Linear Sorting

Chapter 8

Counting Sort

- Depends on a key assumption:
  - numbers to be sorted are integers in {0, 1, ..., k}
- **Input:** A[1..n]
- **Output:** B[1..n], sorted. B is assumed to be already allocated and is given as a parameter
- **Auxiliary storage:** C[0..k]

COUNTING-SORT(A, B, k)

Example

- $2_1, 5_1, 3_1, 0_1, 2_2, 3_2, 0_2, 3_3$

Analysis

- Is counting sort stable?
  - What does stable mean?
- Analysis:
- How big of k is practical?

Your Turn

- A: <6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2>

Radix Sort

- How IBM made its money. Punch card readers for census tabulation in early 1900's. Card sorters, worked on one column at a time. It's the algorithm for using the machine that extends the technique to multi-column sorting. The human operator was part of the algorithm!

- We're going to sort d digits

RADIX-SORT(A, d)

	one's place	ten's place	100s place
329			
457			
657			
839			
436			

Bucket Sort

- Assumption: input is generated by a random process that distributes elements uniformly over [0,1)

- Idea:

Bucket Sort

- Input: A[1..n], where for all i
- Auxiliary array: B[0..n-1] of linked lists, each list initially empty.

BUCKET-SORT(A)

Example

- A:<.78, .17, .39, .26, .72, .94, .21, .12, .23, .68>