Erweitern Sie Ihre Implementierung **ADS_set** um die globale Funktion
`bool z(const ADS_set& a, const ADS_set& b);`
Diese soll genau dann `true` liefern, wenn

- **a** und **b** beide leer sind, oder
- mehr als 90 % der Elemente von **a** auch in **b** enthalten sind und mehr als 90 % der Elemente von **b** auch in **a** enthalten sind.

Erlaubt ist nur das Verwenden der Methode **`ADS_set::count`**. Aufruf von anderen Methoden oder Funktionen insbesondere die Verwendung von Iteratoren (und damit auch die Verwendung einer range based for loop) ist nicht erlaubt.

Die Zeitkomplexität der Funktion **z** muss bei Hashing O(n) bzw. beim B+-Baum O(n logn) sein (n ist das Maximum der Größen der beteiligten Sets), die Speicherkomplexität O(1). Es ist beispielsweise nicht erlaubt, die Werte zu sortieren, auf ein eventuell vorhandenes Werte-Cache zuzugreifen, oder zusätzliche Felder mit einer nicht konstanten Größe zu verwenden.

Beispiele (übereinstimmende Werte sind fett dargestellt):

- z({**4**,**7**,**1**,**5**,3,**6**,**0**,**8**,**10**,**2**,**9**},{**10**,**7**,**1**,**4**,**8**,**2**,**5**,**6**,**0**,**9**}) liefert **true**
- z({**4**,**7**,**1**,**5**,3,**6**,**0**,**8**,**10**,**2**,**9**},{**10**,**7**,**1**,**4**,11,**8**,**2**,**5**,**6**,**0**,**9**}) liefert **true**
- z({4,**7**,1,5,3,6,0,8,10,2,9},{**7**}) liefert **false**

Eine mathematische Formulierung der Aufgabenstellung für jene, die diese Art der Beschreibung bevorzugen: Seien A bzw. B die Mengen der in **a** bzw. **b** gespeicherten Werte, dann liefert

$$z(a,b): \begin{cases} true, wenn\ |A \cap B| > 0.9 \max(|A|,|B|) \\ true, wenn\ |A| = |B| = 0 \\ false\ sonst \end{cases}$$

Tipp:

Sie können **z** als friend von ADS_set definieren oder eine Hilfsmethode schreiben, die von **z** aufgerufen wird und die Arbeit intern im **`ADS_set`** verrichtet. (Diese Methode dürfen Sie natürlich von **z** aufrufen).