1 Hashtabelle

**Lösungsidee**Mithilfe der doppelten hashing Methode eine Map (hashtable) implementieren. Der Hashwert wird bei dieser Methode mithilfe zweier verschiedenen hash funktionen generiert. Bei der ersten Hash-funktion **h()** rechnen wir die Position mit **key % size** aus. Bei der zweiten Hash-funktion **p()** wird eine alternative Position mit **key % (size – 1)** dannach **+ 1** ausgerechnet, somit ist sie unabhängig von der ersten Hash-funktion. Der finale Hashwert wird dann bei jeder Iteration mit **(h() + index \* p()) % size** von der Map berechnet.  
Bei der Lineraren und Quadratischen Kollisionsstrategie wird statt **p()** einfach entweder **i** oder **i^2** verwendet.

**Code**Beiliegende .c und .h files.

**Testfälle**  
siehe main.c file.

2 Zusammenfügen 2er sortierter, einfach-verketteter Listen zu einer   
 neuen sortierten Liste

**Lösungsidee**Die Klassen SingleLinkedList, Node und Merger selber implemetieren. Bei der Merger Klasse gibt es eine **mergeSortedLists** funktion mit der man zwei Listen zusammen fügen kann. Dabei wird beim Merge zuerst mit einer While über beide Listen drüber iteriert bis man bei einer am Ende angekommen ist. Dannach beide einzeln vom currentNode weiterlaufen lassen falls eine größer war als die andere. Beide Listen müssen natürlich gesortet sein.

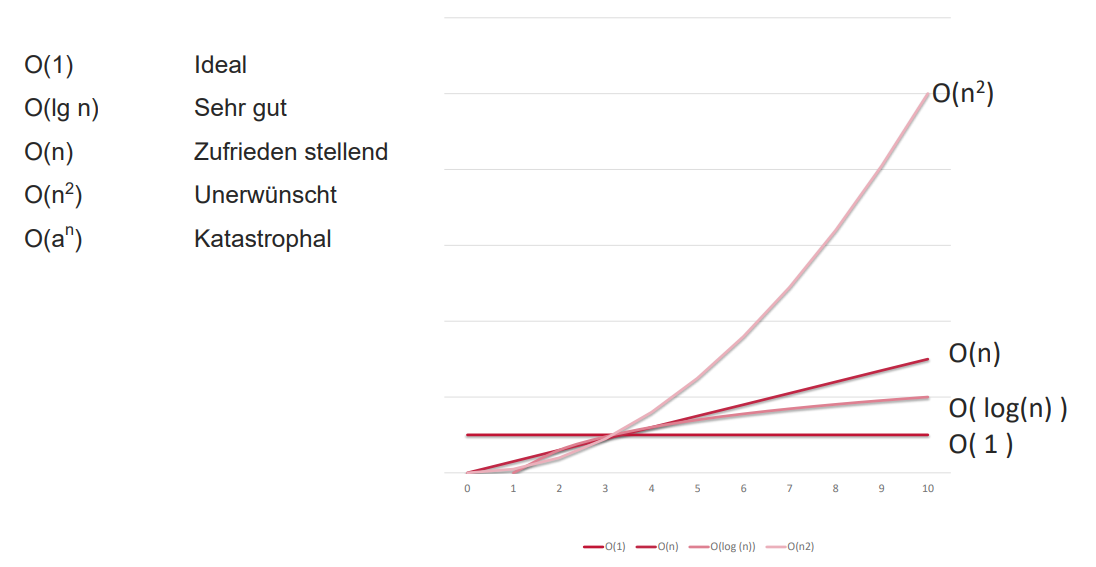
**Code**Beiliegendes MergeSortedLists java Projekt.

**Testfälle**  
siehe test package.

3 Laufzeitkomplexität

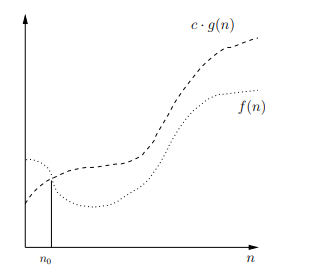
**a) Was beschreibt die O-Notation?**

Die O-Notation beschhreibt den Wachstum der Laufzeit in abhängigkeit der Problemgröße. Dabei wird immer der schlechteste Fall betrachtet. Dazu gibt es verschiedene Komplexitätsklassen, die im folgendem Graph beschrieben sind. Auf der x Achse haben wir die Problemgröße und auf der y Achse die Laufzeit.

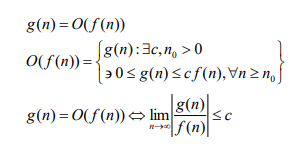
****

**Welche Bedingung ist für die asymptotische Laufzeitkomplexität von essentieller Bedeutung?**

Für die asysmtopische Laufzeitkomplexität ist es essentiell, dass diese erste ab einer grwissen Problemgröße gilt, da zuvor Konstate Operationen stärker ins Gewicht fallen können und somit kann es sein, dass bei einer kleineren Problemgröße ein Algorithmus mit O(log(n)) lansamer sein kann als ein Alorithmus mit O(n).

****

**b)** Erklären Sie die nachfolgende mathematische Definition der O-Notation



Die Betrachtung der Laufzeit nur in Bezug auf eine obere Schranke.  
Die Aussage gilt für alle n die >= n0 sind. Sprich erst bei einer gewissen Problemgröße.