Strong Randomness Properties of (Hyper-)Graphs Generated by Simple Hash Functions

Martin Aumüller

Technische Universität Ilmenau, Germany

AofA'15
Strobl, June 8, 2015

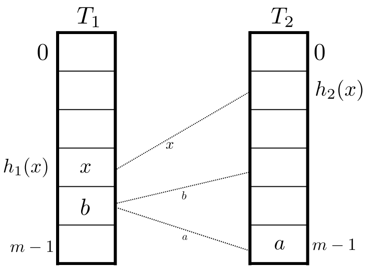
Joint work with Martin Dietzfelbinger and Philipp Woelfel.

Example: Cuckoo Hashing (Pagh/Rodler, 2001/2004)

A hashing-based implementation of the dictionary data type.

Setting:

- set $S \subseteq U$ of n keys
- two tables $T_1[0..m-1]$ and $T_2[0..m-1]$, $m \geq (1+\varepsilon)n$
- two (hash) functions h_1, h_2 with $h_i \colon U \to [m]$



Rules:

- each table cell can hold exactly one key
- a key x must be stored either in $T_1[h_1(x)]$ or $T_2[h_2(x)]$ (fast lookups and deletions!)

Definition

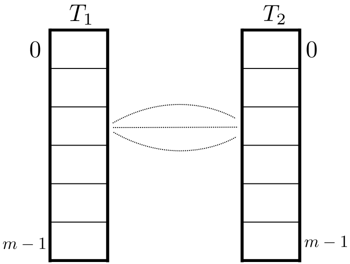
If S can be stored according to these rules, we call (h_1, h_2) **suitable for** S.

Example: Cuckoo Hashing (Pagh/Rodler, 2001/2004)

A hashing-based implementation of the dictionary data type.

Setting:

- set $S \subseteq U$ of n keys
- two tables $T_1[0..m-1]$ and $T_2[0..m-1]$, $m \geq (1+\varepsilon)n$
- two (hash) functions h_1, h_2 with $h_i \colon U \to [m]$



Rules:

- each table cell can hold exactly one key
- a key x must be stored either in $T_1[h_1(x)]$ or $T_2[h_2(x)]$ (fast lookups and deletions!)

Definition

If S can be stored according to these rules, we call (h_1, h_2) **suitable for** S.

Improving Cuckoo Hashing: Stash

- Original Analysis: (h_1, h_2) unsuitable with probability $O(1/n)$.
  In fact: $\Theta(1/n)$ (Schellbach '09, Drmota/Kutzelnigg '12)
- (Kirsch/Mitzenmacher/Wieder '08): $\Theta(1/n)$ is too large.
- Proposal: Can put up to $s = O(1)$ keys into additional storage

Theorem (K/M/W '08)

Let $S \subseteq U$ with $|S| = n$. If (h_1, h_2) are **fully random**, then

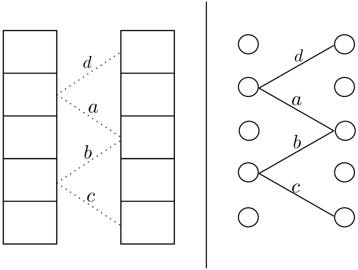
$$\Pr((h_1, h_2) \text{ unsuitable for } S \text{ with stash size } s) = O(1/n^{s+1}).$$

Again: $\Theta(1/n^{s+1})$. (Kutzelnigg '10)

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

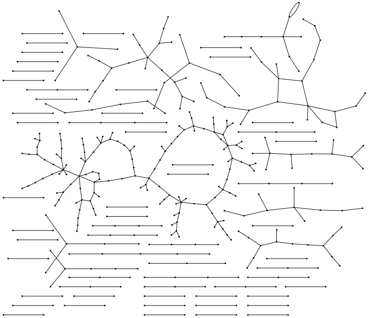
Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

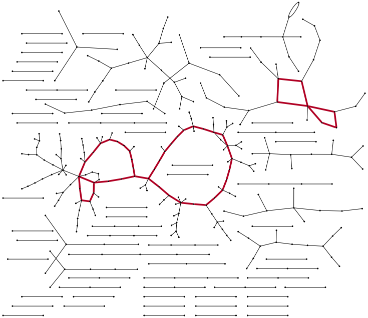
Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

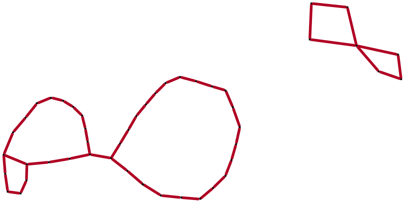
Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)

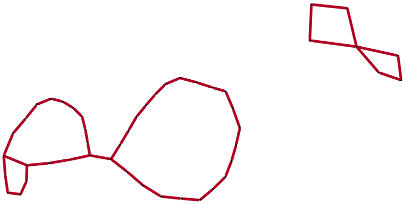


Excess (Janson et al. '93): #edges - #vertices (Here: 3)

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



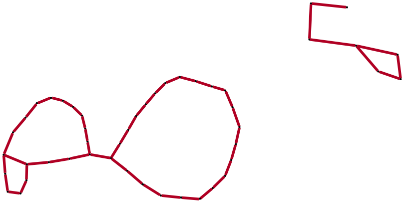
Excess (Janson et al. '93): #edges - #vertices (Here: 3)

3 more keys than table cells $\Rightarrow$ **at least** 3 keys must be put into stash

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



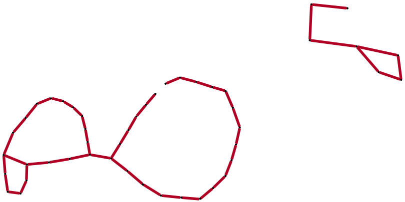
Excess (Janson et al. '93): #edges - #vertices (Here: 3)

3 more keys than table cells $\Rightarrow$ **at least** 3 keys must be put into stash

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



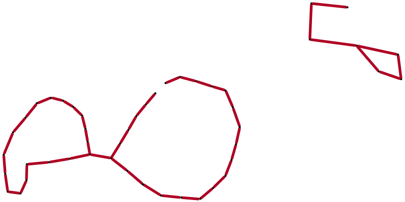
Excess (Janson et al. '93): #edges - #vertices (Here: 3)

3 more keys than table cells $\Rightarrow$ **at least** 3 keys must be put into stash

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



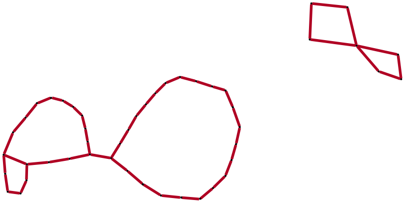
Excess (Janson et al. '93): #edges - #vertices (Here: 3)

3 more keys than table cells $\Rightarrow$ **at least** 3 keys must be put into stash

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



Excess (Janson et al. '93): #edges - #vertices (Here: 3)

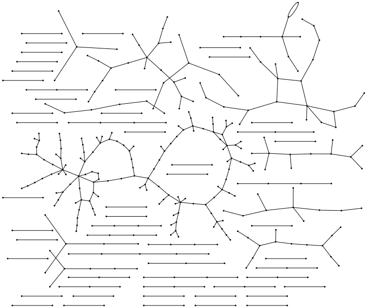
3 more keys than table cells $\Rightarrow$ **at least** 3 keys must be put into stash

Minimal "bad subgraph": a MOS_s. (Example: $s = 2$.)

Analysis of Cuckoo Hashing with a Stash

What is a criteria for (h_1, h_2) being unsuitable for stash size s?

Tool: Cuckoo graph $G(S, h_1, h_2)$ (Devroye/Morin '03)



Theorem (K/M/W '08)

Let (V', E') consists of all connected components of $G(S, h_1, h_2)$ having more than one cycle. Then

$$\text{Stash size} = |E'| - |V'|.$$

The Quest

The Quest

Analysis well understood when hash functions are **fully random**.

The Quest

Analysis well understood when hash functions are **fully random**.

- Replace **fully random** hash functions by an **explicit, efficient construction of hash functions.**

The Quest

Analysis well understood when hash functions are **fully random**.

- Replace **fully random** hash functions by an **explicit, efficient construction of hash functions.**

  "Simple hash functions that work in as many applications as possible"

The Quest

Analysis well understood when hash functions are **fully random**.

- Replace **fully random** hash functions by an **explicit, efficient construction of hash functions.**

  "Simple hash functions that work in as many applications as possible"

  Other recent approaches, e. g., Thorup/Pătrașcu '11, Reingold/Rothblum/Wieder '14

The Quest

Analysis well understood when hash functions are **fully random**.

- Replace **fully random** hash functions by an **explicit, efficient construction of hash functions.**

  "Simple hash functions that work in as many applications as possible"

  Other recent approaches, e. g., Thorup/Pătrașcu '11, Reingold/Rothblum/Wieder '14
- Focus on hashing-based algorithms and data structures that allow good enough bounds via **first-moment method** (C.H. [stash], generalized C.H., load balancing, ...)

The Quest

Analysis well understood when hash functions are **fully random**.

- Replace **fully random** hash functions by an **explicit, efficient construction of hash functions.**

  "Simple hash functions that work in as many applications as possible"

  Other recent approaches, e. g., Thorup/Pătrașcu '11, Reingold/Rothblum/Wieder '14

- Focus on hashing-based algorithms and data structures that allow good enough bounds via **first-moment method** (C.H. [stash], generalized C.H., load balancing, ...)

  Generic approach?

Key Ingredient: Linear Functions

$$h(x) = ((a \cdot x + b) \mod p) \mod m,$$

where

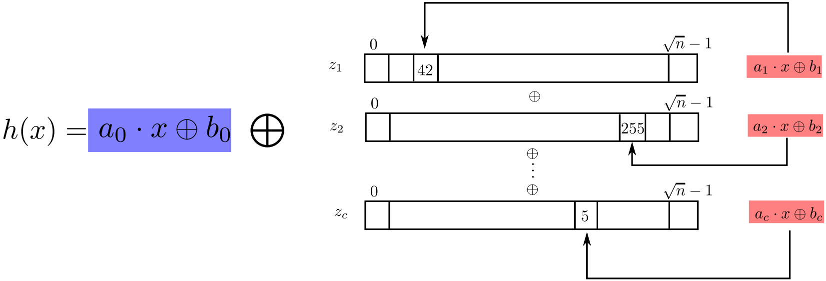
- $p \geq |U|$ is a prime, and
- a and b are chosen uniformly at random from $\{0, \ldots, p-1\}$.

$\rightarrow$ very simple structure!

(Remark: This function is 2-wise independent, i. e., for any pair $x, y \in U, x \neq y$, $h(x)$ and $h(y)$ are fully random.)

The Hash Class (Version for this Talk)

For given $c, n \geq 1$, we combine linear functions with lookups in tables of size $\sqrt{n}$ filled with random values.



$$h_i(x) = f_i(x) \oplus \bigoplus_{j=1}^{c} z_j^{(i)}[\, g_j(x)\,], \qquad i = 1, 2$$

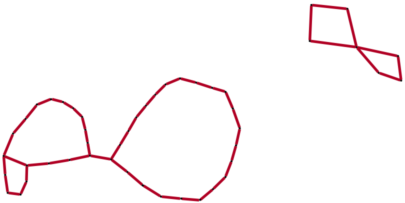
Class of all these pairs (h_1, h_2) **of hash functions**: $\mathcal{Z}$.
(Extension of hash functions from (Dietzfelbinger/Woelfel '03))

Example: Cuckoo Hashing with a Stash

Main Task

For given S and stash size s, calculate

$$\Pr((h_1, h_2) \text{ unsuitable for } S \text{ with stash size } s).$$



Minimal bad subgraph: MOS_s. (Example: $s = 2$.)

Thus, we have

$$
\begin{aligned}
&\Pr_{(h_1,h_2)\in\mathcal{Z}}((h_1,h_2) \text{ unsuitable for } S \text{ with stash size } s)\\
&= \Pr_{(h_1,h_2)\in\mathcal{Z}}(\exists T \subseteq S : G(T,h_1,h_2) \text{ forms a } \mathrm{MOS}_s)\\
&\le \sum_{T\subseteq S} \Pr_{(h_1,h_2)\in\mathcal{Z}}(G(T,h_1,h_2) \text{ forms a } \mathrm{MOS}_s)
\end{aligned}
$$

- if (h_1,h_2) are fully random, we provide a direct counting argument that this is $O(1/n^{s+1})$

  giving an alternative proof to the original analysis by Kirsch, Mitzenmacher and Wieder (who used machinery like Markov chain coupling)

Behavior of the Hash Class on Fixed $T \subseteq S$

Recall:

$$h_i(x) = \boxed{f_i(x)} \oplus \bigoplus_{j=1}^{c} z_j^{(i)}[\boxed{g_j(x)}], \qquad i = 1, 2$$

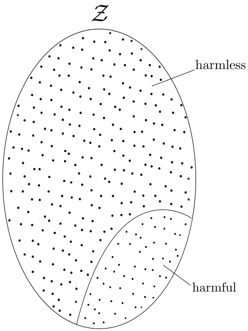
Central Observation

Let $T \subseteq S$. If there is a g_j such that at most one pair of keys in T collides under g_j (i. e., $g_j(x) = g_j(y)$), then h_1, h_2 are fully random on T.

- if this is the case: (h_1, h_2) T-**good**.
- otherwise (each g_j has more than one colliding pair of keys): (h_1, h_2) is T-**bad**.

Collecting "Harmful" Hash Functions

We split our set of hash functions into "harmful" and "harmless" ones.



(h_1, h_2) are harmful, if there exists $T \subseteq S$ s.t.

- $G(T, h_1, h_2)$ forms a MOS_s, and
- (h_1, h_2) is T-bad.

$B^{\mathbf{MOS}_s} :=$ the set of all the harmful pairs (h_1, h_2). (An event in our probability space!)

Splitting the Calculation

We calculate:

$$\Pr(N_S^{\mathsf{MOS}_s} > 0) \;\leq\; \Pr(N_S^{\mathsf{MOS}_s} > 0 \cap \neg B^{\mathsf{MOS}_s}) \;+\; \Pr(B^{\mathsf{MOS}_s})$$

Splitting the Calculation

We calculate:

$$\Pr(N_S^{\mathrm{MOS}_s} > 0) \;\le\; \boxed{\Pr(N_S^{\mathrm{MOS}_s} > 0 \cap \neg B^{\mathrm{MOS}_s})} \;+\; \Pr(B^{\mathrm{MOS}_s})$$

- for **this** summand, we have

$$\Pr(N_S^{\mathrm{MOS}_s} > 0 \cap \neg B^{\mathrm{MOS}_s}) \le \mathrm{E}^*\left(N_S^{\mathrm{MOS}_s}\right),$$

  which is $O(1/n^{s+1})$.

Splitting the Calculation

We calculate:

$$\Pr(N_S^{\mathrm{MOS}_s} > 0) \ \leq \ \boxed{O(1/n^{s+1})} \ + \ \Pr(B^{\mathrm{MOS}_s})$$

- for **this** summand, we have

$$\Pr(N_S^{\mathrm{MOS}_s} > 0 \cap \neg B^{\mathrm{MOS}_s}) \leq \mathrm{E}^*\left(N_S^{\mathrm{MOS}_s}\right),$$

which is $O(1/n^{s+1})$.

Splitting the Calculation

We calculate:

$$\Pr(N_S^{\mathsf{MOS}_s} > 0) \;\leq\; \boxed{O(1/n^{s+1})} + \boxed{\Pr(B^{\mathsf{MOS}_s})}$$

- for **this** summand, we have

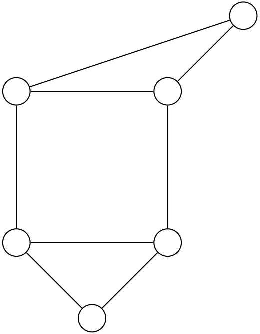
  $$\Pr(N_S^{\mathsf{MOS}_s} > 0 \cap \neg B^{\mathsf{MOS}_s}) \leq \mathrm{E}^* \left(N_S^{\mathsf{MOS}_s}\right),$$

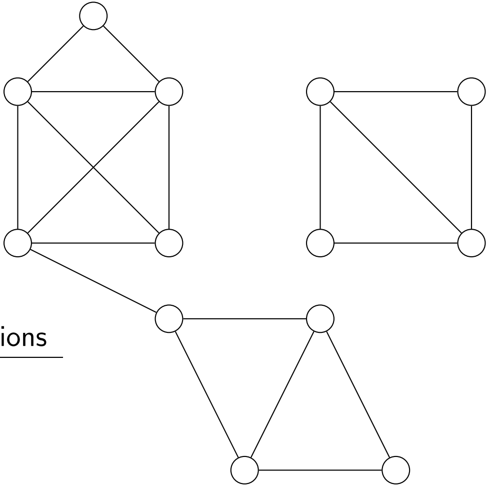
  which is $O(1/n^{s+1})$.
- **The hard part**: Calculating/bounding

  $\Pr(B^{\mathsf{MOS}_s}) = \Pr(\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\mathsf{MOS}_s \cap (h_1, h_2)$ are T-bad)
- Wish: Use full randomness nonetheless
- Idea: Find a suitable event that contains B^{MOS_s}

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad ".

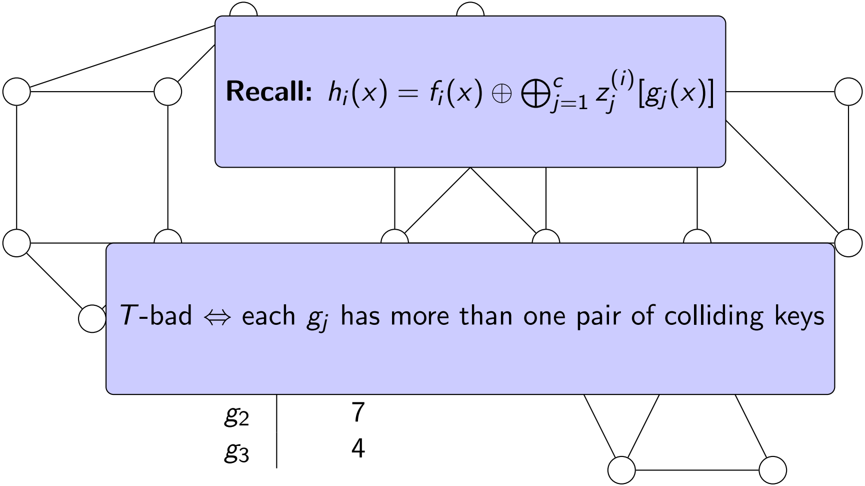




	#collisions
g_1	5
g_2	7
g_3	4

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\text{MOS}_s \cap (h_1, h_2)$ are T-bad ".



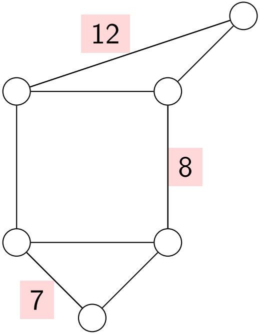
Recall: $h_i(x) = f_i(x) \oplus \bigoplus_{j=1}^{c} z_j^{(i)}[g_j(x)]$

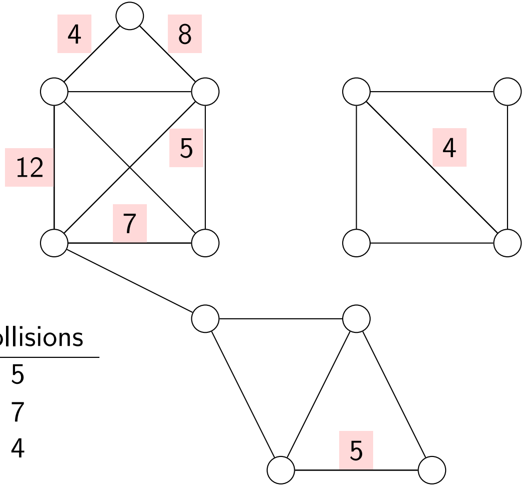
T-bad $\Leftrightarrow$ each g_j has more than one pair of colliding keys

g_2	7
g_3	4

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\text{MOS}_s \cap (h_1, h_2)$ are T-bad ".

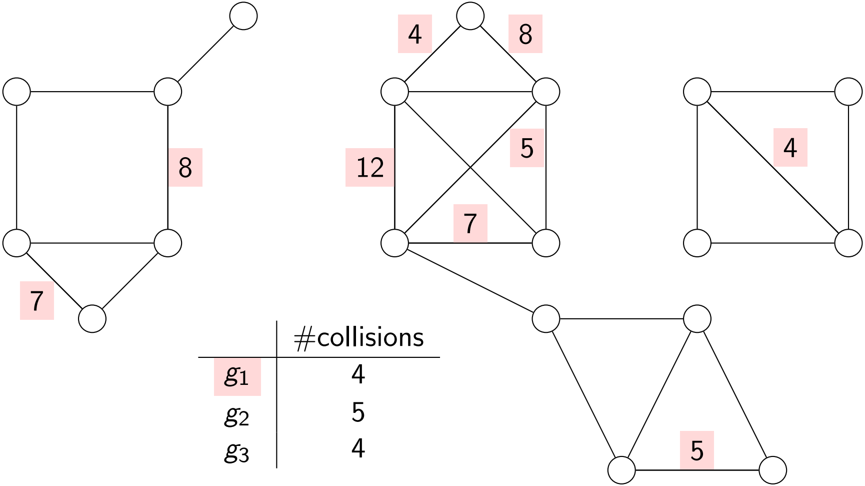




	#collisions
g_1	5
g_2	7
g_3	4

Peeling of Bad Graphs (Simplified)

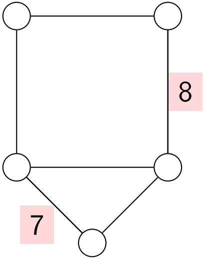
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad ".

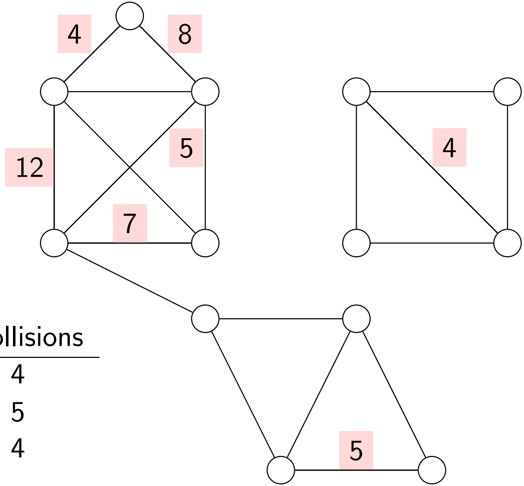


	#collisions
g_1	4
g_2	5
g_3	4

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap$ (h_1, h_2) are T-bad ".

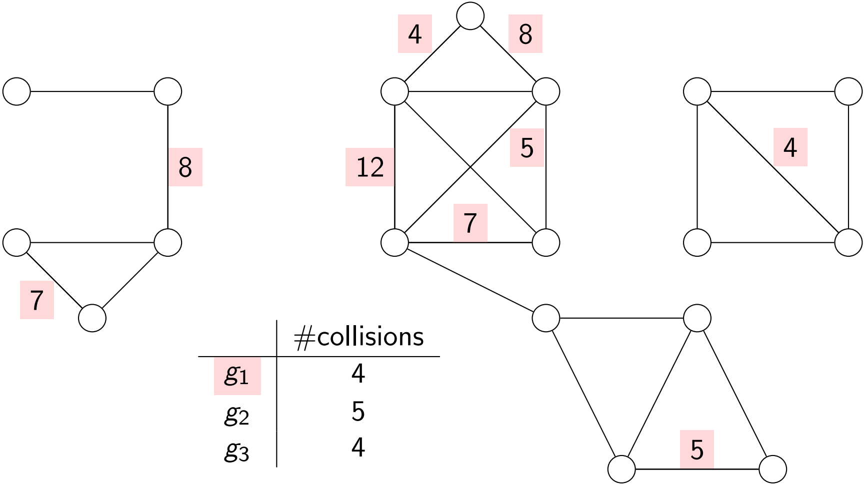




	#collisions
g_1	4
g_2	5
g_3	4

Peeling of Bad Graphs (Simplified)

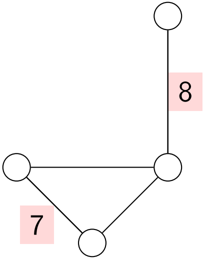
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap$ (h_1, h_2) are T-bad".

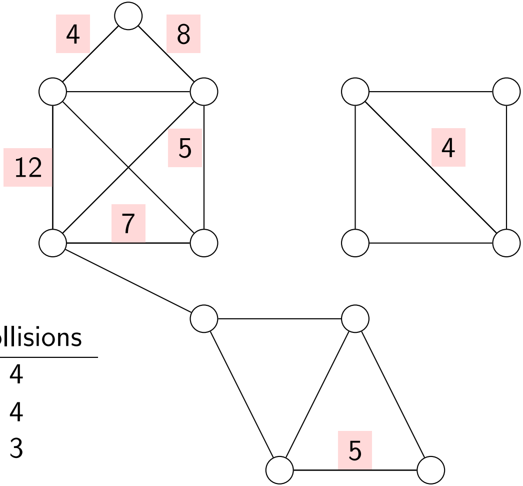


	#collisions
g_1	4
g_2	5
g_3	4

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap$ (h_1, h_2) are T-bad ".

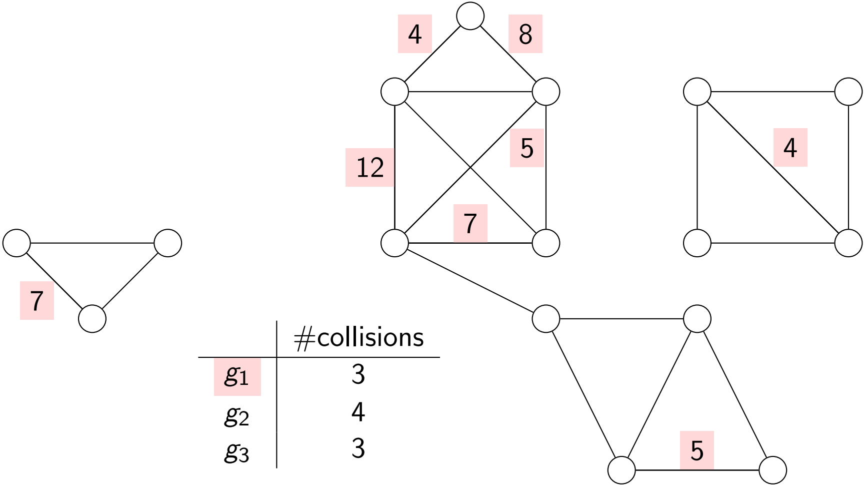




	#collisions
g_1	4
g_2	4
g_3	3

Peeling of Bad Graphs (Simplified)

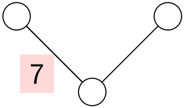
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap (h_1, h_2)$ are T-bad ".

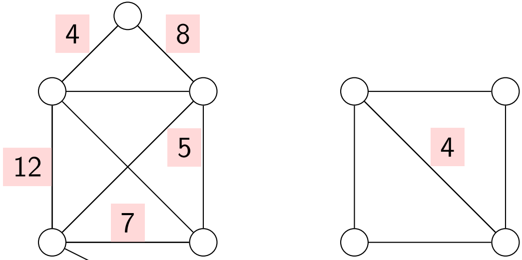


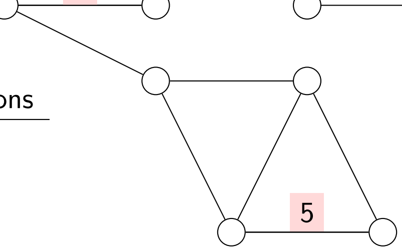
	#collisions
g_1	3
g_2	4
g_3	3

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\text{MOS}_s \cap (h_1, h_2)$ are T-bad ".



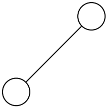


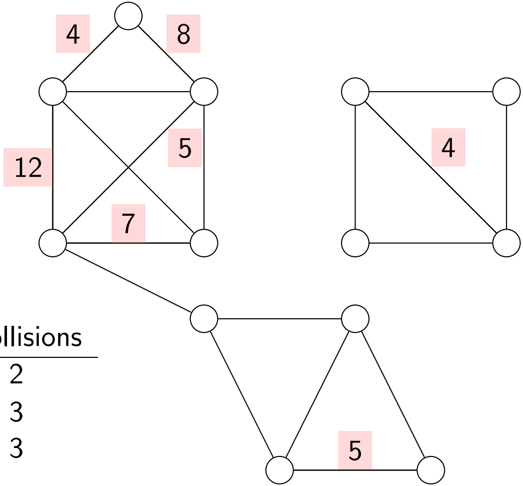


	#collisions
g_1	3
g_2	4
g_3	3

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad ".

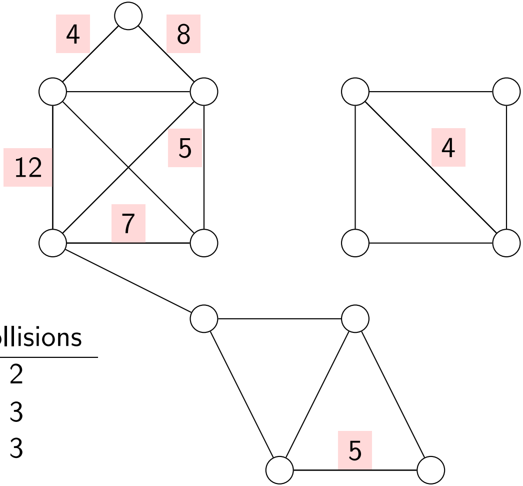




	#collisions
g_1	2
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

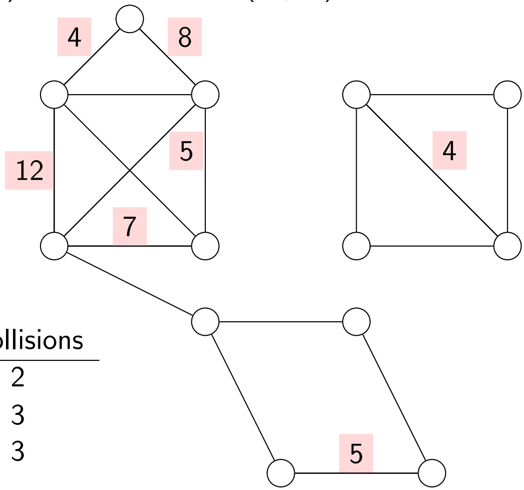
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad ".



	#collisions
g_1	2
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

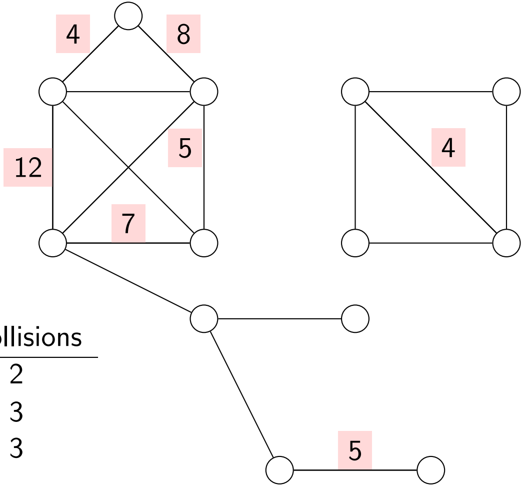
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad ".



	#collisions
g_1	2
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

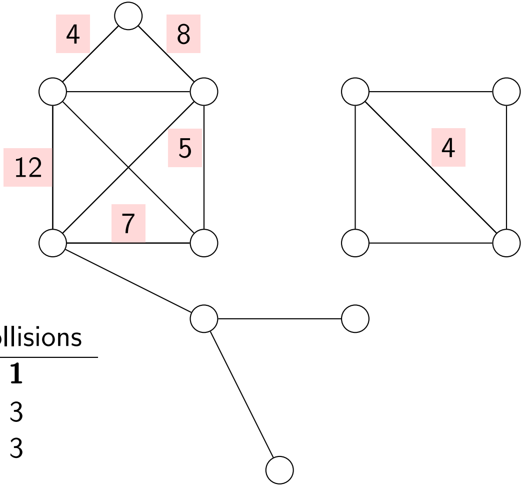
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\text{MOS}_s \cap (h_1, h_2)$ are T-bad "



	#collisions
g_1	2
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

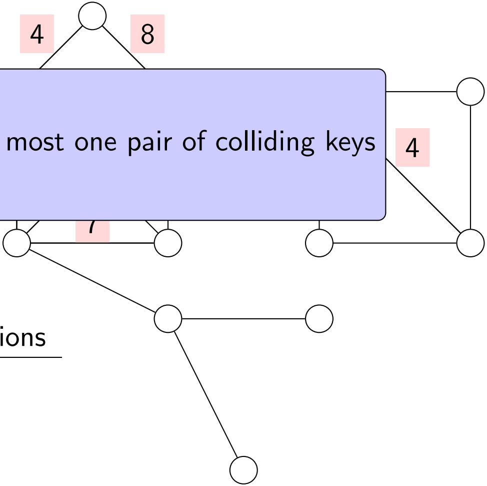
Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad "



	#collisions
g_1	**1**
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

Assume "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a MOS_s $\cap\, (h_1, h_2)$ are T-bad "

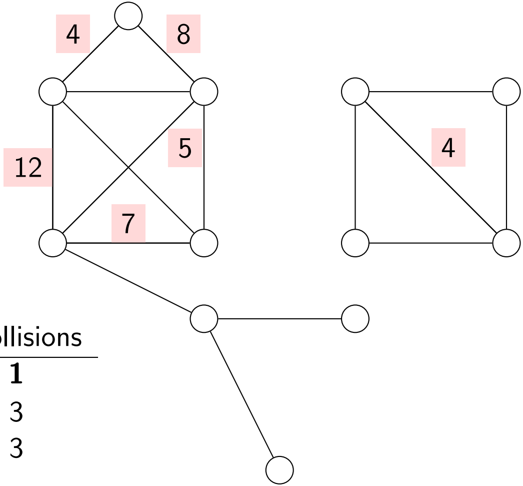


T-good $\Leftrightarrow$ exists g_j with at most one pair of colliding keys

	#collisions
g_1	**1**
g_2	3
g_3	3

Peeling of Bad Graphs (Simplified)

If "$\exists T \subseteq S : G(T, h_1, h_2)$ forms a $\text{MOS}_s \cap (h_1, h_2)$ are T-bad "



	#collisions
g_1	**1**
g_2	3
g_3	3

then "$\exists T' \subseteq S: G(T', h_1, h_2)$ forms "peeled graph" $\cap (h_1, h_2)$ are T'-good"

Result of Peeling

$$\Pr(\exists T \subseteq S : G(T, h_1, h_2) \text{ forms a } \mathrm{MOS}_s \ \cap (h_1, h_2) \ T\text{-bad })$$
$$\leq \Pr(\exists T' \subseteq S\colon G(T', h_1, h_2) \text{ is peeling result} \cap (h_1, h_2) \ T'\text{-good})$$

- can again use first-moment approach
- resulting graphs are sparser $\rightarrow$ they are more likely to occur
- use: when process stops each $g_j, 1 \leq j \leq c$, has a colliding pair of keys
- probability boost of $\approx (1/\sqrt{n})^c$
- probability of B^{MOS_s} is $O(n/\sqrt{n}^c)$, which is $O(1/n^{s+1})$ for $c = \Theta(s)$

Some applications need an additional "reduction step".
(Preserve collisions, make graphs smaller.)

Result

- Graph property of interest: $\mathcal{A}$, via first-moment approach

$$\mathrm{E}^*(\#\text{subgraphs with property } \mathcal{A}) = O\left(n^{-\alpha}\right).$$

- Assume there exists peelable graph property $\mathcal{B} \supseteq \mathcal{A}$ with

$$\sum_{t=2}^{n} t^{O(1)} \mathrm{E}^*(\#\text{subgraphs with property } \mathcal{B} \text{ with } t \text{ edges}) = O\left(n^{\beta}\right).$$

Trick: $\mathcal{B}$ can be quite general, e. g., “leafless”.

Result

- Graph property of interest: $\mathcal{A}$, via first-moment approach

$$\mathrm{E}^*(\#\text{subgraphs with property } \mathcal{A}) = O\left(n^{-\alpha}\right).$$

- Assume there exists peelable graph property $\mathcal{B} \supseteq \mathcal{A}$ with

$$\sum_{t=2}^{n} t^{O(1)} \mathrm{E}^*(\#\text{subgraphs with property } \mathcal{B} \text{ with } t \text{ edges}) = O\left(n^{\beta}\right).$$

Trick: $\mathcal{B}$ can be quite general, e. g., "leafless".

Using $c \geq 2(\alpha + \beta)$ g-functions and tables gives

$$\Pr_{(h_1,h_2)\in\mathcal{Z}}(\text{Graph contains subgraph with property } \mathcal{A}) = O(n^{-\alpha}).$$

Examples

Graphs:

Examples

Graphs:

- Cuckoo hashing (with a stash)
- Applications which need that largest component is $O(\log n)$ w.h.p.
- Simulation of a uniform hash function (Pagh/Pagh '03)
- Constructing a perfect hash function (Bothelo/Pagh/Ziviani '13)

Examples

Graphs:

- Cuckoo hashing (with a stash)
- Applications which need that largest component is $O(\log n)$ w.h.p.
- Simulation of a uniform hash function (Pagh/Pagh '03)
- Constructing a perfect hash function (Bothelo/Pagh/Ziviani '13)

Hypergraphs:

Examples

Graphs:

- Cuckoo hashing (with a stash)
- Applications which need that largest component is $O(\log n)$ w.h.p.
- Simulation of a uniform hash function (Pagh/Pagh '03)
- Constructing a perfect hash function (Bothelo/Pagh/Ziviani '13)

Hypergraphs:

- Parallel/Sequential Load Balancing: basically match bounds from fully random case (Schickinger/Steger '00).
- Generalized cuckoo hashing (≥ 3 hash functions, $\ell \geq 2$ keys per cell): Admits first-moment approach, but **could not find** suitable peelable graph property in the hypergraph setting to prove table loads $\rightarrow 1$.

Conclusion

We have seen:

- a class of hash functions that behaves "well" in different applications
- in first-moment type analyses: Can use full randomness, no properties of hash class exposed

Open:

- better bounds for some applications?
- bounds beyond first moment?

Conclusion

We have seen:

- a class of hash functions that behaves “well” in different applications
- in first-moment type analyses: Can use full randomness, no properties of hash class exposed

Open:

- better bounds for some applications?
- bounds beyond first moment?

Thank you!