CMPT307: Sorting in Linear Time

Week 5-3

Xian Qiu

Simon Fraser University

xianq@sfu.ca

Outline

- counting sort
- radix sort
- bucket sort

Counting Sort

- assume $a_i \in \{0, 1, \ldots, k\}$ for $i = 1, \ldots, n$
- idea: place a_i directly into position $[a_i]$

0	1	2	3	4	5
0			3	4	

- problem: some numbers may have the same value
- solution: record the number of a_i at position $[a_i]$

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

- how to output result?

Straight Forward Way

- define $C[i] = \#i$
- add $C[i]$ copies of i to sorted list for $i = 1, \ldots, k$

	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	0	2	3	0	1

$C[i] = \#i$

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

- drawback: only yields sorted "keys", and loses the information of A, when $A[i]$ is an "object"

General Approach

- define $C[i] = \#$elements that are at most i
- if $a_i = i$, place a_i into position $[C[i]]$

	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	0	2	3	0	1

$C[i] = \#i$

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

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

- stable sorting: identical numbers appear in B in the same order as they do in A

Pseudocode

COUNTING-SORT(A, B, k)

```
1 let C[0..k] = 0 be a new array;
2 for j = 1 to A.length do
3     C[A[j]] = C[A[j]] + 1;                          // count #A[j]
4 for i = 1 to k do
5     C[i] = C[i] + C[i − 1];                         // count #{elements ≤ A[j]}
  // stable sorting
6 for j = A.length down to 1 do
7     B[C[A[j]]] = A[j];
8     C[A[j]] = C[A[j]] − 1;
```

- running time $= \Theta(n + k)$
- what if k is very large?

Radix Sort

- large number can be represented as d-digit number
- each digit takes values in $\{0, \ldots, k\}$

Example. 3 digits decimal numbers ($k = 9$)

$$\begin{array}{ccccccc} 457 & & 382 & & 457 & & 382 \\ 756 & \longrightarrow & 457 & \longrightarrow & 756 & \longrightarrow & 756 \\ 382 & & 756 & & 382 & & 457 \end{array}$$

wrong!

left to right

- sort column by column (left to right or right to left?)

Radix Sort

- we can show that right-to-left yields correct sequence
- however, the sorting algorithm must be stable

$$\begin{array}{cccccccc} 759 & & 382 & & 746 & & 382 \\ 746 & \longrightarrow & 746 & \longrightarrow & 759 & \longrightarrow & 759 \\ 382 & & 759 & & 382 & & 746 \end{array}$$

right to left — unstable sorting

- digits order: $d, d-1, \ldots, 1$

RADIX-SORT(A, d)

```
1 for i = 1 to d do
2     apply a stable sort to A on digit i;
```

Remarks

- applying Counting-Sort for Radix-Sort yields a running time $\Theta(d(n+k))$
- given a b-bit number, we can represent it as a d-digit number by splitting up

$$\underbrace{10110010}_{r}\underbrace{11010100}_{r}\underbrace{01101010}_{r}\underbrace{10111100}_{r}$$

$$d \leq \lceil b/r \rceil, \text{thus running time is } \Theta\left(\frac{b}{r}(n+2^r)\right)$$

- if $b \leq \lfloor \log n \rfloor$, letting $r = b$ yields $\Theta(n)$
- else letting $r = \lfloor \log n \rfloor$ yields $\Theta(bn/\log n)$

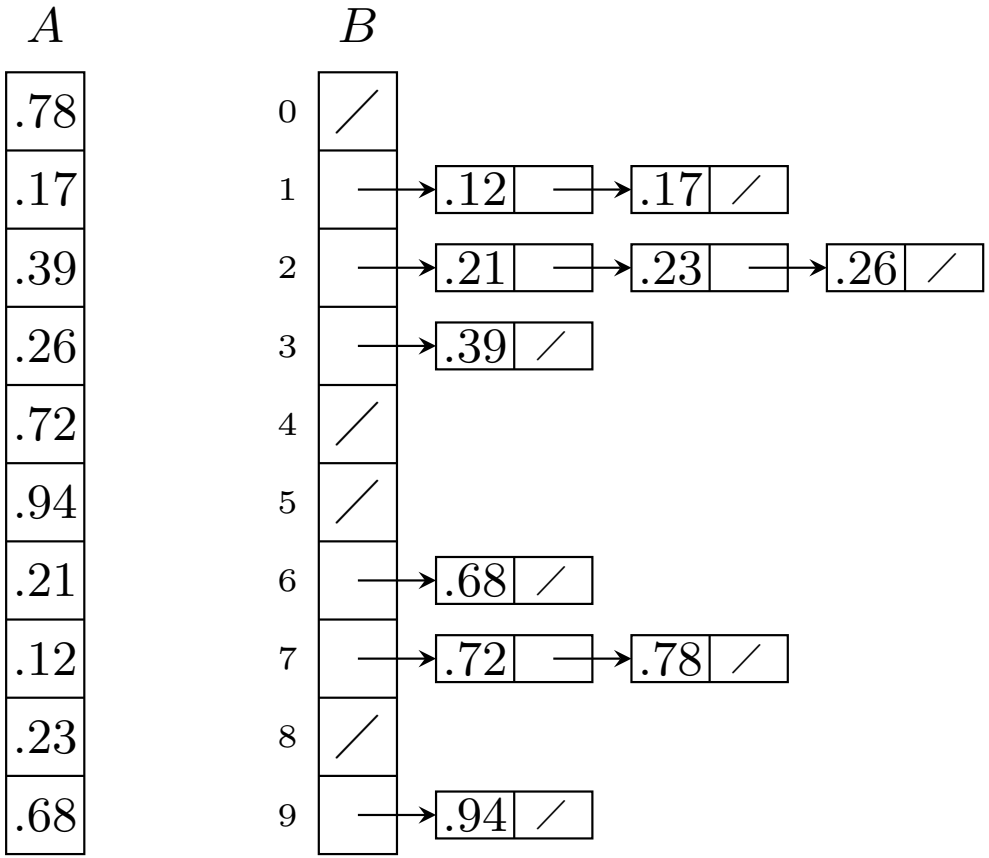
Bucket Sort

Assumptions

$a_1, \ldots, a_n$ are independent and uniformly distributed in [0,1)

- divide $[0, 1)$ into n equal-sized subintervals, or buckets
- first place numbers into buckets, then sort the numbers in each bucket
- use data structure – linked list

Example


A
B
.78
.17
.39
.26
.72
.94
.21
.12
.23
.68
0
1
2
3
4
5
6
7
8
9
.12
.17
.21
.23
.26
.39
.68
.72
.78
.94

Pseudocode

BUCKET-SORT(A, B, k)

```
1 let B[0..n − 1] be a new array;
2 n = A.length;
3 for i = 0 to n − 1 do
4     make B[i] an empty list;
5 for i = 1 to n do
6     insert A[i] into list B[⌊nA[i]⌋];
7 for i = 0 to n − 1 do
8     sort list B[i] with insertion sort;
9 concatenate the lists B[0], . . . , B[n − 1] together in order;
```

Analysis

$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2), \quad \text{where } n_i := |B[i]|$$

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

$$X_{ij} = \mathrm{I}\{A[j] \text{ falls in bucket } i\} \Rightarrow n_i = \sum_{j=1}^{n} X_{ij}$$

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

$$\mathbb{E}[X_{ij}^2] = 1^2\frac{1}{n} = \frac{1}{n} \quad \mathbb{E}[X_{ij}X_{ik}] = \mathbb{E}[X_{ij}]\mathbb{E}[X_{ik}] = \frac{1}{n^2}$$

$$\mathbb{E}[n_i^2] = n\cdot\frac{1}{n} + n(n-1)\cdot\frac{1}{n^2} = 2 - \frac{1}{n} \quad \Rightarrow \mathbb{E}[T(n)] = \Theta(n)$$

Summary of Sorting Algorithms

algorithm	worst-cast time	average-case time	note
insertion sort	$\Theta(n^2)$	$\Theta(n^2)$	
merge sort	$\Theta(n \log n)$	$\Theta(n \log n)$	not in place
heap sort	$\Theta(n \log n)$	—	
quick sort	$\Theta(n^2)$	$\Theta(n \log n)$	fast in practice
counting sort	$\Theta(k+n)$	$\Theta(k+n)$	$a_i \in \{0, \ldots, k\}$
radix sort	$\Theta(d(n+k))$	$\Theta(d(n+k))$	d-digit numbers
bucket sort	$\Theta(n^2)$	$\Theta(n)$	uniform distribution