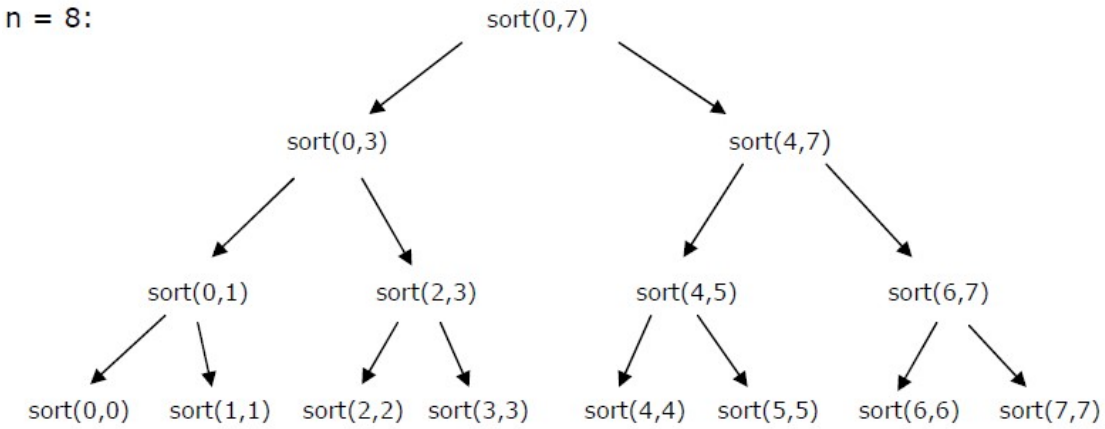
13. **MergeSort**

Gegeben ist das aus den acht Komponenten **a[0], a[1], ... , a[7]** bestehende array **a**, die gemäß beigefügtem Arbeitsblatt mit ganzen Zahlen belegt sind; das array soll schrittweise gemäß dem Algorithmus MergeSort aufsteigend sortiert werden.

Bemerkung:
Im folgenden schreiben wir sort(left,right) statt sort(a,left,right) und merge(left,middle,right) statt merge(a,left,middle,right).

Mit dem Aufruf **sort(a,0,7)** bzw. **sort(0,7)** wird der Vorgang zum Sortieren des aus 8 Komponenten bestehenden arrays **a** eingeleitet; dabei veranlaßt die rekursiv formulierte Funktion **sort** weitere Aufrufe von sich selbst gemäß folgendem Baumdiagramm:



Diese Baumstruktur ist auf der Seite 1 des beigefügten Arbeitsblatts nachempfunden. Nachdem das array a in Teillisten jeweils der Länge 1 zerlegt wurde (eine aus 1 Element bestehende Liste ist bereits sortiert), werden jeweils 2 sortierten Teillisten mit merge zu 1 sortierten Liste gemischt (Seite 2).

Aufgabe: In der beigefügten Übersicht MergeSort_Arbeitsblatt.doc wird der Sortiervorgang zum Sortieren eines aus 8 Komponenten bestehenden arrays schrittweise vollzogen; ergänze alle fehlenden Einträge in MergeSort_Arbeitsblatt.doc (oder handschriftlich in der ausgedruckten Version MergeSort_Arbeitsblatt.pdf).

14. **SelectionSort**

Der Algorithmus **sorting_by_direct_selection.py**
hat noch Optimierungspotential hinsichtlich des Zeitbedarfs zum Sortieren einer als array gegebenen Liste. Hierzu läßt sich die Funktion **`min(x,j)`** in geeigneter Weise modifizieren; ergreife diese Möglichkeit!