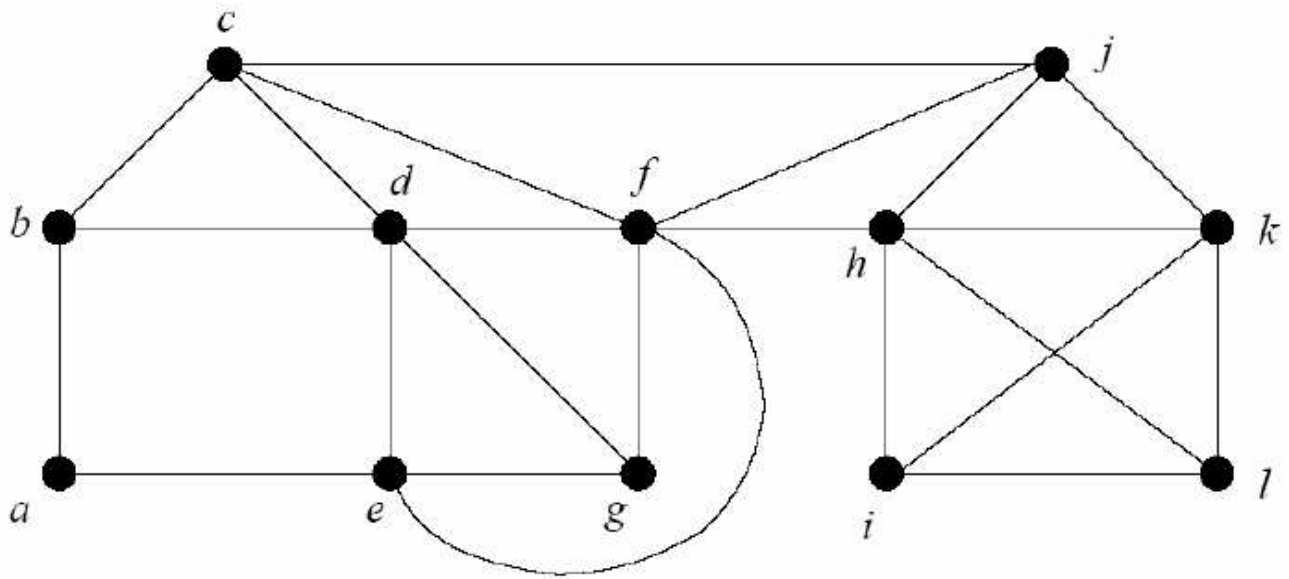
Aufgabe 1 (Tiefen- und Breitensuche)

Führen Sie beim folgenden Graphen eine Tiefen- bzw. Breitensuche durch. Starten Sie die Suche jeweils bei Knoten a. Die Adjazenzlisten sind alphabetisch sortiert.



Aufgabe 4 (Backtracking)

a) Schreiben Sie ein Programm in einer beliebigen Programmiersprache, welches unter Verwendung von Backtracking alle n - dimensionalen Vektoren mit Komponenten aus {0,1} erzeugt, welche die Eigenschaft besitzen, dass maximal drei aufeinander folgende Komponenten identisch sind. Beispiel für n=6: (0, 0, 0, 1, 1, 0) ist zulässig, (1, 0, 0, 0, 0, 1) dagegen nicht.

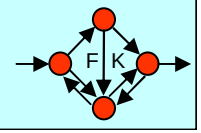
b) Schreiben Sie ein Programm in einer beliebigen Programmiersprache, welches unter Verwendung von Backtracking alle Permutationen der Zahlen (1, 2, ... , n) erzeugt, für die gilt, dass zwei benachbarte Komponenten sich um höchstens 2 voneinander unterscheiden. Beispiel für n=5: (1, 3, 5, 4, 2) ist zulässig, (1, 3, 4, 2, 5) dagegen nicht.

Aufgabe 5

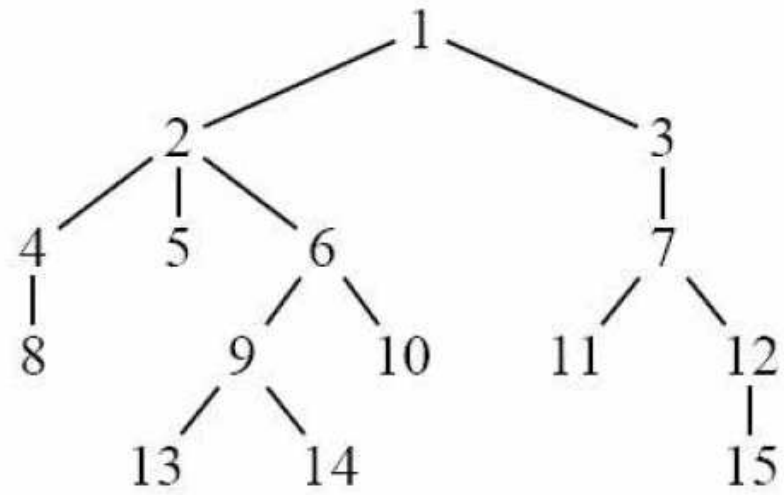
Mit Hilfe von 3 LKW mit Ladevermögen $t1$, $t2$, $t3$ Tonnen sollen t Tonnen ($t \leq t1 + t2 + t3$) der
(nicht partitionierbaren) Güter (mit den Gewichten $g1$, .., gk wobei $g1 + .. + gk = t$) einer Lagerhalle abtransportiert werden. Aus Sicherheitsgründen dürfen manche Güter nicht mit anderen Gütern gemeinsam auf einem LKW transportiert werden. Das Ladevermögen der LKW darf nicht überschritten werden. Wie können die Güter auf die 3 LKW verteilt werden?

Iterative Tiefensuche, wobei auf dem Stack Kanten liegen

```
Procedure Tiefensuche (S,Z:Knoten) is
begin
  push (S, S);    (* Speichere Knoten und Vorgänger (S hat keinen) *)
  while ( stack nicht leer ) do begin
    pop ( (x', x) );
    if (not besucht [x]) then begin
      besucht[x] := true;
      if X = Z then fertig;
      for alle y ist Nachbar von x do push (x, y);
    end if;
  end while;
end procedure;
```



Gegeben sei die grafische Darstellung eines Baums (mit Wurzel 1):



Geben Sie den Baum als Graph (*V,E*) an. Geben Sie die Tiefe des Baums an. Geben Sie die Blätter, und die inneren Knoten des Baums an. Geben Sie für jedes Blatt einen Pfad an, der von der Wurzel zu führt. Zeichnen Sie alle Teilbäume des Baums, die den Knoten 7 als Wurzel haben.

Die Länge des längsten Wegs von der Wurzel w zu einem Blatt bezeichnet man als die Tiefe (oder Höhe) des Baums.

3.7.3 Rekursive Definition für Bäume

Man kann Bäume auch rekursiv definieren:

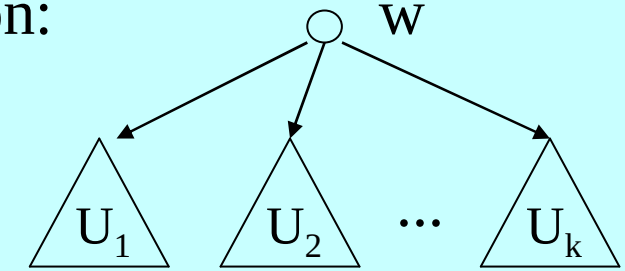
1) Die leere Menge ist ein Baum.

2) Wenn w ein Knoten und U eine endliche Menge von Bäumen sind, so ist auch $w(U)$ ein Baum.

w heißt Wurzel des Baums $w(U)$, die Elemente von U heißen Unterbäume oder Teilbäume von w im Baum $w(U)$.

Skizze: Leerer Baum:

Rekursion:



Hinweis: Ist die Menge der Bäume U = $\{U_1, ..., U_k\}$ geordnet, so spricht man von einem geordneten Baum.

Aufgabe 2 (Binärbäume)

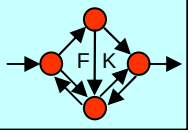
Unter der Höhe $h(b)$ eines (binären) Baumes verstehen wir folgendes:

> ein leerer Baum besitze die Höhe -1
> ein Baum b mit Wurzel w, linkem Unterbaum l und rechtem Unterbaum r besitzt die Höhe $h(b)=1+\max(h(l),h(r))$.

Schreiben Sie eine **function hoehe(*t* : tree) return integer**, die die Höhe des Baumes *t* nach obiger Definition berechnet.

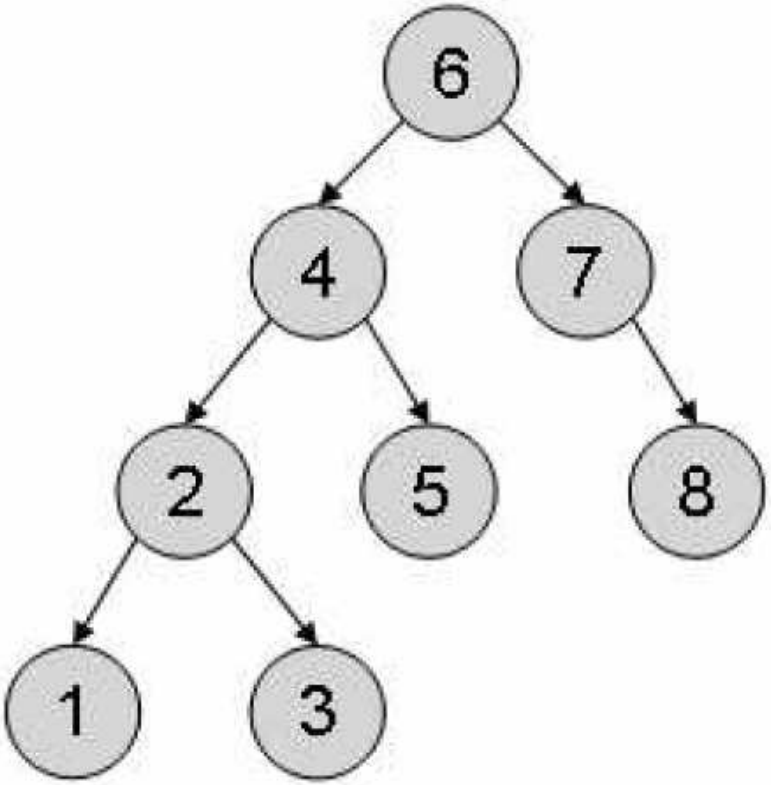
Unter der Tiefe eines Knotens *k* im (binären) Baum *b* verstehen wir die Anzahl der Kanten, die man minimal durchlaufen muss, um von der Wurzel zum Knoten zu gelangen. Die Wurzel eines Baumes hat danach die Tiefe 0. Unter einer Schicht eines Baumes verstehen wir die Menge der Knoten gleicher Tiefe. Unter der Dicke eines Baumes verstehen wir das Maximum der Kardinalitäten aller seiner Schichten.

Schreiben Sie eine **function dicke(t: tree): integer**, die die Dicke des Baumes t nach obiger Definition ermittelt.



Testen Sie Ihr Programm an den folgenden Bäumen.

Baum 1



Baum 2