Algorithmen und Programmierung 3, WS 2003/2004 — 2. Übungsblatt

Abgabe bis Donnerstag, 6. November 2003

10. (4 Punkte) Was passiert bei *Sortieren durch Einfügen*, wenn die gegebene Folge bereits (a) aufsteigend oder (b) absteigend sortiert ist? (c) Was passiert, wenn alle Elemente gleich sind? Ist es möglich, dass man bloß $O(n)$ Zeit benötigt?

11. (2 Punkte) Beantworten Sie die vorige Frage für *Sortieren durch Auswahl.*

12. (0 Punkte) Nehmen wir an, dass die Eingabefolge *fast sortiert* ist. Das heißt, dass jedes Element in der Ausgangsreihenfolge höchstens k Stellen von der endgültigen Position in der sortierten Reihenfolge entfernt ist. Geben Sie eine Schranke für die Laufzeit von *Sortieren durch Einfügen* in Abhängigkeit von n und k an.

13. (0 Punkte) Nehmen wir an, wir haben ein Liste mit n sortierten Elementen, gefolgt von k Elementen in beliebiger Reihenfolge. Welchens Sortierverfahren empfehlen Sie zum Sortieren der Gesamtliste für

    (a) $k = \Theta(1)$,
    (b) $k = \Theta(\log n)$,
    (c) $k = \Theta(\sqrt{n})$?

14. (0 Punkte) Wie kann man *Sortieren durch Einfügen* so anpassen, dass es für Folgen, die absteigend oder fast absteigend sortiert sind, möglichst schnell läuft?

15. (3 Punkte) Geben Sie alle möglichen topologischen Sortierungen für folgende Eingabe an: $n = 6$ und $\{(6,4),(3,5),(4,1),(3,4),(5,4),(2,1),(2,6),(3,6)\}$.

16. (0 Punkte) Was passiert beim in der Vorlesung angegebenen Algorithmus für *Topologisches Sortieren*, wenn ein Paar (i,j) in der Eingabe mehrfach auftritt? Was passiert, wenn ein Paar (i,i) auftritt?

17. (0 Punkte) Untersuchen Sie verschiedene Möglichkeiten, wie man die Liste der freien Elemente beim topologischen Sortieren verwalten kann, im Hinblick auf ihre Effizienz. Kann man auf diese Liste auch gänzlich verzichten?

    Welche Variablen oder Felder im Programm aus der Vorlesung[1] könnte man ohne Nachteil einsparen?

18. (5 Punkte) Erweitern Sie den Algorithmus zum topologischen Sortieren aus der Vorlesung, sodass bei der Existenz eines Kreises nicht einfach mit einer Meldung abgebrochen wird, sondern auch ein Kreis (als „Beweis“) ausgegeben wird.

    Beschreiben Sie Ihren erweiterten Algorithmus auf der Ebene von Pseudo-Code. (Zum Beispiel können Sie eine Schleife “für alle Elemente x ...” schreiben.)

    Erklären Sie Ihren Algorithmus *in Worten*, ohne notwendigerweise auf Details der Implementierung einzugehen. (Sie müssen nur das erklären, was gegenüber dem ursprünglichen Algorithms neu ist. Ein Korrektheitsbeweis ist nicht verlangt.)

19. (6 Punkte) Schreiben Sie ein Java-Programm für die vorige Aufgabe. Sie können das Programm aus der Vorlesung[1] erweitern.

    Ihr Programm sollte nicht mehr als $O(m+n)$ zusätzliche Zeit und nicht mehr als $O(n)$ zusätzlichen Speicher brauchen.

[1] `http://www.inf.fu-berlin.de/~rote/Lere/2003-04-WS/Algorithmen+Programmierung3/TopoSort.java`