Sorting Methods I

Andrew C. Lee

EECS, Syracuse

Contents

The Sorting Problem

- A common computer science problem:
  Search through a given data domain and find a solution
- Data can be preprocessed to speed up search (eg. applying binary search to a sorted array)
- Will discuss *one dimensional* sorting methods
- Like to know how to choose a suitable sorting method

The Sorting Problem

- A list of items
- Items identified by a key
- keys can be ordered
- *in-place* sorting method:
  requires at most a *constant* amount (i.e. independent of input size) of additional memory
- *stable* sorting method:
  relative order of duplicate occurrences of a data value is preserved
  Note: A stable sorting method can help us to sort an additional dimension at no extra costs.

Basic Sorting Methods: Insertion Sort

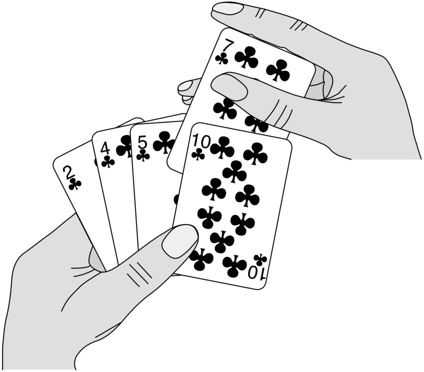

Figure : Insertion Sort: Intuitions

Basic Sorting Methods: Insertion Sort

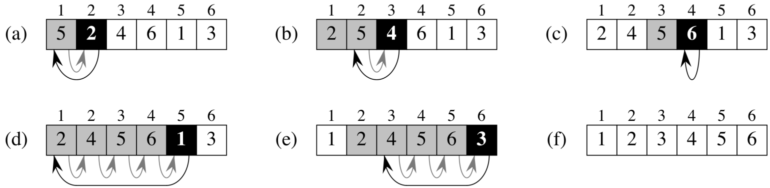

Figure : Applying Insertion Sort to an Array

Basic Sorting Methods: Insertion Sort

INSERTION-SORT(A, n)	*cost*	*times*
for $j = 2$ **to** n	c_1	n
$\quad key = A[j]$	c_2	$n - 1$
$\quad$ // Insert $A[j]$ into the sorted sequence $A[1 \mathinner{..} j-1]$.	0	$n - 1$
$\quad i = j - 1$	c_4	$n - 1$
$\quad$ **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$\qquad A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$\qquad i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$\quad A[i+1] = key$	c_8	$n - 1$

Figure : Insertion Sort: Pseudocode

Basic Sorting Methods: Bubble Sort

```
BUBBLESORT(A)
for i = 1 to A.length - 1
    for j = A.length downto i + 1
        if A[j] < A[j - 1]
            exchange A[j] with A[j - 1]
```

Figure : Bubble Sort

Basic Sorting Methods: Shell Sort

- Both insertion sort and bubble sort apply operations (move, exchange) to adjacent elements only
- Shell sort: attempt to apply these operations with *larger jumps*
- Example:
  1. Begin by sorting every 4^{th} element of the list
  2. Sort every 2^{nd} element of the list
  3. Sort every element of the list

Tree Based Sorting Methods

Recall the use of the *heap* data structure

- Heap sort (a.k.a. tree sort)
- Is Heap sort *in-place* ?
- Is Heap sort *stable*?

Divide and Conquer Based Methods

Question Name a problem where you can apply the *divide and conquer* paradigm to solve it.

For sorting, we consider the following two examples:

1. Merge Sort
2. Quick Sort