B.Y. Choueiry 8 Non-comparison sort algorithms Bucket sort: assumes something about input Radix sort, $\Theta(dn + kd)$ O(n), stable Counting sort: assumes something about input When d constant, $k = O(n) \Rightarrow$ linear time

March 5, 2001

Counting sort

В А Сропеіту

of 1 to kAssumes that each of the n input element is an integer in the range

When k = O(n), counting sort is linear

Principle

₽

• Determine for each input element x, the number of elements less than x

Every element x can be placed directly in its position in output

Example

If \exists 17 elements less than x, x must be in position 18

Slight modification for same value cases

March 5, 2001

March 5, 2001 В А Сропеіту Ţ CSCE310: Data Structures and Algorithms choueiry@cse.unl.edu, Tel: (402)472-5444 Textbook, Chapter 9, Sections 9.2 and 9.3 www.cse.unl.edu/~choueiry/S01-310/ Berthe Y. Choueiry (Shu-we-ri) Sorting in Linear Time Ferguson Hall, Room 104

В А Сропеіту 7 $\Omega(n \lg n)$: $O(n \lg n)$: • Mergesort, heapsort, quicksort • quicksort: average-case • Mergesort, heapsort: worst-case

comparisons between the input elements Interesting common property: sorted order is based only on

 \rightarrow Comparison sorts algorithms

We can prove that: any comparison sort algorithm is in $\Omega(n \lg n)$ (Section 1.1)

```
March 5, 2001
                                                                                                                                                 B. A. Choueiry
                                                                                     8
                                                                                                                   1 2 3 4 5 6 7 8
A 3 6 4 1 3 4 1 4
                                                                                                 1 2 3 4 5 6
C 1 2 4 6 7 8
                                                                 1 2 3 4 5 6 7 8
                                                                                        (a)
                                     (d)
                                                                                                           1 2 3 4 5 6
C 2 2 4 7 7 8
                                               1 2 3 4 5 6
C 1 2 4 5 7 8
                                                                 1 2 3 4 5 6 7 8
                                      (e)
                                                                                         Э
                                                                                                  1 2 3 4 5 6
C 2 2 4 6 7 8
                                                          1 1 3 3 4 5 6 7 8
                                                                                                                      1 2 3 4 5 6 7 8
                                       \ni
```

```
В.А. Сропецъ
March 5, 2001
                                                                                                                                                                                                                                      ç
                                                                                                                                                                                                                                                                                                                            Temporary working storage: Array C[1...2]
                                                                                                                                                                                                                                                                                                                                                        Output: Array B[1...n]
                                                                                                                                                                                                                                                                                                                                                                                    Input: Array A[1 \dots n], length[A] = n
                                                                                                                                                                                                                                                                                                                                                                                                                     Counting sort
                                                                                                                                                                                                                                                                                                Counting-Sort(A, B, k)
                                                                                                     \triangleright C[i] now contains the number of elements less than or equal to i.
                                                                                                                                                                         \triangleright C[i] now contains the number of elements equal to i.
                                                                                                                                                                                                                     for j \leftarrow 1 to length[A]
                                                                                                                                                                                                                                                                      for i \leftarrow 1 to k
                                                                                                                                                      for i \leftarrow 2 to k
                                                                                  for j \leftarrow
                                                                                                                                                                                                                                          do C[i] \leftarrow 0
                                                                                                                            do C[i] \leftarrow C[i] + C[i-1]
                                                                                                                                                                                              do C[A[j]] \leftarrow C[A[j]] + 1
                                                      do B[C[A[j]]] \leftarrow A[j]
                                                                                   length[A] downto 1
                                    C[A[j]] \leftarrow C[A[j]] - 1
```

```
9
Lines 9–11: A[j] is place in correct position in B
                                                                                                                   Lines 3–4: inspect each element, get values of C[i]
                                                                                                                                                         Lines 1-2: initialization
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    Counting sort
                                   Lines 6–7: number of elements \leq i (a running sum of C)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            Counting-Sort(A, B, k)
                                                                                                                                                                                                                                 10
                                                                                                                                                                                                                                                                                      9 8
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              1 for i \leftarrow 1 to k
                                                                                                                                                                                                                                                                                                                                                                                          \triangleright C[i] now contains the number of elements equal to i.
                                                                                                                                                                                                                                                                                                          \triangleright C[i] now contains the number of elements less than or equal to i.
                                                                                                                                                                                                                                                                                                                                                                                                                                                         for j \leftarrow 1 to length[A]
                                                                                                                                                                                                                                                                                      for j \leftarrow length[A] downto 1
                                                                                                                                                                                                                                                                                                                                                                         for i \leftarrow 2 to k
                                                                        C[i]: number of elements equal to i
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  do C[i] \leftarrow 0
                                                                                                                                                                                                                               do B[C[A[j]]] \leftarrow A[j]
C[A[j]] \leftarrow C[A[j]] - 1
                                                                                                                                                                                                                                                                                                                                                                                                                           do C[A[j]] \leftarrow C[A[j]] + 1
                                                                                                                                                                                                                                                                                                                                           do C[i] \leftarrow C[i] + C[i-1]
```

В.А. Сропецъ Π General use: sort records keyed by multiple fields Radix sort is stable Digit sorting must be stable Given numbers of d-digit, Radix-sort: Example: sort records by dates (years, months, and days) ယ 2 Radix sort 5. .. and the numbers are sorted! Repeats from 2, until last digit dMoves to the next least-significant digit Sorts the numbers according this digit Starts with the least significant digit first using a stable sorting algorithm

Exercise: 9.3-1

```
Counting sort

COUNTING-SORT(A, B, K)

1 for i - 1 to k

2 do C[j] - 0

3 for j - 1 to length[A]

4 do C[A[J]] - C[A[J]] + 1

5 \triangleright C[j] now contains the number of elements equal to i.

6 for i - 2 to k

7 do C[j - C[j] + C[i - 1]

8 \triangleright C[j] now contains the number of elements less than or equal to i.

9 for j - length[A] downing.

10 for j - length[A] downing.

11 C[A[J]] - A[J] - A[J]

Lines 1 - 2: O(k)

Lines 3 - 4: O(n)

Lines 6 - 7: O(k)

Lines 9 - 11: O(n)

Counting sort: O(k + n)

Counting sort: O(k + n)

Usually, used with k = O(n), this in O(n)
```

```
Counting sort: stable

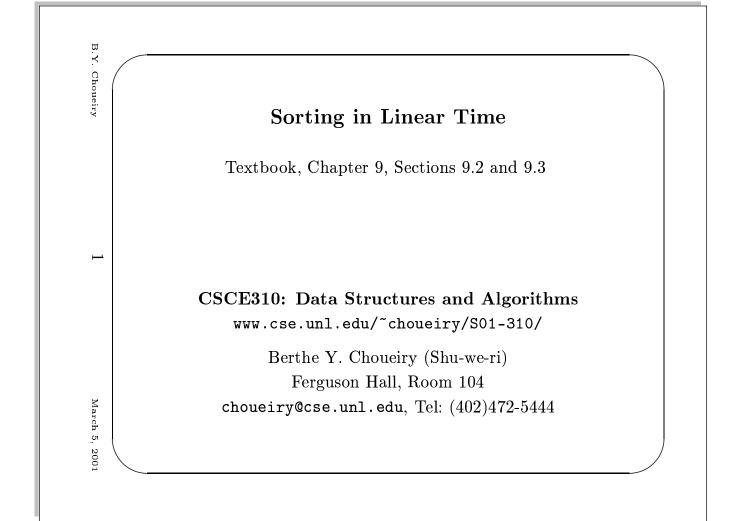
Counting sort: stable

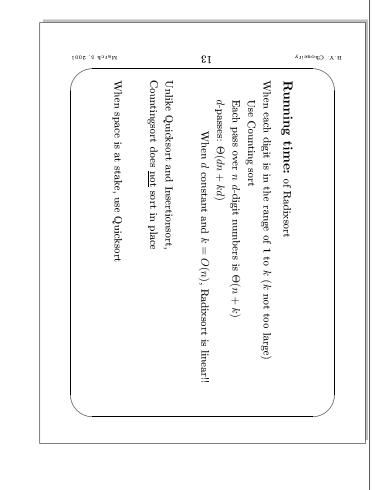
Numbers with the same value appear in B in same order as in A

Important in presence of satellite data

Exercise: 9.2-1

Try again in 9.2-3
```





Non-comparison sort algorithms

- Counting sort: assumes something about input O(n), stable
- Radix sort, $\Theta(dn + kd)$ When d constant, $k = O(n) \Rightarrow$ linear time
- Bucket sort: assumes something about input O(n)

March 5, 2001

B.Y. Choueiry

 $O(n \lg n)$:

- \bullet Mergesort, heapsort: worst-case
- quicksort: average-case

 $\Omega(n \lg n)$:

 \bullet Mergesort, heapsort, quicksort

2

Interesting common property: sorted order is based only on comparisons between the input elements

 \rightarrow Comparison sorts algorithms

March 5, 2001

We can prove that:

(Section 1.1)

any comparison sort algorithm is in $\Omega(n \lg n)$

```
B.Y. Choueiry
       Counting sort
      Input: Array A[1...n], length[A] = n
       Output: Array B[1 \dots n]
      Temporary working storage: Array C[1...2]
       Counting-Sort(A, B, k)
             for i \leftarrow 1 to k
                   do C[i] \leftarrow 0
         2
\mathbf{c}
          3
             for j \leftarrow 1 to length[A]
                   do C[A[j]] \leftarrow C[A[j]] + 1
             \triangleright C[i] now contains the number of elements equal to i.
             for i \leftarrow 2 to k
                   do C[i] \leftarrow C[i] + C[i-1]
          7
             \triangleright C[i] now contains the number of elements less than or equal to i.
             for j \leftarrow length[A] downto 1
          9
March 5, 200
                   do B[C[A[j]]] \leftarrow A[j]
        10
                        C[A[j]] \leftarrow C[A[j]] - 1
        11
```

B.Y. Choueiry

Counting sort

Assumes that each of the n input element is an integer in the range of 1 to k

When k = O(n), counting sort is linear

Principle

- Determine for each input element x, the number of elements less than x
- Every element x can be placed directly in its position in output array

Example

If \exists 17 elements less than x, x must be in position 18

Slight modification for same value cases

```
B.Y. Choueiry 7
```

Counting sort

```
COUNTING-SORT(A, B, k)

1 for i \leftarrow 1 to k

2 do C[i] \leftarrow 0

3 for j \leftarrow 1 to length[A]

4 do C[A[j]] \leftarrow C[A[j]] + 1

5 \triangleright C[i] now contains the number of elements equal to i.

6 for i \leftarrow 2 to k

7 do C[i] \leftarrow C[i] + C[i-1]

8 \triangleright C[i] now contains the number of elements less than or equal to i.

9 for j \leftarrow length[A] downto 1

10 do B[C[A[j]]] \leftarrow A[j]

11 C[A[j]] \leftarrow C[A[j]] - 1
```

March 5, 2001

```
Lines 9–11: A[j] is place in correct position in B correct final position for A[j] is C[A[j]] Since some x may not be different, need to decrement C[A[j]] when placing an A[j] into B
```

B.Y. Choueiry

Counting sort

```
COUNTING-SORT(A, B, k)

1 for i \leftarrow 1 to k

2 do C[i] \leftarrow 0

3 for j \leftarrow 1 to length[A]

4 do C[A[j]] \leftarrow C[A[j]] + 1

5 \triangleright C[i] now contains the number of elements equal to i.

6 for i \leftarrow 2 to k

7 do C[i] \leftarrow C[i] + C[i-1]

8 \triangleright C[i] now contains the number of elements less than or equal to i.

9 for j \leftarrow length[A] downto 1

10 do B[C[A[j]]] \leftarrow A[j]

11 C[A[j]] \leftarrow C[A[j]] - 1
```

6

Lines 1–2: initialization

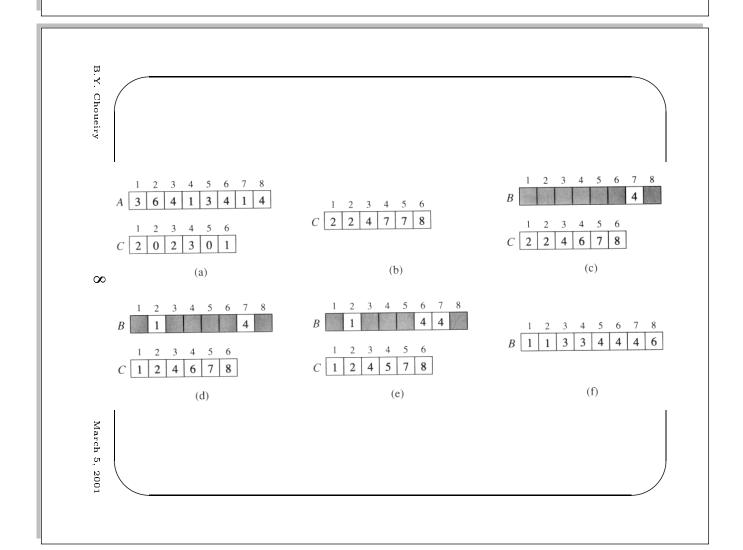
Lines 3–4: inspect each element, get values of C[i]

C[i]: number of elements equal to i

Lines 6–7: number of elements $\leq i$ (a running sum of C)

Lines 9–11: A[j] is place in correct position in B

```
B.Y. Choueiry
        Counting sort
                            Counting-Sort(A, B, k)
                                 for i \leftarrow 1 to k
                                     do C[i] \leftarrow 0
                             3 for j \leftarrow 1 to length[A]
                                      do C[A[j]] \leftarrow C[A[j]] + 1
                                 \triangleright C[i] now contains the number of elements equal to i.
                                 for i \leftarrow 2 to k
                                      do C[i] \leftarrow C[i] + C[i-1]
                              8 \triangleright C[i] now contains the number of elements less than or equal to i.
                              9 for j \leftarrow length[A] downto 1
                                      do B[C[A[j]]] \leftarrow A[j]
9
                                         C[A[j]] \leftarrow C[A[j]] - 1
                             11
        Lines 1–2: O(k)
        Lines 3–4: O(n)
        Lines 6–7: O(k)
        Lines 9–11: O(n)
March 5, 2001
        Counting sort: O(k+n)
        Usually, used with k = O(n), this in O(n)
```



B.Y. Choueiry

Radix sort

Given numbers of d-digit, Radix-sort:

- 1. Starts with the least significant digit first
- 2. Sorts the numbers according this digit

using a stable sorting algorithm

- 3. Moves to the next least-significant digit
- 4. Repeats from 2, until last digit d
- 5. .. and the numbers are sorted!

Digit sorting must be stable

Radix sort is stable

Example: sort records by dates (years, months, and days)

General use: sort records keyed by multiple fields

March 5, 2001

B.Y. Choueiry

Counting sort: stable

Numbers with the same value appear in B in same order as in A

Important in presence of satellite data

Exercise: 9.2-1

Try again in 9.2-3

10

13

Running time: of Radixsort

When each digit is in the range of 1 to k (k not too large)

Use Counting sort

Each pass over n d-digit numbers is $\Theta(n+k)$

d-passes: $\Theta(dn + kd)$

When d constant and k = O(n), Radixsort is linear!!

Unlike Quicksort and Insertionsort, Countingsort does not sort in place

When space is at stake, use Quicksort

March 5, 2001

B.Y. Choueiry

Input: A, array of n elements,

each of d digits: 1 lowest-order digit, d highest-order digit

For $i \leftarrow 1$ to d

 \mathbf{do} use a stable sort to sort array A on digit i

329		720		720		329
457		355		329		355
657		436		436		436
839	\Rightarrow	457	\Rightarrow	839	\Rightarrow	457
436		657		355		657
720		329		457		720
355		839		657		839
		1		1		1

12

Correctness: proof by induction on column being sorted Running time: depends on intermediate sorting algorithm

March 5, 2001

Exercise: 9.3-1