

Übung 12

PK, Blatt 11, Blatt 12

Ergebnisse der Probeklausur

Nachbereitung Blatt 11

Häufigkeitszählung

Vorbereitung Blatt 12

Typklassen



- Ausgeteilt: 404
- Abgegeben: 378
- Korrigiert: 341 (bisher)

- A1 Multiple Choice
- A2 Binärarithmetik in Haskell
- A3 Sportliche Stadtteile in SQL
- A4 ADTs Keller, Array
- A5 Chomsky, Markov
- A6 Boole'sche Algebra
- A7 Prädikatenlogik
- A8 Lambda-Kalkül

(diverse)

(Kap. 1, 5)

(Kap. 2)

(Kap. 5, 6)

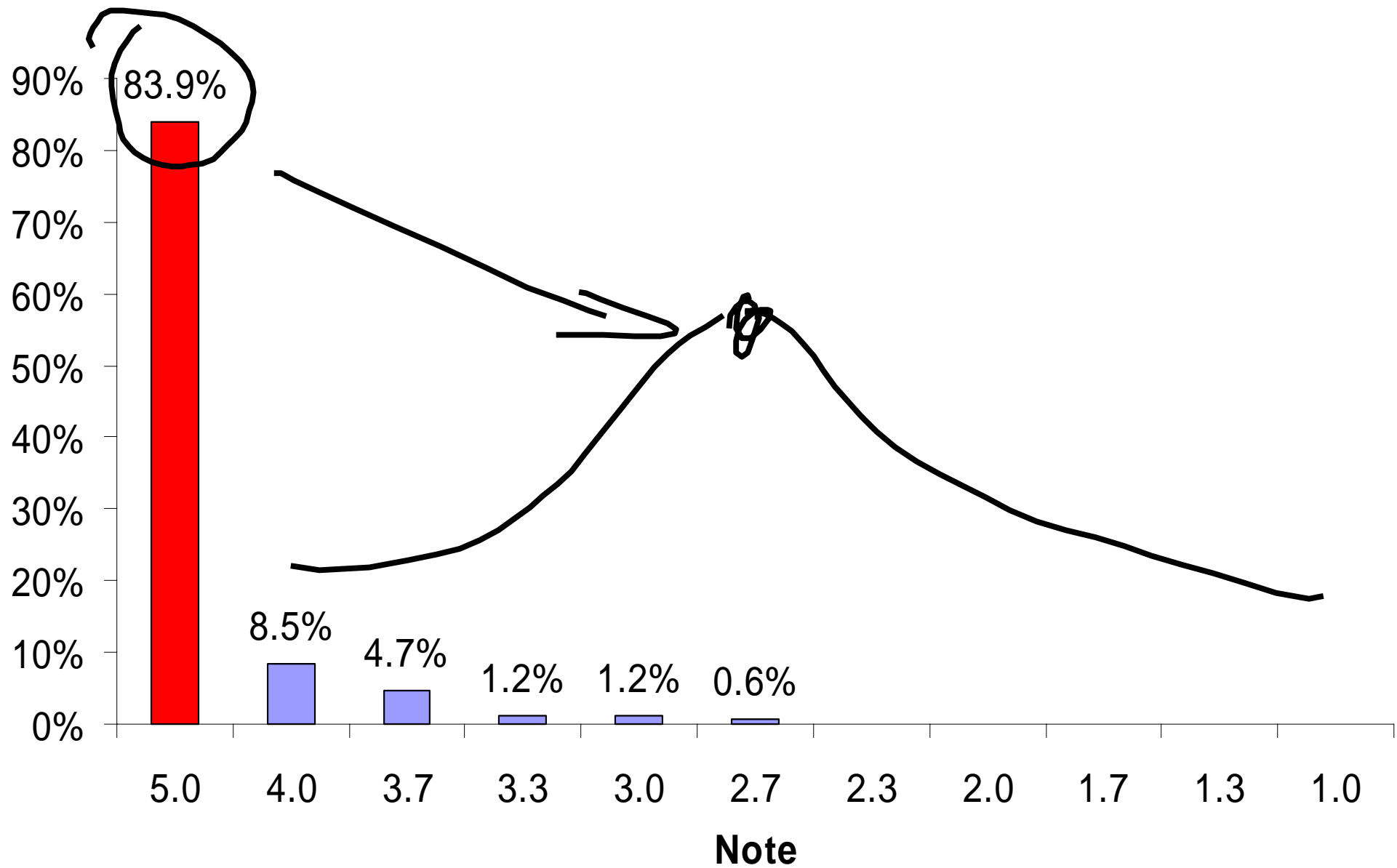
(Kap. 1)

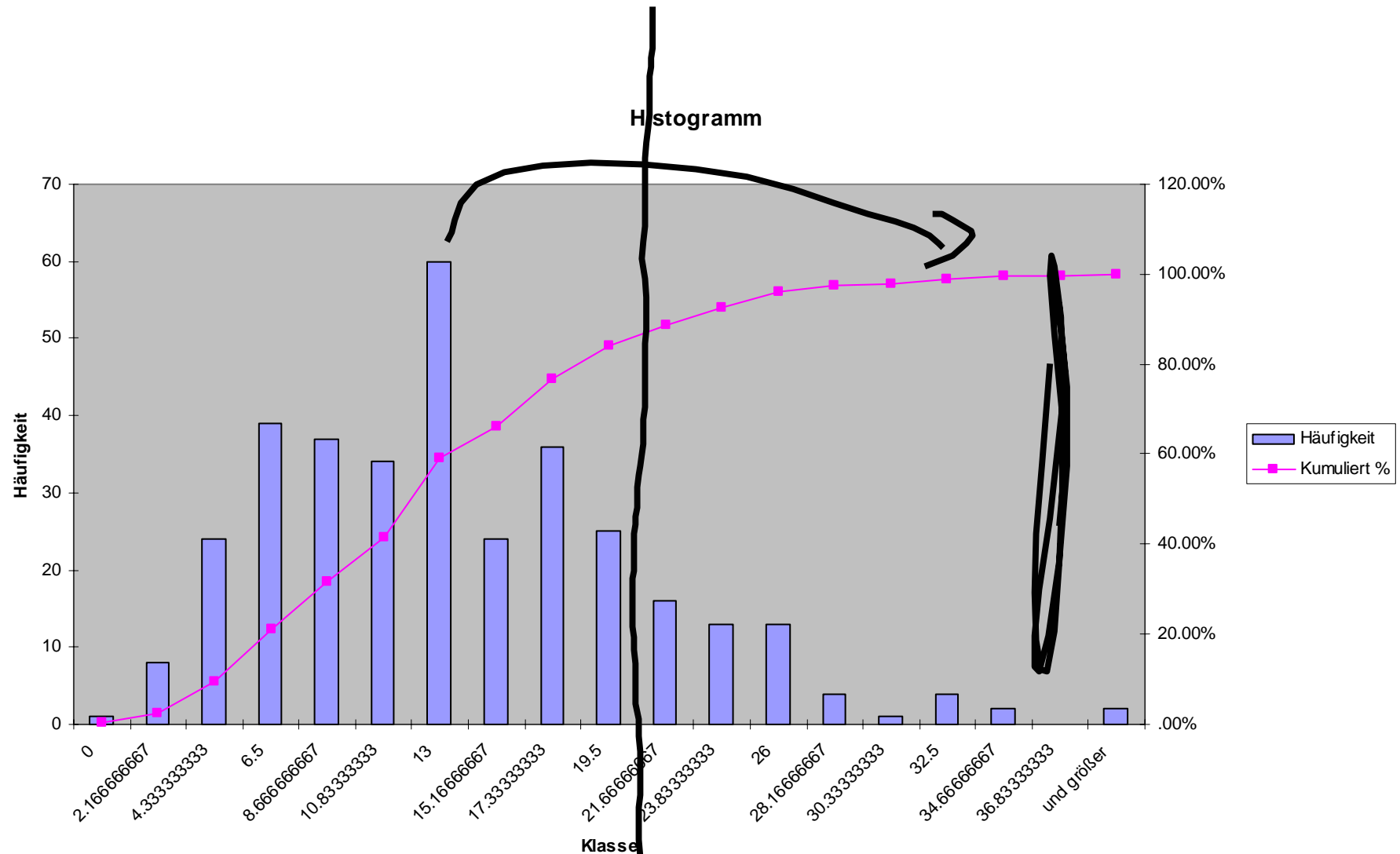
(Kap. 3)

(Kap. 4)

(Kap. 5)





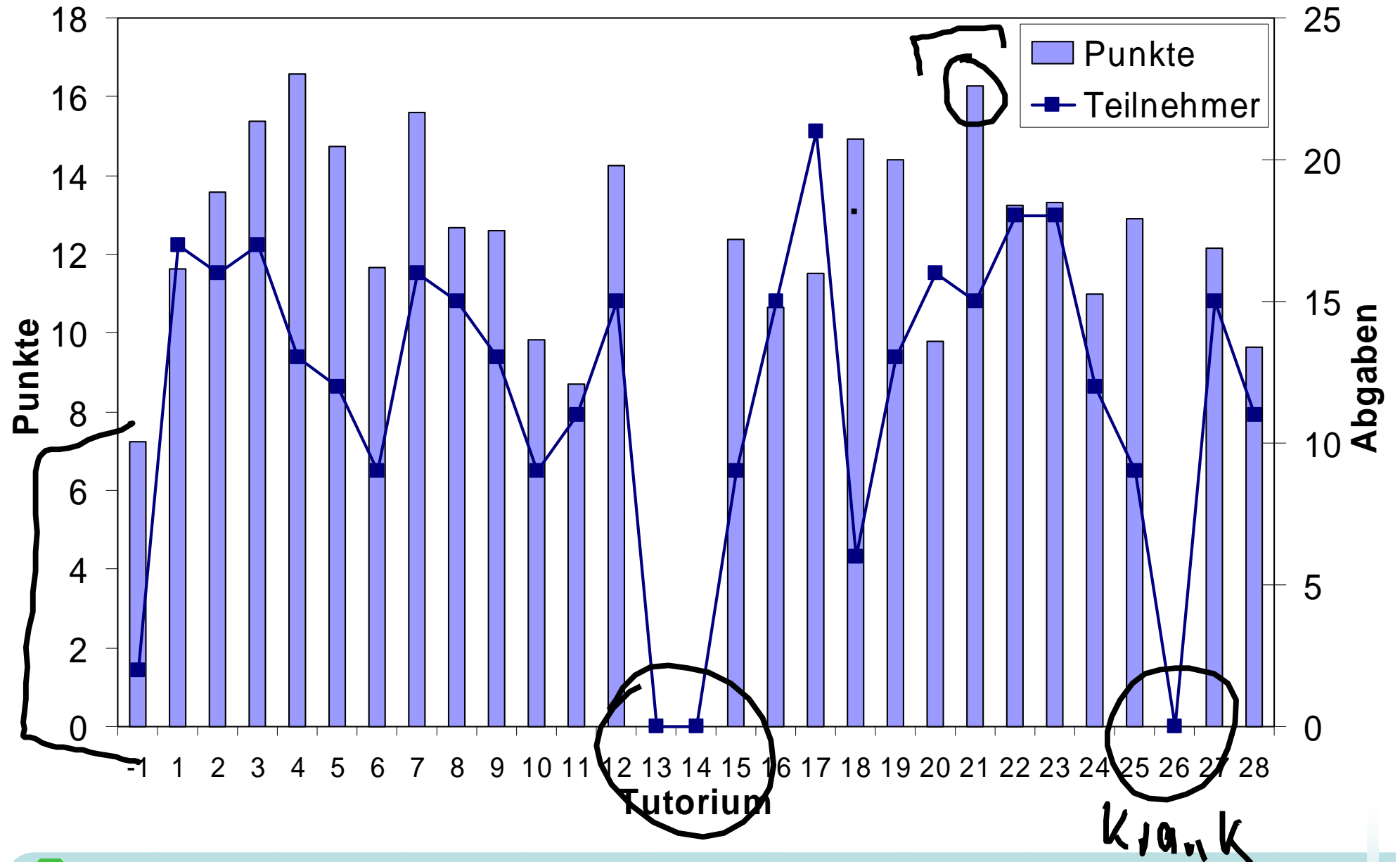


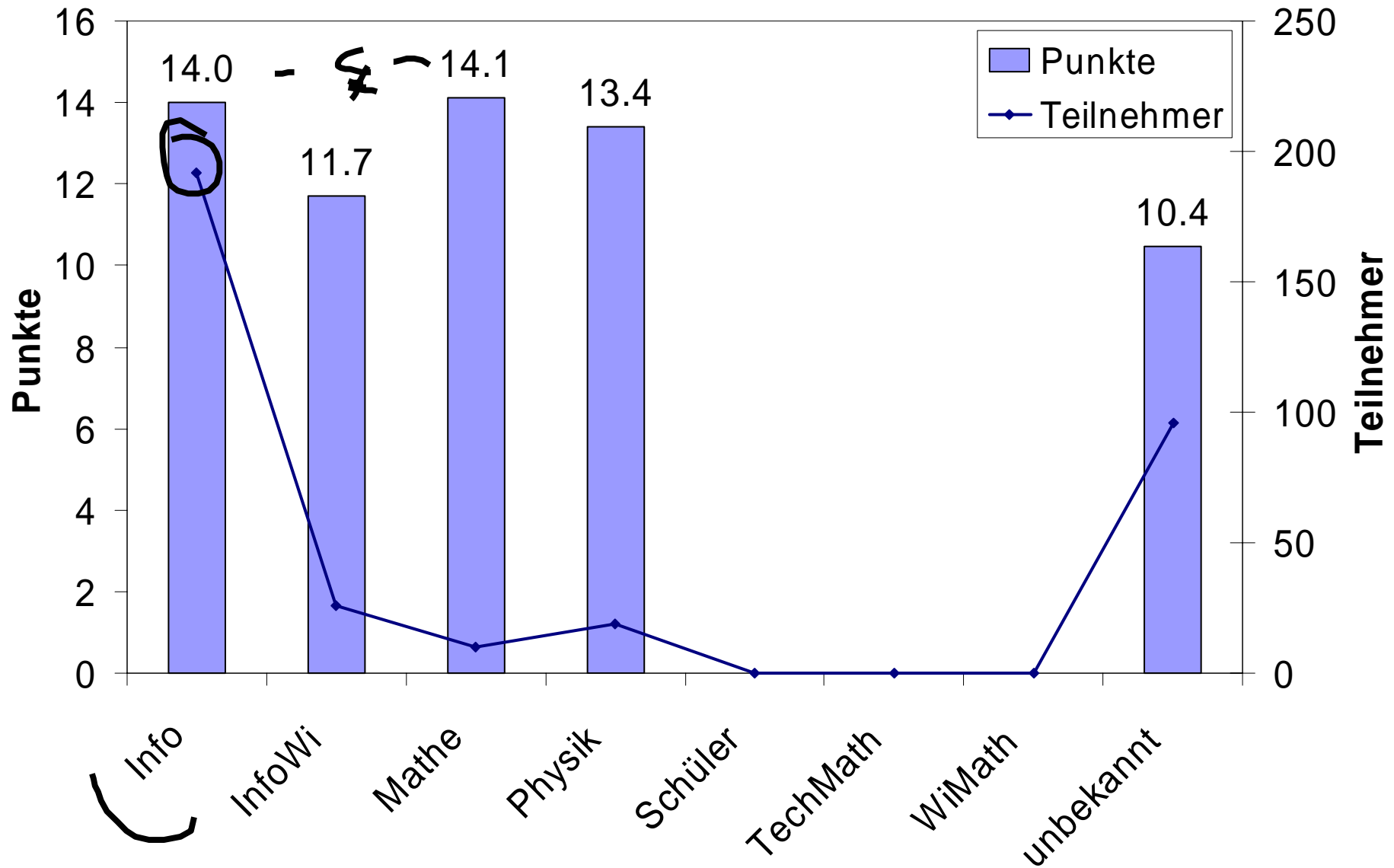
0

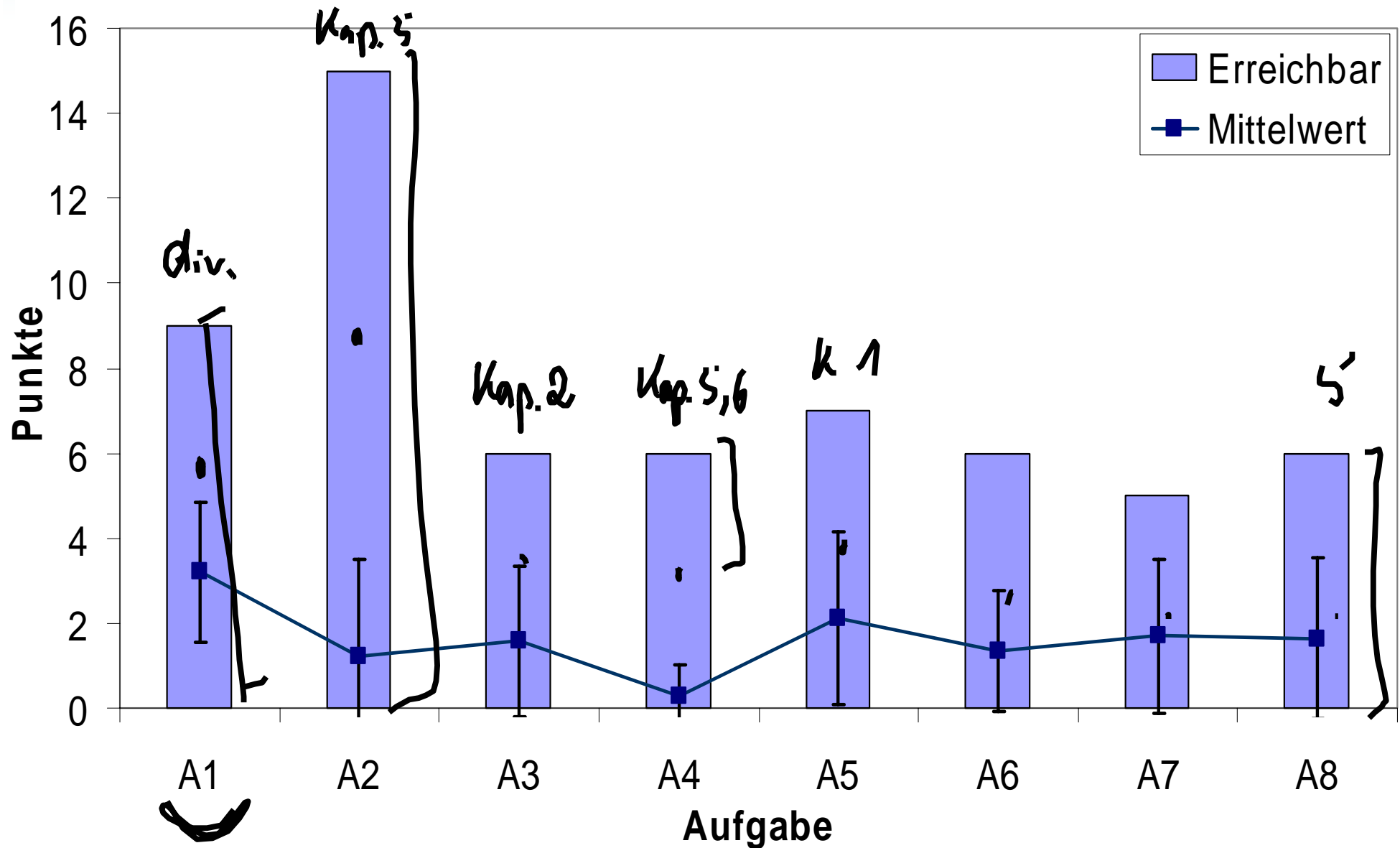
13

30









- Durchfallquote 83%, warum?
 - Tendenziell zu umfangreich
 - Teilnehmer inhaltlich unvorbereitet
 - Teilnehmer formell unvorbereitet

- Das soll in der 1. Klausur nicht passieren!

- Was könnt ihr tun?
 - Inhalte vorbereiten (ab jetzt)
 - Folien, Buch aufarbeiten
 - Übungsblätter **wirklich selbst** rechnen
 - Form einüben (letzte Vorbereitungswoche)
 - Geschwindigkeit und Aufgabentypen
 - Alte Klausuren rechnen



- Vor dem Hörsaal warten, bis Einlaß erfolgt
- Mobiltelefone **vor Betreten** des Saales ausschalten
- Jacken, Taschen, Papier etc. vorne ablegen
- Zulässig am Platz
 - Stifte,
 - Studentenausweis und
 - **sonst nichts!**
- Raum nicht vorzeitig verlassen.



- Vorlesezeit nutzen
 - Stift **nicht** anfassen
 - Aufgabentext verstehen, nicht überlesen
 - Aufgaben priorisieren
(absteigend nach Punkte/Zeit)
 - Erste Aufgabe **mental** lösen

- Bearbeitungszeit nutzen
 - Nicht festbeißen, im Zweifel zur nächsten Aufgabe
 - Nicht nervös werden
 - Kompakt und präzise schreiben
(`SELECT * FROM` ist für sich allein wertlos)



- Aus der Vorlesung sind Ihnen Häufigkeitszählungen auf Texten bekannt. Diese sind ein probates Mittel, um Verschlüsselungen durch Substitution und insbesondere die Caesar-Verschlüsselung zu brechen. Anstelle der absoluten Häufigkeit eines Buchstabens kommt dabei seine relative Häufigkeit zum Einsatz. Es gilt

$$\text{rel}_i = \text{abs}_i / \sum \text{rel}_j.$$

- Um eine Caesar-Verschlüsselung zu brechen, führt man eine Häufigkeitszählung auf dem Ciphertext aus. Diese wird mit einer Häufigkeitszählung für Plaintext derselben Sprache und Textgattung verglichen, um die Verschiebung des Alphabets zu ermitteln.



- Nachricht: JWKZWYNXYLJKFQQJS
ERFURTI STGEFALLEN



- Implementieren Sie absolute und relative Häufigkeitszählung für CryptoStrings in Haskell. Speichern Sie Häufigkeitsverteilungen dabei in Arrays. Ermitteln Sie die relative Häufigkeitsverteilung für den Text dieser Aufgabe (ausgenommen Ciphertexte).

- Argumentieren Sie: Welche Abstandsmaße sind auf relativen Häufigkeitsverteilungen sinnvoll? Wählen Sie Ihren Favoriten und implementieren Sie ihn in Haskell.

$$\sum_{b \in \mathcal{B}} (h(b) - h'(b))^2$$

Mean Squared Distance.

- Implementieren Sie eines der Sortierverfahren, die in der Vorlesung vorgestellt wurden. Ihre Implementierung muß mit einer Funktion parametrisierbar sein, die zwei Listenelemente vergleicht.



- Wir wollen feststellen, welche Verschiebung des Alphabets die relative Häufigkeitsverteilung eines Ciphertexts einer bekannten Verteilung am Ähnlichsten ist. Verwenden Sie Ihr Abstandsmaß und Ihre Sortierfunktion, um für eine gegebene Vergleichsverteilung die Liste der möglichen Verschiebungen nach absteigender Übereinstimmung zu sortieren. Wenden Sie diese Funktion auf die Häufigkeitsverteilung der Aufgabe und die folgenden Chiffre an.
- TGATEMXKMNF
- PBXOBXEXLMKTLLXGFNLLWXKFXGLVAXGMETGZLITSBXKXG



- Typklassen definieren eine abstrakte Schnittstelle

```
class Person a where  
  name      :: a -> String  
  adresse   :: a -> String
```

- Vererbung faßt Schnittstellen zusammen

```
class (Person a) => NatuerlichePerson a where  
  vorname   :: a -> String  
  geboren   :: a -> Datum
```

- Konkrete Klassen erklären sich konform zur Schnittstelle

```
data Mensch = EinMensch String String Datum String  
instance NatuerlichePerson Mensch where  
  name      (EinMensch x _ _ _) = x  
  vorname   (EinMensch _ x _ _) = x  
  geboren   (EinMensch _ _ x _) = x  
  adresse   (EinMensch _ _ _ x) = x
```



- Typklassen definieren eine abstrakte Schnittstelle

```
interface Person {  
    String getName();  
    String getAdresse();  
}
```

- Vererbung faßt Schnittstellen zusammen

```
interface NatuerlichePerson extends Person {  
    String getVorname();  
    Date    getGeboren();  
}
```

- Konkrete Klassen erklären sich konform zur Schnittstelle

```
class Mensch implements NatuerlichePerson {  
    String name, vorname, adresse;  
    Date    geboren;  
    String getName    () = { return name; }  
    String getVorname() = { return vorname; }  
    Date    getGeboren() = { return geboren; }  
    String getAdresse() = { return adresse; }  
}
```



- Mehrere Untertypen sind möglich

```
class (Person a) => JurPerson a where
  gruendung :: a -> Datum
```

```
class (JurPerson a) => GbRPerson a where
  numGesellsch :: a -> Int
  gesellschafter :: a -> Int -> Person
```

```
data GbR = (Person a) => EineGbR [a] String Datum
instance GbRPerson GbR where
  name          (EineGbR g _ _)
    = concat (concat (map name g), "-GbR")
  adresse       (EineGbR _ a _)    = a
  gruendung     (EineGbR _ _ g)    = g
  numGesellsch  (EineGbR g _ _)    = length g
  gesellschafter (EineGbR g _ _) i = g !! i
```



- Nehmen eine Anzahl Gegenstände auf
 - Gegenstände erfüllen eine Typschränke, d.h., stellen Gegenstandsfunktionen bereit



- Behälter als reiner Behälter
 - Zugriff auf Elemente, Elementzahl
- Behälter als Aggregat
 - Zugriff auf synthetisierte Informationen (berechnet aus Gegenstandsfunktionen)
- Behälter als Gegenstand derselben Art
 - Spezialfall von Behälter als Aggregat
 - Implementiert Gegenstandsfunktionen mit synthetisierter Information

