23. Bestimmen der Mehrheit, 10 Punkte

b) Sofern ein Mehrheitselement existiert, dann muss es auch in mindestens einer der Teilmengen existieren.
Ab hier rekursiv:

- Wenn Menge leer, brich ab.
- Wenn mehr als zwei Elemente vorhanden sind, dann entferne alle Elemente, die paarweise verschieden sind. Wenn ein Paar gleich ist, dann streiche nur ein Element.
- Wenn Menge nur noch aus einem Element besteht, gebe Element zurück.
- Halbiere Menge und suche in Teilmengen weiter.

Laufzeit

Da man jedes Element mindestens einmal ansieht $\rightarrow \Theta(n)$. Im schlimmsten Fall kommt dann nochmal $\log n$ dazu.

c) Mit dem Algorithmus aus b) geht es nicht.
Idee: für $\mathcal{O}(n)$
Liste durchlaufen, Histogramm(Wert, Anzahl) erstellen, Histogramm durchlaufen, wenn Wert Bedingung B erfüllt ausgeben

```
1  in: e[0,..,i]          ; Werte
2  var: i = 0             ; Laufvariable
3  array: hist = []       ; Leeres Array, mit 0 iniziert
4  var: n = len(a)        ; laenge von Eingabe
5  var: B = n*0.3       ; Abschlussbedingung
6  ; Histogramm befüllen
7  for i=0, i<n, i++:
8    ; eingelesener Wert entspricht der Position im Histogramm
9    hist[e[i]]++
10 ; Histogramm durchlaufen
11 for val in hist:
12   ; Wert erfüllt Bedingung
13   if val >= B:
14     ; Wert zur Ausgabe hinzufügen
15     out.append(val)
16 return out
```

Laufzeit

Argumentativ: $\mathcal{O}(c * n)$, mit $n =$ Länge der Eingabe.
Zeilen 1-5: konstante Laufzeit
Zeile 7-9: $\mathcal{O}(n)$
Zeile 11-15: $\mathcal{O}(l(hist))$

Obere und untere Schranke unterscheiden sich nicht wesentlich und sind abhängig in der Größe vom Histogramm. Allgemein reicht es allerdings aus sich die Eingabe nur einmal $\Theta(n)$ anzusehen. Im schlimmsten Fall (alles verschiedene Werte) beträgt die Laufzeit $\mathcal{O}(2n)$.
Dieser Algorithmus verlagert das Laufzeitproblem auf die verwendete Datenstruktur, die man für das Histogramm `hist` annimmt. Empfehlen würden sich eine (Hash-)Map, da sie fürs suchen und einfügen (amortisiert) nur jeweils $\mathcal{O}(1)$ benötigen. `B = n*0.3`.

25. Lokales Maximum in Bäumen, 10 Punkte

a) Ja. Tiefen oder Breitensuche. Dadurch traversiert man den Baum, wodurch man in linearer Zeit ein Maximum bestimmen kann. Alternativ nehme Vorgehen aus b) weil $\mathcal{O}(h) \in \mathcal{O}(n)$.

b) Man folgt nur dem größten Pfad im Baum. D.h. wenn sich zwei Kindknoten unterscheiden dann wählt man den Teilbaum mit den größeren Wert im Kindknoten. In diesem Teilbaum muss es dann ein lokales Maximum geben. Die Höhe h eines Baumes ist dabei als längster Weg im Baum definiert.