

Tutorium Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 13

Aufgabe 13.1

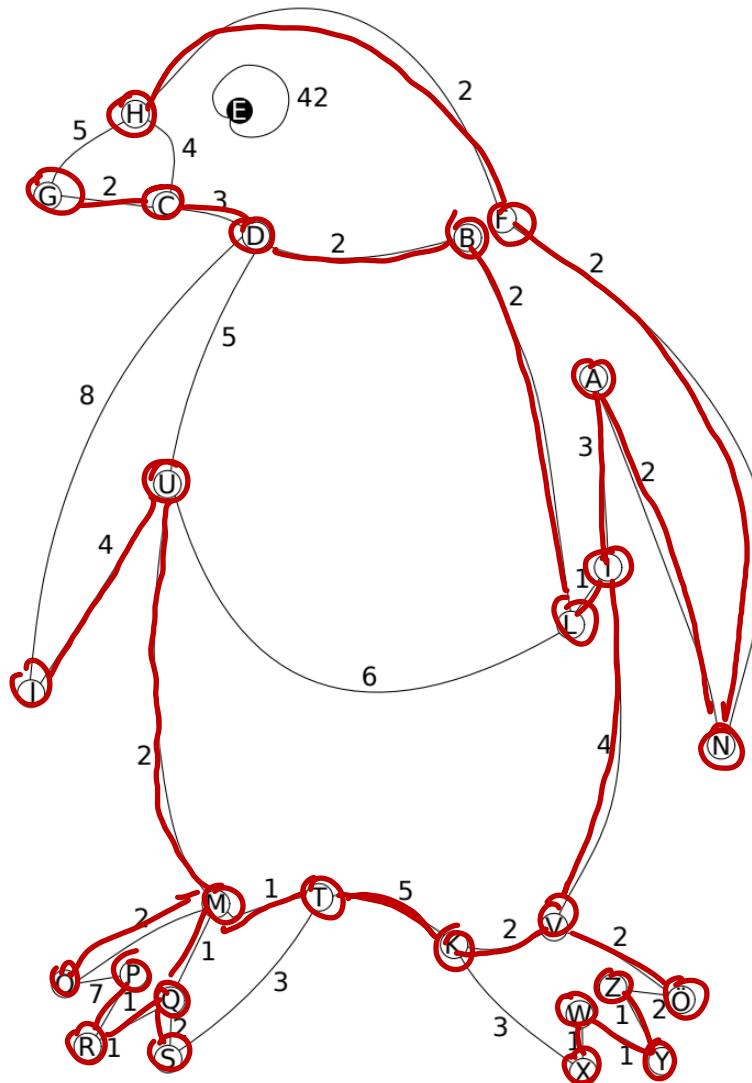
Spannbäume

Aufgabe 13.1

Gegeben sei der folgende Graph. Bearbeiten Sie mit diesem folgende Teilaufgaben:

- a) Benutzen Sie den Jarník-Prim-Algorithmus, um im nachfolgenden Graphen aus von Knoten A einen minimalen Spannbaum zu berechnen. Vervollständigen Sie dazu die Tabelle unter dem Graphen, die den Inhalt der Prioritätswarteschlange nach jedem Schritt darstellt. Die Warteschlange wird nach aufsteigender Priorität geordnet, bei mehreren Elementen mit gleicher Priorität wird alphabetisch nach Knotenbezeichnung sortiert (der Buchstabe "Ö" folgt auf "O"). In ein Kästchen ist jeweils die Knotenbezeichnung zusammen mit der Priorität einzutragen. Markieren Sie Ihren minimalen Spannbaum außerdem im Graphen.
- b) Wie kann man den Kruskal-Algorithmus so definieren, dass immer derselbe Spannbaum bei einem zusammenhängenden Graphen generiert wird, auch wenn der Graph z.B. nur Kanten mit demselben Gewicht enthält?
- c) Knobelaufgabe: Beweisen Sie, dass auch Jarník-Prim so angepasst werden kann, dass immer derselbe Spannbaum generiert wird, egal bei welchem Knoten man beginnt.

Aufgabe 13.1 (a)



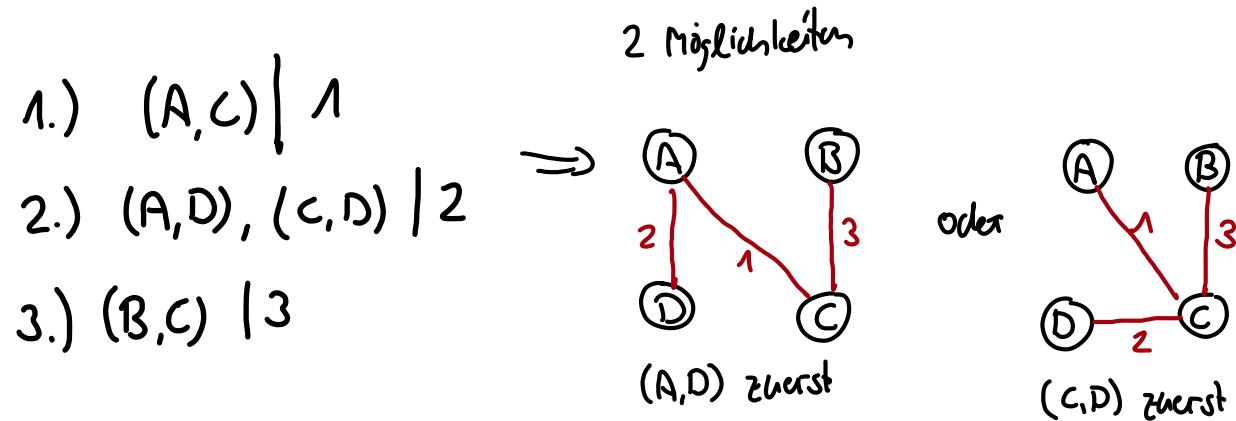
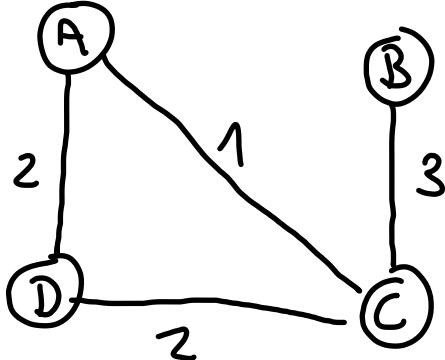
Schritt	Prioritätswarteschlange		
1	A	0	
2	N	2	I
3	F	2	I
4	H	2	I
5	I	3	C
6	L	1	C
7	B	2	C
8	D	2	C
9	C	3	V
10	G	2	V
11	V	4	U
12	K	2	O
13	Ö	2	X
14	Z	2	X
15	Y	1	X
16	U	1	X
17	X	1	T
18	T	5	U
19	M	1	S
20	Q	10	2
21	R	10	2
22	P	10	2
23	O	2	S
23	S	2	U
24	V	2	J
25	J	4	

Aufgabe 13.1 (b)

Wie kann man den Kruskal-Algorithmus so definieren, dass immer derselbe Spannbaum bei einem zusammenhängenden Graphen generiert wird, auch wenn der Graph z.B. nur Kanten mit demselben Gewicht enthält?

- Kruskal betrachtet Kanten aufsteigend nach Gewicht
↳ Der resultierende Spannbaum ist von der Sortierung von Kanten mit gleichem Gewicht abhängig.

Bsp.



Lösung: Sortierung für Kanten mit gleichem Gewicht festlegen (z.B. Alphabetisch nach Ziel- & Startknoten)

Aufgabe 13.1 (c)

Knobelaufgabe: Beweisen Sie, dass auch Jarník-Prim so angepasst werden kann, dass immer derselbe Spannbaum generiert wird, egal bei welchem Knoten man beginnt.

- Lässt auf selbes Problem wie in Aufgabe (b) hinaus.
 - ↳ An einem bestimmten Zeitpunkt werden 2 Teile des Spannbaums über eine von mehreren gleichgewichteten Kanten verknüpft
 - z.B. in (a) die Kanten (T, K) und (D, U) verbinden potentiell dieselben Teile des MST
 - ↳ Die Kante die in der Queue besser steht wird als erstes abgeschaut und bestimmt somit den MST.
 - ⇒ Schöne Lösung: Startknoten-unabhängige Sortierung von "Kanten" mit gleichen Gewichten
 - (Bei Jarník-Prim etwas umständlicher da eigentlich Knoten in der Queue stehen)

Aufgabe 13.2

Eure Fragen

Fragestellung 1

- a) Was ist die Laufzeit von $f(n)$ in Abhängigkeit von $n \in \mathbb{N}$, wenn gilt $g(n) \in \mathcal{O}(1)$
- b) Wie häufig wird $g(n)$ bei dem Aufruf $f(1, \text{false})$ insgesamt ausgeführt?
- c) Wie häufig wird $g(n)$ bei dem Aufruf $f(2, \text{false})$ insgesamt ausgeführt?
- d) Wie häufig wird $g(n)$ bei dem Aufruf $f(4, \text{false})$ insgesamt ausgeführt?
- e) Wie häufig wird $g(n)$ bei dem Aufruf $f(8, \text{false})$ insgesamt ausgeführt?
- f) Wie häufig wird $f(n)$ bei dem Aufruf $f(n, \text{false})$ mit $n \in \{2^k \mid k = 0, 1, \dots\}$ insgesamt (inklusive des ersten Aufrufs) ausgeführt? Vervollständigen sie den Ausdruck mit den fehlenden Koeffizienten. Es gilt $\sum_{k=0}^n x^k = \frac{x^{n+1}-1}{x-1}$.
- g) Gleiche Aufgabe wie f) nur für $g(n)$

```
void f(int n, boolean b) {  
    g(n);  
    if (n > 1) {  
        f(n / 2, true);  
        f(n / 2, b);  
    }  
    if(b) {  
        g(n);  
    }  
}
```

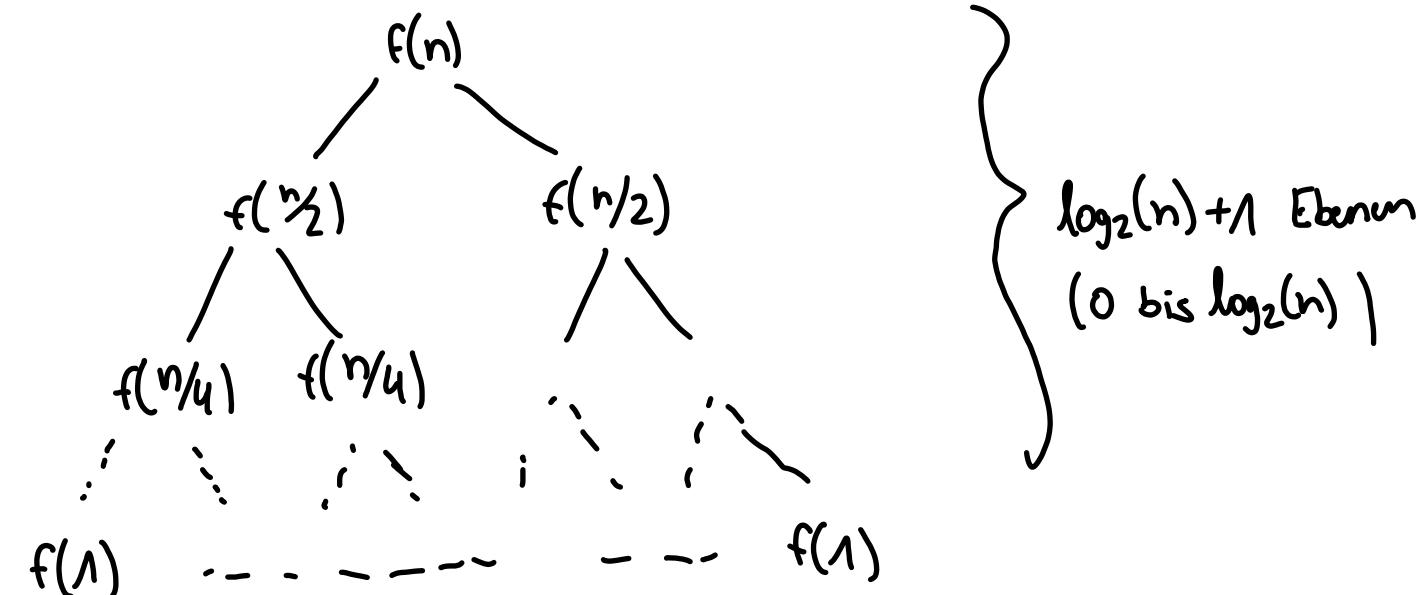
Funktion $g(\text{int } n)$ ist gegeben

___ $2^n +$ ___ $n^2 +$ ___ $n -$ ___ $\log_2(n) -$ ___

Fragestellung 1 (a)

Was ist die Laufzeit von $f(n)$ in Abhangigkeit von $n \in \mathbb{N}$, wenn gilt
 $g(n) \in \mathcal{O}(1)$

Calltree von f mit initislem Aufruf $f(n)$:
(boolean Argument nicht nötig da die rekursiven Aufrufe
von f nicht davon abhängen)



```
void f(int n, boolean b) {  
    g(n);  
    if (n > 1) {  
        f(n / 2, true);  
        f(n / 2, b);  
    }  
    if(b) {  
        g(n);  
    }  
}
```

Funktion $g(\text{int } n)$ ist gegeben

i-te Ebene hat 2^i Aufrufe von f

$$\Rightarrow \sum_{i=0}^{\log_2(n)} 2^i = 2^{\log_2(n)+1} - 1 = 2n - 1 \text{ Calls}$$

\Rightarrow Jeder Aufruf hat konstante Zeit da $g \in O(1)$

$\Rightarrow f(n) \in O(2n-1) = \boxed{O(n)}$

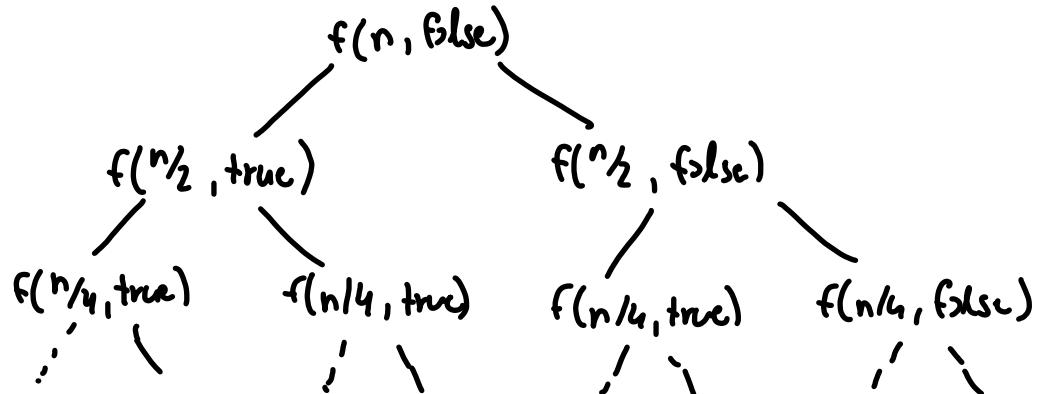
Fragestellung 1 (b-e)

Wie häufig wird $g(n)$ bei dem Aufruf

- b) $f(1, \text{false}) \rightarrow 1$
- c) $f(2, \text{false}) \rightarrow 4$
- d) $f(4, \text{false}) \rightarrow 11$
- e) $f(8, \text{false}) \rightarrow 26$

insgesamt ausgeführt?

- Betrachte erneut Calltree von f
↳ jetzt mit booleosn



16.07.2024

```

void f(int n, boolean b) {
    g(n);
    if (n > 1) {
        f(n / 2, true);
        f(n / 2, b);
    }
    if(b) {
        g(n);
    }
}
  
```

Funktion $g(\text{int } n)$ ist gegeben

- ⇒ Anzahl f-Calls nur von n abhängig (immer noch 2^{n-1})
- ⇒ Nur ein Pfad von Wurzel zu Blatt hat false als Argument, alle anderen Aufrufe true.
- ⇒ Ein false Aufruf verursacht einen Call von g .
- ⇒ Ein true Aufruf verursacht 2 Calls von g .
- ⇒ Zähle 2 Calls für jeden f-Call und subtrahiere 1 für jeden false - Call.
- ⇒ $2 \cdot (2^{n-1}) - (\log_2(n) + 1) = 4n - \log_2(n) - 3$ → für 1/2/4/8 berechnen

Tobias Eppacher

10

Fragestellung 1 (f-g)

- f) Wie häufig wird $f(n)$ bei dem Aufruf $f(n, \text{false})$ mit $n \in \{2^k \mid k = 0, 1, \dots\}$ insgesamt (inklusive des ersten Aufrufs) ausgeführt? Vervollständigen sie den Ausdruck mit den fehlenden Koeffizienten. Es gilt $\sum_{k=0}^n x^k = \frac{x^{n+1}-1}{x-1}$.
- g) Gleiche Aufgabe wie f) nur für $g(n)$

- Verwende Formeln, die in den vorigen Aufgaben hergeleitet wurden.

```
void f(int n, boolean b) {  
    g(n);  
    if (n > 1) {  
        f(n / 2, true);  
        f(n / 2, b);  
    }  
    if(b) {  
        g(n);  
    }  
}
```

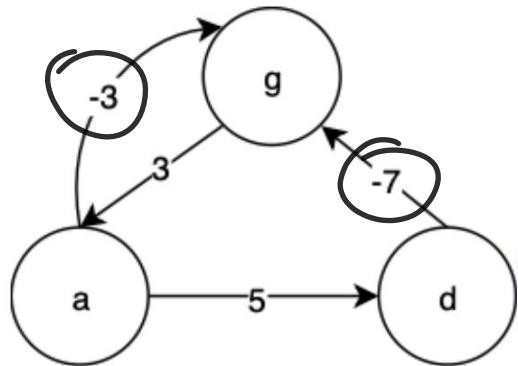
Funktion $g(\text{int } n)$ ist gegeben

$$\underline{0} 2^n + \underline{0} n^2 + \underline{2} n - \underline{0} \log_2(n) - \underline{1}$$

$$\underline{0} 2^n + \underline{0} n^2 + \underline{4} n - \underline{1} \log_2(n) - \underline{3}$$

Fragestellung 2

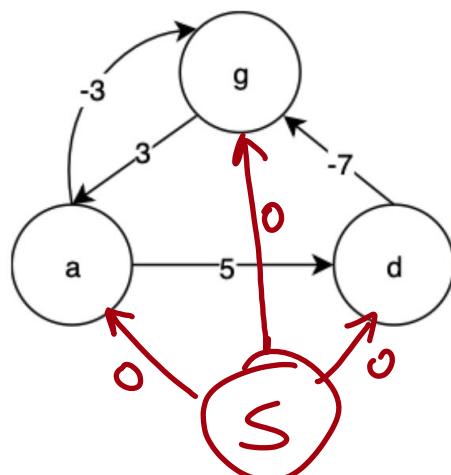
Gegeben ist der folgende Graph G_3



- a) Warum lässt sich das APSP-Problem auf diesem Graphen G_3 nicht mittels des *Dijkstra-Algorithmus* lösen? \rightarrow Nein, weil neg. Kantengewichte
- b) Als erster Schritt des Johnson-Algorithmus zur Lösung des APSP-Problems muss der Graph G_3 zuerst in einen Graphen mit nicht-negativen Kantengewichten transformiert werden.
Berechnen Sie die dazu nötigen Potenziale $h(v)$ für alle Knoten $v \in G_3$ unter Verwendung des Hilfsknotens s . Geben Sie ausreichend Zwischenschritte an. Zeichnen Sie anschließend den transformierten Graphen mit den nicht-negativen Kantengewichten.

Fragestellung 2 (b)

Als erster Schritt des Johnson-Algorithmus zur Lösung des APSP-Problems muss der Graph G_3 zuerst in einen Graphen mit nicht-negativen Kantengewichten transformiert werden. Berechnen Sie die dazu nötigen Potenziale $h(v)$ für alle Knoten $v \in G_3$ unter Verwendung des Hilfsknotens s . Geben Sie ausreichend Zwischenschritte an. Zeichnen Sie anschließend den transformierten Graphen mit den nicht-negativen Kantengewichten.



- Hilfsknoten s mit 0-Kanten zu allen anderen Knoten erstellen
⇒ Berechne Knotenpotentiale als kürzesten Pfad von s zum jeweiligen Knoten
(mit Bellman-Ford)

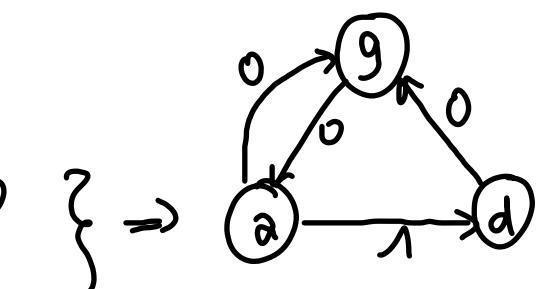
$$\text{Hier } h(a) = -4 \quad h(d) = 0 \quad h(g) = -7$$

⇒ Update Kantengewichte im ursprünglichen Graphen

$$d'(v,w) = d(v,w) - h(w) + h(v)$$

$$\begin{aligned} \Rightarrow d'(a,d) &\rightarrow 5 - 0 + (-4) = 1 & d'(a,g) &= -3 - (-7) + (-4) = 0 \\ d'(d,g) &\rightarrow -7 - (-7) + 0 = 0 & d'(g,a) &= 3 - (-4) + (-7) = 0 \end{aligned}$$

Funktioniert nicht mit neg. Kreisen



Fragestellung 3 (a)

Gegeben sei die Familie von Hashfunktionen

$$h_a(x) \left\lfloor \frac{(a \cdot x) \bmod 16}{2^{4-k}} \right\rfloor$$

wobei $a \in \mathbb{N}$, $k \in \{1, 2, 3, 4\}$ und die floor-Operation $\lfloor \cdot \rfloor$ auf ganze Zahlen abrundet.

Angenommen für ein Paar von beliebigen Werten $(x, y)_{x \neq y}$ gilt $h_a(x) = h_a(y)$.

Wir betrachten den Ausdruck $z = (a \cdot (x - y)) \bmod 16$ in Binärdarstellung.

Welche Aussage können Sie über die höchsten k Bits von z treffen, wenn mindestens ein anders

Bit von z gesetzt ist? (Nur eine Aussage richtig)

- Identische zu den niedrigsten k Bits
- Abwechselnd 0 und 1
- Identisch zu den höchsten k Bits von $(a \cdot x) \bmod 16$
- Identisch zu den höchsten k Bits von $(a \cdot y) \bmod 16$
- Alle 0
- Alle gleich
- Alle 1
- Gleichverteilt

• $h_a(x)$ gibt die obersten k Bits der 4-Bit Zahl $(a \cdot x) \bmod 16$ aus
→ Für x, y mit $h_a(x) = h_a(y)$

haben $(a \cdot x) \bmod 16$ und $(a \cdot y) \bmod 16$ die gleichen
obersten k Bits

$$\Rightarrow z = (a \cdot (x - y)) \bmod 16 = (a \cdot x - a \cdot y) \bmod 16 = \underbrace{((a \cdot x) \bmod 16 - (a \cdot y) \bmod 16)}_{\text{Subtraktion von 4 Bit Zahlen mit gleichen}} \bmod 16$$

obersten k -Bits

⇒ Oberste k Bits subtrahieren sich zu 0 → außer Subtraktion carried 1 von tieferen Bits
⇒ dann oberste k Bits gleich 1

Fragestellung 3 (b)

Gegeben sei die Familie von Hashfunktionen

$$h_a(x) \left\lfloor \frac{(a \cdot x) \bmod 16}{2^{4-k}} \right\rfloor$$

Für c -universelle Hashfamilien \mathcal{H} gilt

$$\forall (x, y)_{x \neq y}: \Pr\{h(x) = h(y)\} \leq \frac{c}{m}$$

wenn h zufällig aus \mathcal{H} gewählt wird. Sei $m = 2^k$.

Wir untersuchen nun die Universalität der Hashfamilie $\tilde{\mathcal{H}} = \{h_a \mid 0 < a < 16 \text{ und } a \text{ ungerade}\}$.

Benutzen Sie die Lösung aus dem vorigen Aufgabenteil um das kleinste c zu bestimmen, für das die obige Bedingung für $\tilde{\mathcal{H}}$ im Allgemeinen erfüllt ist.

- 1) $\left\{ \begin{array}{l} \Rightarrow a \dots \text{vier-Bit Zahl, unterste Stelle 1, andere gleichverteilt} \\ \Rightarrow \text{höherer Stellen von } z \text{ gleichverteilt, da } z \text{ ein Produkt von 2 mit einer ungeraden Zahl ist} \end{array} \right.$
- 2) $\left\{ \begin{array}{l} \Rightarrow \text{Aus vorheriger Aufgabe folgt } h(x) = h(y) \text{ wenn oberste } k \text{ Bits von } z \text{ gleich sind} \\ \Rightarrow \Pr\{h(x) = h(y)\} = \frac{2}{2^k} \text{ (folgt zur gleichverteilung)} = \frac{2}{m} \xrightarrow{\text{einsetzen}} \frac{2}{m} \leq \frac{c}{m} \Rightarrow c \geq 2 \Rightarrow c \underline{\underline{= 2}} \end{array} \right.$