Den Ablauf der rekursiven Aufrufe macht man sich am besten mit der folgenden Übersicht klar.

```
void findeBaum()
{
if (!kara.treeFront())
{
kara.move();
findeBaum();   void findeBaum()
               {
               if (!kara.treeFront())
               {
               kara.move();
               findeBaum();   void findeBaum()
                              {
                              if (!kara.treeFront())
                              { // entfaellt }
                              else { kara.turnLeft();
                              kara.turnleft(); }
                              } // Ende von findeBaum
               kara.move();
               }
               else { // entfaellt }
               } // Ende von findeBaum
kara.move();
}
else { // entfaellt }
} // Ende von findeBaum
```

Man kann es sich so vorstellen, dass sich der Computer bei jedem Aufruf der Methode merkt, wo er die Methode verlassen hat, um die gleiche Methode noch einmal abzuarbeiten. Nach der Abarbeitung macht der Computer dann jeweils an der Stelle weiter, die er sich beim Aufruf gemerkt hat.

1. Rekursive Methoden benötigen **unbedingt** an einer Stelle eine Abbruchbedingung. Wenn nicht, dann merkt sich der Computer so viele Stellen, an denen er eine Methode rekursiv verlassen hat, bis der Speicher voll ist.

2. Viele Probleme lassen sich rekursiv und nicht-rekursiv lösen. Ein rekursiver Ansatz bietet sich immer dann, wenn man ein gegebenes Problem schrittweise vereinfachen kann und nach einem Schritt im Prinzip das gleiche Problem (aber um einen Schritt einfacher) vorliegen hat.

3. Manchmal sind nicht-rekursive schneller als rekursive Lösungen. Oft sind sie aber nur sehr viel komplizierter zu programmieren. Meistens sind rekursive Lösungen eleganter formulierbar.