Teilaufgabe IV

Es sei A[0.. n - 1] ein Array von paarweise verschiedenen ganzen Zahlen.

Wir interessieren uns für die Zahl der Inversionen von A; das sind Paare von Indices (i, j), sodass i <j aber A[i] > A[j]. Die Inversionen im Array [2, 3, 8, 6, 1] sind (0, 4), da A[0] > A[4] und weiter (1, 4), (2, 3), (2, 4), (3, 4). Es gibt also 5 Inversionen.

(a) Wie viel Inversionen hat das Array [3, 7, 1, 4, 5, 9, 2]?

(b) Welches Array mit den Einträgen l, ..., 2 hat die meisten Inversionen, welches hat die wenigsten?

(c) Entwerfen Sie eine Prozedur int merge(int[ ]a, int i, int h, int j);

welche das Teilarray a[i.,j] sortiert und die Zahl der in ihm enthaltenen Inversionen zurückliefert, wobei die folgenden Vorbedingungen angenommen werden:

- OQ<i<h<j <n, wobei n die Länge von a ist (n = a.length).

- ali.h] und a[h + 1..,j] sind aufsteigend sortiert.

- Die Einträge von ali..j] sind paarweise verschieden. Ihre Prozedur soll in linearer Zeit, also O(j - i) laufen. Orientieren Sie sich bei Ihrer Lösung an der Mischoperation des bekannten Mergesort-Verfahrens.

(d) Entwerfen Sie nun ein Divide-and-Conquer-Verfahren zur Bestimmung der Zahl der Inversionen, indem Sie angelehnt an das Mergesort-Verfahren einen Algorithmus ZI beschreiben, der ein gegebenes Array in sortierter Form liefert und gleichzeitig dessen Inversionsanzahl berechnet.

Im Beispiel wäre also

ZI([2, 3, 8, 6, 1) = C1, 2, 3, 6, 8], 5)

Die Laufzeit Ihres Algorithmus auf einem Array der Größe n soll O(n log(n)) sein.

Sie dürfen die Hilfsprozedur merge aus dem vorherigen Aufgabenteil verwenden, auch, wenn Sie diese nicht gelöst haben.

(e) Begründen Sie, dass Ihr Algorithmus die Laufzeit O(n log(n)) hat.

(f) Geben Sie die Lösungen folgender asymptotischer Rekurrenzen (in O-Notation) an:

(g) T(n)=2 * T(n/2) + Ollog n)

(h) T(n) =2 * T(n/2) + O(n′)

(i) T(n) =3 * T(n/2) + O(n)