3.5/6

1a) Vergleiche von Insertion-Sort:
$\Theta(n^2)$ -

Vertauschungen von Insertion-Sort:
$\Theta(n^2)$ ✓

Vergleiche von Bubble-Sort:
$\Theta(n^2)$ ✓

Vertauschungen von Bubble-Sort:
$\Theta(n^2)$ ✓

Vergleiche von Selection-Sort:
$\Theta(n^2)$ ✓

Vertauschungen von Selection-Sort:
$\Theta(n)$ ✓

b) Vergleiche von Insertion-Sort:
$\Theta(n^2)$ t n

Vertauschungen von Insertion-Sort:
$\Theta(1)$ c n

Vergleiche von Bubble-Sort:
$\Theta(n^2)$ ✓

Vertauschungen von Bubble-Sort:
$\Theta(n)$ ✓

Vergleiche von Selection-Sort:
$\Theta(n^2)$ -

Vertauschungen von Selection-Sort:
$\Theta(n)$ ✓

Begründung fehlt

7 0/6 +1 Ausführung

1.2 a) quicksort (22, 23, 27, 3, 16, 10, 21)

partitioniere (22, 23, 27, 3, 16, 10) mit p = 21

= swap (22, 10) swap (23, 16) swap (27, 3)

⇒ (10, 16, 3, 27, 23, 22) füge p an der Stelle l ≥ r ein, 27 kommt ans Ende

⇒ (10, 16, 3, 21, 23, 22, 27)

quicksort (10, 16, 3)

partitioniere (10, 16) mit p = 3, keine swaps, füge p an der Stelle l ≥ r ein

⇒ (3, 16, 10)

quicksort (16, 10)

partitioniere (16) mit p = 10, füge p an der Stelle l ≥ r ein

⇒ (10, 16)

quicksort (23, 22, 27)

partitioniere (23, 22) mit p = 27

füge p an der Stelle l ≥ r ein

⇒ (23, 22, 27)

quicksort (23, 22)

partitioniere (23) mit p = 22, füge p an der Stelle l ≥ r ein ✓ sicher

⇒ (22, 23)

Das Array sieht nun so aus (3, 10, 16, 21, 22, 23, 27)

1.3 Es sei die Anordnung A der Zahlen 1..n mit A = n, ..., 2, 1 die Worst-Case Anordnung für alle Algorithmen, die paarweise vertauschen. Denn die erste (und größte) Zahl muss bis zum Ende „durchgeschoben" werden muss (n-viele Vergleiche & Vertauschungen)

6/6

Die nächste Zahl muss n-1-mal verglichen und vertauscht

1.2b) quicksort (23, 25, 26, 1, 3, 29, 30) swap (p, rechts)
partitioniere (30, 25, 26, 1, 3, 29) mit p = 23
swap (30, 3), swap (25, 1), füge p an der Stelle l≥r ein,
die Stelle l≥r (26) kommt ans Ende
⇒ (3, 1, 23, 25, 30, 29, 26)
quicksort (3, 1)
partitioniere (1) mit p=3, füge p an der Stelle l≥r ein
⇒ (1, 3)
quicksort (25, 30, 29, 26)
partitioniere (30, 29, 26) mit p=25, füge p an der Stelle
l≥r ein
⇒ (25, 29, 26, 30)
quicksort (29, 26, 30)
partitioniere (26, 30) mit p=29, füge p an der Stelle
l≥r ein
⇒ (26, 29, 30) sehr schön
Das Array sieht nun so aus: (1, 3, 23, 25, 26, 29, 30) ✓

1.3 Fortsetzung (sry dafür) kein Thema :)
werden. Dies wird sich (je nach Algorithmus) mindestens
bis zur Hälfte des Arrays so verhalten, weshalb sich
die Anzahl der Vergleiche und Vertauschungen als folgende
Summe darstellen lässt:
$\sum_{i=1}^{n/2} (n-i)$, was nichts anderes ist als die halbe Gauss-Summe
rückwärts: $\sum_{i=n/2}^{n} i$, was in Theta-Notation
$\Theta(n^2)$ entspricht $\Rightarrow \Omega(n^2)$ ✓

5/6

1.4 Vorweg: Es wird nicht durch die Aufgabenstellung ausgeschlossen, dass ein Kellner mehrfach zum selben Tisch gehen darf!

Deshalb darf jeder Kellner k_i zum Tisch t_i gehen, und $a(t_i)$ um $b(k_i)$ verringern, bis $a(t_i) < 1$ ist.

Algorithmus zur Zuordnung:

Initialisiere Tupel-Array [(Tisch, Kellner)] A der Länge n

Für i = 1 ... n

$A[i] = [(T_i, K_i)]$;

Laufzeit: $O(n) \Rightarrow o(n^2)$

Bemerkung: Jeder Kellner läuft in Gefahr, sehr oft laufen zu müssen, bis sein Tisch abgeräumt ist.

Formulierung ungünstig. Das Wort "Heap" hätte ich gern irgendwo gelesen :-

In der Klausur immer irgendeinen Tutor fragen