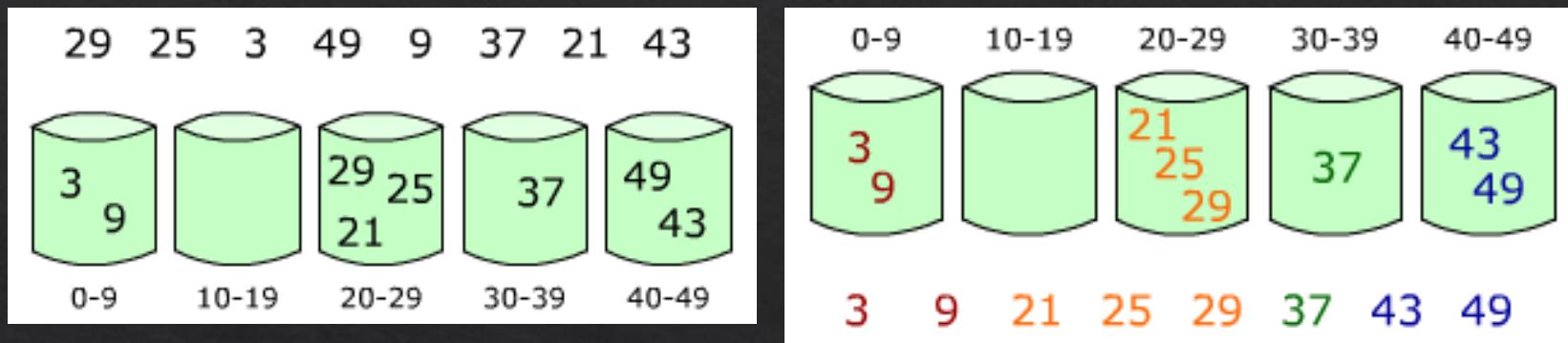
- ❖ Non-comparison Sort
 - ❖ Counting Sort
 - ❖ Bucket Sort
 - ❖ Radix Sort

Bucket Sort

- ❖ Instead of simple integer, each key can be a complicated record, such as a real value.
- ❖ Then instead of incrementing the count of each bucket, **distribute** the records **by their keys** into appropriate buckets.
- ❖ Algorithm:
 1. Set up an array of initially empty “buckets”.
 2. Scatter: Go over the original array, putting each object in its bucket.
 3. Sort each non-empty bucket by a comparison sort.
 4. Gather: Visit the buckets in order and put all elements back into the original array.

Bucket Sort

- ❖ Example



- ❖ Time complexity

- ❖ Suppose we are sorting cN items and we divide the entire range into N buckets.
- ❖ Assume that the items are **uniformly distributed** in the entire range.
- ❖ The average case time complexity is $O(N)$.

2nd Break

❖ Q&A

Outline

- ❖ Non-comparison Sort
 - ❖ Counting Sort
 - ❖ Bucket Sort
 - ❖ Radix Sort

Radix Sort

- ❖ **Radix sort** sorts integers by looking at one digit at a time.
- ❖ Procedure: Given an array of integers, from the least significant bit (LSB) to the most significant bit (MSB), repeatedly do **stable** bucket sort according to the current bit.
- ❖ For sorting base- b numbers, bucket sort needs b buckets.
 - ❖ For example, for sorting decimal numbers, bucket sort needs 10 buckets.

Radix Sort

Example

- ❖ Sort 815, 906, 127, 913, 098, 632, 278.
- ❖ Bucket sort 815, 906, 127, 913, 098, 632, 278 according to the least significant bit:

0	1	2	3	4	5	6	7	8	9
		63 <u>2</u>	91 <u>3</u>		81 <u>5</u>	90 <u>6</u>	12 <u>7</u>	09 <u>8</u>	27 <u>8</u>

- ❖ Bucket sort 632, 913, 815, 906, 127, 098, 278 according to the second bit.

Radix Sort

Example

- ❖ Bucket sort 632, 913, 815, 906, 127, 098, 278 according to the second bit.

0	1	2	3	4	5	6	7	8	9
<u>906</u>	<u>913</u>	<u>127</u>	<u>632</u>				<u>278</u>		<u>098</u>
	<u>815</u>								

- ❖ Bucket sort 906, 913, 815, 127, 632, 278, 098 according to the most significant bit.

Radix Sort

Example

- ❖ Bucket sort 906, 913, 815, 127, 632, 278, 098 according to the most significant bit.

0	1	2	3	4	5	6	7	8	9
<u>098</u>	<u>127</u>	<u>278</u>				<u>632</u>		<u>815</u>	<u>906</u>

- ❖ The final sorted order is: 098, 127, 278, 632, 815, 906, 913.

Radix Sort: Correctness

- ◊ Claim: after bucket sorting the i -th LSB, the numbers are sorted according to their last i digits
- ◊ Proof by mathematical induction
- ◊ Base case is obviously true
- ◊ Inductive step
 - ◊ Assume that according to the last i digits, order is $a_1 < \dots < a_n$
 - ◊ For two adjacent numbers a_k and a_{k+1} if they are not in the same bucket, they are sorted according to their last i^{th} digits
 - ◊ If they are in the same bucket, then $a_k < a_{k+1}$ for the last $(i - 1)$ bits. They are also sorted due to stability of bucket sort

Radix Sort

Time Complexity

- ❖ Let k be the maximum number of digits in the keys and N be the number of keys.
- ❖ We need to repeat bucket sort k times.
 - ❖ Time complexity for the bucket sort is $O(N)$.
- ❖ The total time complexity is $O(kN)$.

Radix Sort

- ❖ Radix sort can be applied to sort keys that are built on **positional notation**.
 - ❖ **Positional notation**: all positions uses the same set of symbols, but different positions have different weight.
 - ❖ Decimal representation and binary representation are examples of positional notation.
 - ❖ Strings can also be viewed as a type of positional notation. Thus, radix sort can be used to sort strings.
- ❖ We can also apply radix sort to sort records that contain multiple keys.
 - ❖ For example, sort records (year, month, day).

That is All for today!

❖ Questions?

