CSC3100 Data Structures Tutorial 11: Sorting

Jiawei Lian

The Chinese University of Hong Kong, Shenzhen
School of Data Science

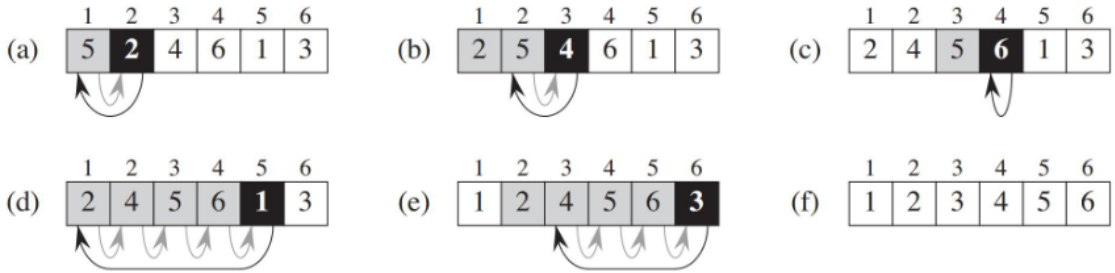
119030043@link.cuhk.edu.cn

December 6, 2021

Content

Outline

Review: Insertion Sort



INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Figure: Insertion sort

Outline

Main Idea: Divide-and-Conquer

1. Divide:
   - Take one element $A[q]$ from A as pivot, then partition the rest of A into two sub-arrays $A[p \cdots q-1]$ and $A[q+1 \cdots r]$, such that $A[p \cdots q-1] \leq A[q] < A[q+1 \cdots r]$
2. Conquer:
   - Sort $A[p \cdots q-1]$ and $A[q+1 \cdots r]$ by recursively calling Quick Sort.

Partition

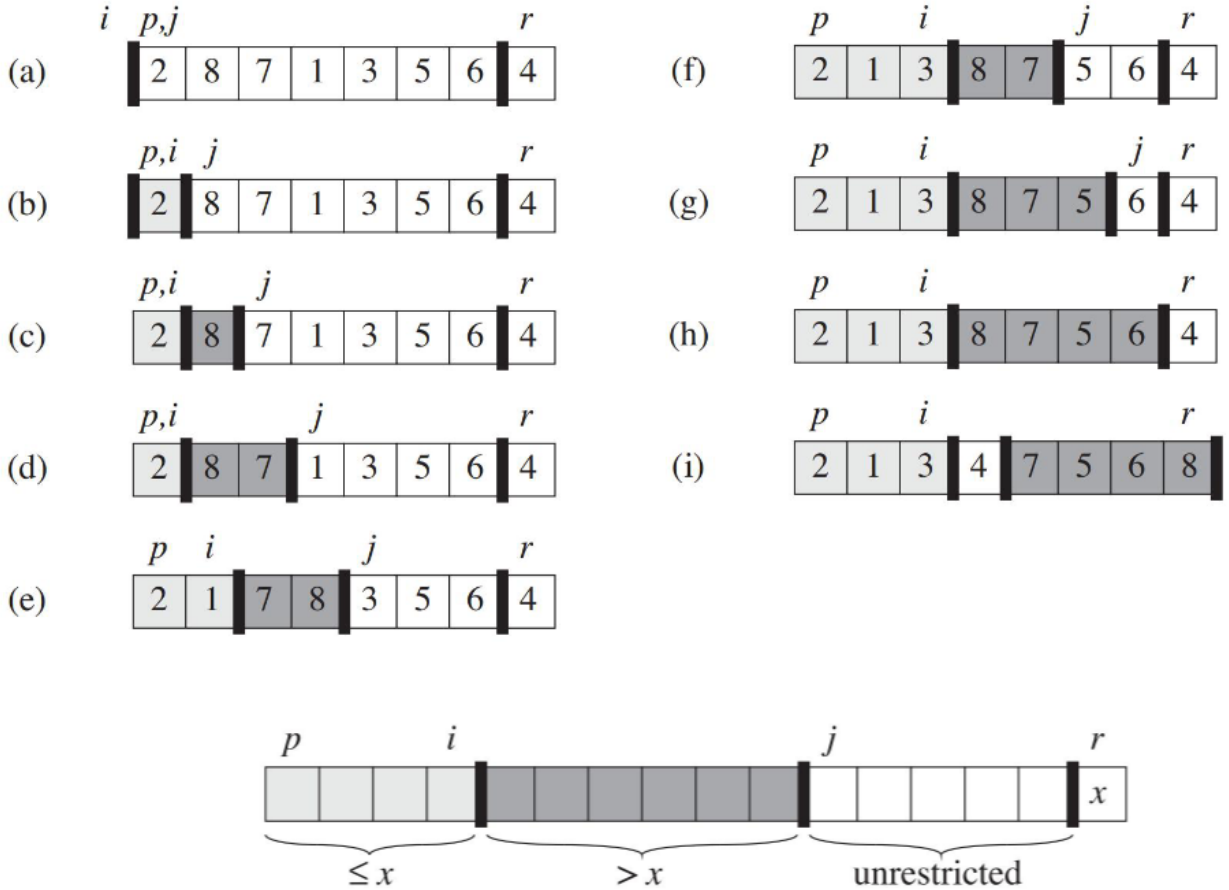


Figure: Partition process

Partition

```
PARTITION(A, p, r)
1   x = A[r]
2   i = p − 1
3   for j = p to r − 1
4       if A[j] ≤ x
5           i = i + 1
6           exchange A[i] with A[j]
7   exchange A[i + 1] with A[r]
8   return i + 1
```

Figure: Partition pseudo code

Recursion

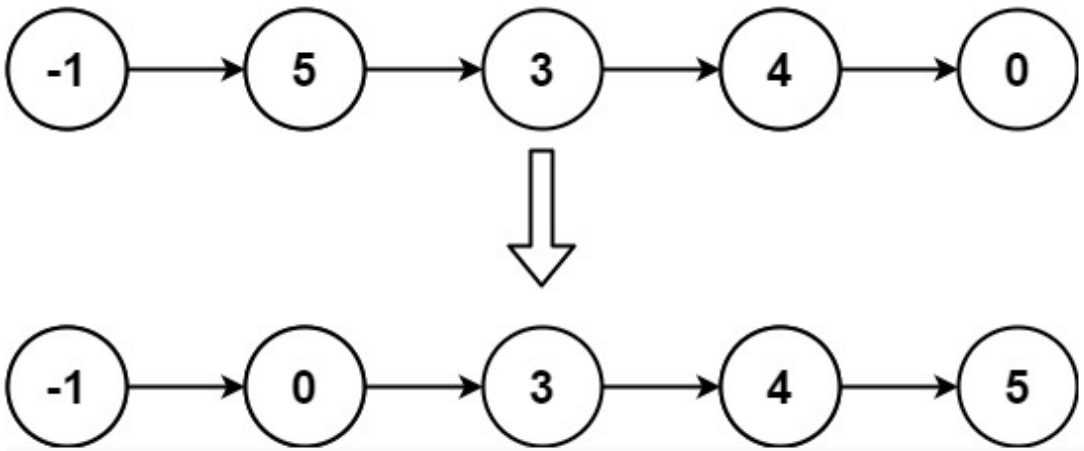
```
QUICKSORT(A, p, r)
1   if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)
```

Figure: Recursion pseudo code

Outline

Problem 1: Insertion Sort List

1. Problem statement:
   Given the head of a singly linked list, sort the list using insertion sort, and return the sorted list's head.
2. Example:




```
Input: head = [-1,5,3,4,0]
Output: [-1,0,3,4,5]
```

Figure: Example

Problem 1: Insertion Sort List - Solution

```java
class Solution {
    public ListNode insertionSortList(ListNode head) {
        ListNode dummy = new ListNode(0), pre;
        dummy.next = head;

        while(head != null && head.next != null) {

            if(head.val <= head.next.val) {
                head = head.next;
                continue;
            }

            pre = dummy;

            while (pre.next.val < head.next.val) pre = pre.next;

            ListNode curr = head.next;
            head.next = curr.next;
            curr.next = pre.next;
            pre.next = curr;
        }
        return dummy.next;
    }
}
```

Figure: Solution of example 1

Outline

Problem 2: Find K Smallest Numbers

1. Problem statement:
   - Given a group of N numbers, determine the first K^{th} smallest numbers, where k $\leq$ N.
2. Example:
   - Input: arr = [1,3,5,7,2,4,6,8], k = 4
   - Output: [1,3,2,4]
3. Constraints:
   - $0 \leq$ len(arr) ≤ 100000
   - $0 \leq$ k $\leq$ min(100000, len(arr))

Find K Smallest Numbers - Solution 1

1. Directly sort this group of N numbers in an ascending order
2. Return the first k numbers

Find K Smallest Numbers - Solution 2

1. Read the first K numbers into an array with size of K, sort these numbers in an ascending order
2. Read the remaining elements one by one
   - Ignore the element if it is larger or equal to the K^{th} element in the array
   - Otherwise, place the element into its correct position
3. After checking all the numbers, the remaining array is the answer.

Find K Smallest Numbers - Solution 3

1. Use Quick Sort

```java
class Solution {
    public int[] smallestK(int[] arr, int k) {
        quickSort(arr, 0, arr.length - 1, k);
        int[] result = new int[k];
        for (int i = 0; i < k; i++) {
            result[i] = arr[i];
        }
        return result;
    }

    private static void quickSort(int[] arr, int startIndex, int endIndex, int k) {
        if (startIndex >= endIndex) {
            return;
        }
        int partitionIndex = quickSortPartition(arr, startIndex, endIndex);
        if (partitionIndex == k) {
            return;
        }
        if (k < partitionIndex) {
            quickSort(arr, startIndex, partitionIndex - 1, k);
        } else {
            quickSort(arr, partitionIndex + 1, endIndex, k);
        }
    }

    private static int quickSortPartition(int[] arr, int startIndex, int endIndex) {
        int pivot = arr[endIndex];
        int i = startIndex;
        for (int j = startIndex; j <= endIndex - 1; j++) {
            if (arr[j] < pivot) {
                int temp = arr[j];
                arr[j] = arr[i];
                arr[i] = temp;
                i++;
            }
        }
        int temp = arr[endIndex];
        arr[endIndex] = arr[i];
        arr[i] = temp;
        return i;
    }
}
```

Figure: Solution of example 2

References I

- https://leetcode.com/problems/insertion-sort-list/
- https://leetcode-cn.com/problems/smallest-k-lcci/

CSC3100 Data Structures Tutorial 11: Sorting

Jiawei Lian

The Chinese University of Hong Kong, Shenzhen
School of Data Science

119030043@link.cuhk.edu.cn

December 6, 2021