Εργαστήριο 03: Πίνακες/Συλλογές
Μέγιστο Άθροισμα Υπο-ακολουθίας

Πρόβλημα Μέγιστο Άθροισμα Υπο-ακολουθίας

Το πρόβλημα Μέγιστο Άθροισμα Υπο-ακολουθίας έχει ως εξής:

- Δοθέντος μιας σειράς από θετικούς και αρνητικούς αριθμούς βρείτε την υπο-σειρά που έχει το μεγαλύτερο άθροισμα.
  Για παράδειγμα στην σειρά { 2, -3, 4, 1, -8 } η υπο-σειρά με το μέγιστο άθροισμα είναι η { 4, 1 } στις θέσεις 2-3 με άθροισμα 5.

- Μια μέθοδος εύρεσης της υπο-σειράς με το μέγιστο άθροισμα είναι η εξής:
  - Χρησιμοποιώντας δύο δείκτες i, j υπολογίζουμε όλα τα αθροίσματα των υπο-σειρών i έως j και κρατάμε το μεγαλύτερο.
  - Η μέθοδος σταματά όταν και οι δύο δείκτες φτάσουν στο τέλος της εισόδου.

  Για παράδειγμα σε μια σειρά μήκους 4 υπολογίζουμε τα αθροίσματα των θέσεων 0-0, 0-1, 0-2, 0-3, 1-1, 1-2, 1-3, 2-2, 2-3, 3-3.

Άσκηση 1

- Στο αρχείο MaxSumTest.java υλοποιούνται κάποιες μεθόδοι για το πιο πάνω πρόβλημα.
  - Συμπληρώστε την μέθοδο MaxSumSub1 έτσι ώστε να υλοποιεί την πιο πάνω μέθοδο.
  - Μελετήστε τις μεθόδους MaxSumSub που είναι υλοποιημένες. Ποια είναι η πολυπλοκότητα της κάθε μεθόδου;

Three/Two loops Method

For the given array {-2, -5, **6, -2, -3, 1, 5**, -6}, the maximum subarray sum is 7 (see highlighted elements).

1. Run three Nested Loops for all possible combinations $O(n^3)$

2. The second naive method is to run two loops. The outer loop picks the **beginning element**, the inner loop finds the maximum possible sum with first element picked by outer loop and compares this maximum with the overall maximum.

Finally return the overall maximum. The time complexity of the Naive method is $O(n^2)$.

3. The idea of the **Kadane's algorithm** is to look for all positive contiguous segments of the array and keep track of maximum sum contiguous segment among all positive segments. Each time we get a positive sum compare it with maxSum and update maxSum if it is greater.

O(n)

Recursive Method

4. Using **Divide and Conquer** approach, we can find the maximum subarray sum in **O(nLogn)** time. Following is the Divide and Conquer algorithm.

1) Divide the given array in two halves

2) Return the maximum of following three

   a) Maximum subarray sum in left half (Make a recursive call)

   b) Maximum subarray sum in right half (Make a recursive call)

   c) Maximum subarray sum such that the subarray crosses the midpoint

The lines 2.a and 2.b are simple recursive calls.

How to find maximum subarray sum such that the subarray crosses the midpoint?

We can easily find the crossing sum in linear time. The idea is simple, find the maximum sum starting from mid point and ending at some point on left of mid, then find the maximum sum starting from mid + 1 and ending with sum point on right of mid + 1. Finally, combine the two and return.

Time Complexity: maxSubArraySum() is a recursive method and time complexity can be expressed as following recurrence relation.

$T(n) = 2T(n/2) + \Theta(n)$

The above recurrence is similar to Merge Sort and can be solved either using Recurrence Tree method or Master method. It falls in case II of Master Method and solution of the recurrence is $\Theta(nLogn)$.

Πολυπλοκότητα

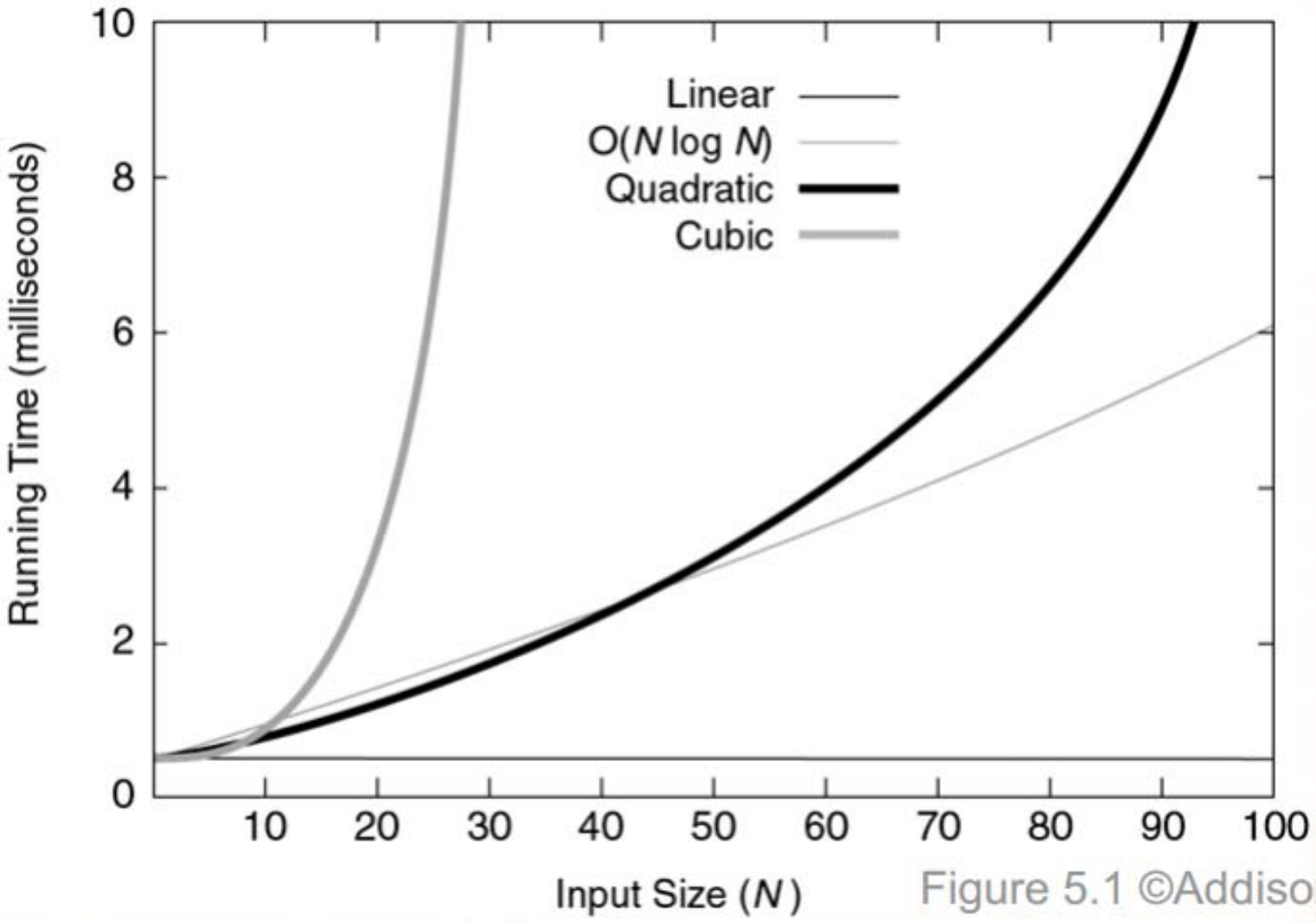

Figure 5.1 ©Addison Wesley