1. (15 Punkte) Zeichnen Sie für jede der folgenden potentiellen Inhalte des Feldes $A[1..12]$ den entsprechenden durch $A[]$ implizit dargestellten binären Baum. Klassifizieren Sie jeden der drei Fälle als Heap, Beinahe-Heap, oder nicht Heap. Illustrieren Sie für den Fall des Beinahe-Heaps eine Heapify Operation.

   (a) $\langle 23, 12, 7, 31, 5, 9, 6, 5, 1, 16, 3, 6\rangle$

   (b) $\langle 25, 18, 20, 12, 13, 14, 6, 5, 2, 10, 7, 4\rangle$

   (c) $\langle 9, 20, 19, 13, 18, 9, 3, 4, 12, 10, 2, 7\rangle$

2. (20 Punkte)
   Sei B ein binärer Beinahe-Heap der Höhe $h > 2$, d.h. der längste Pfad von einem Knoten in B zur Wurzel w von B besteht aus h Kanten. Die in der Vorlesung besprochene Methode SIFT-DOWN wandelt B in einen Heap um. Im schlechtesten Fall passieren dabei $2h$ Schlüsselvergleiche.

   Entwerfen Sie eine andere Methode, die einen Beinahe-Heap in einen Heap verwandelt, die aber auf jeden Fall mit weniger als $2h$ Schlüsselvergleichen auskommt.

   Wie gering können Sie die im schlechtesten Fall benötigte Anzahl von Schlüsselvergleichen machen?

   Je geringer diese Anzahl, desto mehr Punkte!

3. (20 Punkte)
   Im Rahmen von Heapsort haben wir in der Vorlesung eine implizite Darstellung von binären Bäumen in einem Feld $A[]$ besprochen: $A[1]$ stellt die Wurzel des Baums dar, und für den Knoten v (der in $A[v]$ gespeichert ist) stellt $2v$ das linke Kind dar und $2v + 1$ das rechte Kind.

   Betrachten wir nun einen alternativen Ansatz: $v = 1$ stellt wieder die Wurzel des Baumes dar, aber für Knoten v sei das linke Kind gegeben durch $v + t(v)$ und das rechte Kind gegeben durch $v + 2t(v)$. Dabei sei für ganze Zahl $i > 0$ der Funktionswert $t(i)$ die größte Zweierpotenz, die nicht größer als i ist. Also, $t(1) = 1,\ t(2) = t(3) = 2,\ t(4) = t(5) = t(6) = t(7) = 4$, etc.

   (a) Zeichnen Sie den entsprechenden binären Baum für die Zahlen 1 bis 15.

   (b) Beweisen Sie, dass bei dieser alternativen Methode für jedes $n > 0$ die Zahlen 1 bis n einen Baum darstellen?

   (c) Wie kann man bei dieser alternativen Methode für einen Knoten v, den Parentknoten $\mathit{parent}(v)$ berechnen?

   (d) Wie kann man bei dieser alternativen Methode feststellen, ob Knoten v ein Blatt im Baum der Knoten $1, \ldots, n$ bildet?

   (e) Kann man diese alternative für Heapsort verwenden? Ergäben sich wesentliche Änderungen? Gibt es Vorteile, Nachteile?

4. (15 Punkte)

   (a) Entwickeln Sie ein schleifenfreies Programm (in Pseudocode), das mit Hilfe von Vergleichsoperationen den zweitkleinsten der vier Eingabeschlüssel x_1, x_2, x_3, x_4 bestimmt, wobei Sie annehmen dürfen, dass die vier Schlüssel verschieden sind.

   (b) Geben Sie den ihrem Programm entsprechenden Entscheidungsbaum an.

   (c) Wie viele Vergleiche macht Ihr Programm im schlechtesten Fall, im besten Fall, im Durchschnitt (unter welcher probabilistischen Annahme?).

   (d) Ist die Annahme, die vier Schlüssel seien verschieden, wirklich hilfreich? Was müssen Sie ändern, damit sie nicht notwendig ist?