To: Professor Kearns
From: Anh Nguyen
Section: CPE 349 - 09
Assignment: Counting Inversions

Overview

The idea of solving this problem is to sort the array using merge sort. While merging 2 sub-arrays, we will count the number of inversions if the number from the right array is less than the number in the left array.

Pseudo Code:

1) Return 0 when the array has only 1 number.
2) Divide the array into 2 sub array
3) Recursively call the Inversions on those 2-sub array. 2T(n/2)
4) Merge 2 array together. While merging, update the count of inversions when the number from the right array is less than number from the left array. O(n)

The Recurrence Relation

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

Back Substitution

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

(substitute with $T(n/2) = 2T(n/4) + n/2$)

$T(n) = 2(2T(n/4) + n/2) + n$

$= 4T(n/4) + 2n$

(substitute with $T(n/4) = 2T(n/8) + n/4$)

$T(n) = 4(2T(n/8) + n/4) + 2n$

$= 8T(n/8) + 3n$

$\rightarrow$ general form: $2^k T(n/2^k) + kn$

When size n = 1, there is no need computation $\rightarrow T(1) = 0$

$\rightarrow T(n/2^k = 1) = 0$

$\rightarrow T(n = 2^k) = 0$

$\rightarrow T(\log_2 n = k) = 0$

Plug ($k = \log_2 n$) to the general form, we have

$2^{\log 2n} T(n/2^{\log 2n}) + (\log_2 n)*n$

$= n\,T(n/n) + n\log_2 n$

$= n\,T(1) + n\log_2 n$

$= n\log_2 n$

$\rightarrow O(n) = n\log_2 n$

Test Result of Running Inversions.java

- Test as in the assignment spec
  Input: {6, 4, 3, 1}
  Output: 6

  Input: {2, 3, 8, 6, 1}
  Output: 5
- Test of sorted array
  Input: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
  Output: 0
- Test of sorted array in decreasing order
  Input: {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
  Output: 45
- Test of array with all equal number
  Input: {3, 3, 3, 3}
  Output: 0