2.1 $(237)_{10} = (1422)_5$

8/8 $(66)_{10} = (231)_5$

$(554)_{10} = (4204)_5$

$(125)_{10} = (1000)_5$

$(16)_{10} = (31)_5$

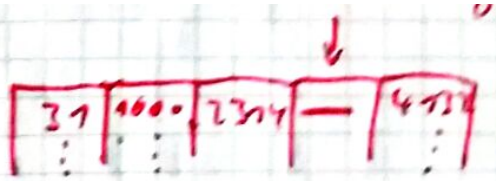
$(334)_{10} = (2374)_5$

$(58)_{10} = (213)_5$

$(542)_{10} = (4132)_5$

$(167)_{10} = (1132)_5$

$(148)_{10} = (1043)_5$ ✓



Update: Sorry, hast du gemacht.

Sammeln nach LSD, Phase 1/4

1000	231 31	1422 4132 1132	213 1043	4204 2314

Verteilphase 1/4

1000	231	31	1422	4132	1132	213	1043	4204	2314

Sammeln nach LSD, Phase 2/4

1000 4204	213 2314	1422	231 31 4132 1132	1043

✓

Verteilphase 2/4

1000	4204	213	2314	1422	231	31	4132	1132	1043

Sammeln nach LSD, Phase 3/4

1000 31 1043	4132 1132	4204 213 231	2314	1422

Verteilen 3/4

1000	31	1043	4132	1132	4204	213	231	2314	1422

Sammeln 4/4

31 213 231	1000 1043 1132 1422	2314 *	4132 4204

Verteilen 4/4

31	213	231	1000	1043	1132	1422	2314

4132	4204

Hier wurd's eng(e)
Aber gemeistert

2.3 a) Da Quicksort ein beliebiges Element a als Pivot-Element wählt, bietet es im Gegensatz zu Merge-Sort schon mal die Möglichkeit, überhaupt mit einem konkreten Element zu arbeiten.
Die Partitionier-Funktion packt alle Elemente, die kleiner als das Pivot-Element sind links vom Pivot-Element ins Array, alle größeren rechts davon. Das heißt, dass nach einmaliger Ausführung der Partitioniere-Funktion der Rang des Pivot-Elements feststeht, er ist gleich seiner Position im Array. O(Partitioniere(n)) = O(n)
Dieses Feature kann Merge-Sort nicht bieten, und überhaupt muss es erstmal $\log_2(n)$ Rekursionen starten, bevor es irgendwie anfängt mit Werten im Array zu arbeiten. ✓

b) Durchlaufe die einfach-verkettete Liste, und speichere jeden Wert in ein Array A, auf das dann Merge-Sort angewendet wird. Schreibe alles zurück in die Liste.
Initialisiere Array A mit Länge der einfach-verketteten Liste
Für i = 1... länge einfach-verkette Liste O(n)
    A[i] = *liste;
    liste ++;
merge(A) O(n · log(n))
Für i = 1... länge einfach-verkettete Liste O(n)
    *liste = A[i];
    liste ++;
Laufzeit: O(n) + O(n · log(n)) + O(n) = O(n · log(n)) ✓

2.3 c) Zum Beispiel ist der Algorithmus nicht stabil (= zwei gleiche Zahlen vertauschen ihre Reihenfolge) auf folgender Eingabe:

merge (1, 1, 2) mit $A = [1_1, 1_2]$

Es wird das Array B wie folgt befüllt:

$B = [1_1, 1_2]$

Es kommt zur Abfrage:

`A[k] = (B[i] < B[j]) ? B[i++] : B[j--];`

Die in diesem Fall so aussieht und evaluiert wird:

$A[1] = (B[1_1] < B[1_2])\ ?\ 1_1\ :\ 1_2$

$\Rightarrow A[1] = \text{false}\ ?\ 1_1\ :\ 1_2$

$\Rightarrow A[1] = 1_2$

... und A[2] wird dementsprechend auf 1_1 gesetzt, $A[1_1, 1_2]$ wird also zu $A[1_2, 1_1]$ "gemerged" ⇒ nicht stabil!

Dies ließe sich einfach beheben, indem man die Abfrage in der letzten Zeile durch ein "="-Zeichen wie folgt erweitert:

`A[k] = (B[i] <= B[j]) ? B[i++] : B[j--];` ✓

d) Folge der Länge 5: (1, 2, 3, 5, 4) ✓

Suche nach dem viert-kleinsten mit Quickselect verläuft so: quickselect (1, 2, 3, 5, 4), pivot = 1

partitioniere (2, 3, 5, 4), p = 1 ≤ 4 ⇒ 4 liegt im rechten Teilproblem

quickselect (2, 3, 5, 4)

partitioniere (3, 5, 4) p = 2 ≤ 4 ⇒ ... rechtes Teilproblem

quickselect (3, 5, 4)

partitioniere (5, 4) p = 3 ≤ 4 ⇒ rechts ...

quickselect (5,4)
partitioniere (4) $p = 5 \geq 4$ ⇒ viert-kleinstes liegt im linken Teilproblem

quickselect (4) ⇒ gefunden nach 5 Aufrufen mit jeweils um 1 schrumpfenden Teilproblemen.

2.2 a) Die Wahrscheinlichkeit dafür, dass das Pivot-Element im ersten Viertel liegt, ist gleich groß der WK, dass es im letzten Viertel liegt. Es genügt also, auszurechnen wie wahrscheinlich es ist, dass das Pivot-Element im ersten Viertel liegt, das dann verdoppeln und von 1 abziehen.

$$WK\left(\frac{n}{4} < r(\text{Pivot}) \leq \frac{3n}{4}\right) \overset{1-2\cdot}{:} \left(\sum_{k=4}^{7} \binom{7}{k} \cdot \left(\frac{1}{4}\right)^k \cdot \left(\frac{3}{4}\right)^{7-k}\right)$$

$$= 1 - 2\left(\frac{289}{4096}\right) = 0{,}858 \approx 86\% \checkmark$$

b) Die Wahrscheinlichkeit, dass das Pivot-Element außerhalb des Intervalls fällt ist die Gegenwahrscheinlichkeit zu der in a) berechneten Wahrscheinlichkeit.
Wie oft muss diese aufaddiert werden, bis sie größer 1 ist? $\frac{1}{2\left(\frac{289}{4096}\right)} = 7{,}086\ldots$ ✓ ⇒ Nach mindestens 8 mal ist zu erwarten, dass das Pivot-Element außerhalb des Intervalls fällt.

Super!