6.851 André Schulz' Notes

Lecture 14 DICTIONARIES (HASHING WITH CHAINING)

(1) Universal Hashing

[Carter, Wegman '79]

We want to use a hash-function, to store subset elements of a big universe $S \subseteq \mathcal{U} = \{0, \dots, u-1\}$ ($= \mathcal{U}$, u) in a small table $\{0, \dots, m-1\}$ ($= T$, m).

We use randomness to avoid bad worst-case behavior in expectation

Def A class of hash functions $\mathcal{H}$ is *c-universal*, iff:

$$\forall x, y \, (x \neq y) \quad |\{h \in \mathcal{H} \mid h(x) = h(y)\}| \leq c \cdot \frac{|\mathcal{H}|}{m}$$

c-universal classes imply: if $h \in \mathcal{H}$ is chosen randomly:

$$\forall x, y, (x \neq y) \quad \mathbb{P}[h(x) = h(y)] \leq \frac{c}{m}$$

(for h chosen uniformly at random in $\mathcal{H}$)

We pick m such that $\frac{n}{m} \in O(1)$

and study the collisions with $x \in U$

(Chain-length)

$\delta_{h(x)h(y)}$ = Kronecker delta

$z_x = |\{h(x) = h(y) \mid y \in T\}| = \sum_{y \in T}' 1$ (1 if $h(x)=h(y)$, 0 else)

(' Collisions of x)

$$E(z_x) = \sum_{y \in T}' \delta_{h(x),h(y)} = \sum_{y \in T}' \Pr[h(x)=h(y)]$$

$$= 1 + \sum_{\substack{y \in T \\ y \neq x}}' \Pr[h(x)=h(y)]$$

$$\leq 1 + \frac{n}{m} \cdot c$$

$$= O(1)$$

□

Example for a 2-universal family of Hash-functions:

$$h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$$

with p prime

Proof: Textbook (e.g. Cormen)

$h_{a,b}(x)$ has : compact representation $O(1)$

: $O(1)$ evaluation

↳ good hash-function

Summary : Universal hashing with chaining needs $O(1)$ time / Op in expectation [amortized]

drawback : worst case, for a "bad" sequence we may have long query/insert times

↑ $\Theta\left(\frac{\log n}{\log \log n}\right)$ = longest chain

Solutions for the Static Case

~~[Mitzenmacher PhD 96]~~ [Azar et al. '94] [Mitzenmacher 96]

Use two (d) hash functions and add the key in the currently shorter list.

↳ longest chain $\Theta(\log \log n)$ $\left(\Theta\left(\frac{\log \log n}{\log d}\right)\right)$

Perfect Hashing (Static)

[Fredman, Komlós, Szemerédi '84]

For a fixed set of n keys S find a hash-function h that produces no collisions.

As usual: • h has compact representation
• h can be evaluated in $O(1)$

$z(h)$ = # of all collisions of h

$$= \sum_{\substack{x,y \in S \\ x<y}} \delta_{h(x),h(y)}$$

Assume h is 2-universal

$$\mathbb{E}(z(h)) = \sum_{\substack{x,y \in S \\ x<y}} \mathbb{E}(\delta_{h(x)h(y)}) = \sum_{\substack{x,y \in S \\ x<y}} \mathbb{P}[h(x)=h(y)]$$

$$\leq \binom{u}{2} \frac{2}{m} < \frac{u^2}{m} \quad (*)$$

($\leq \frac{c}{m}$)

$\Rightarrow$ For table of size $2u^2$: $\mathbb{E}(z(h)) < \frac{1}{2}$

Aside: Markov inequality:

X non-neg Random variable:

$$\mathbb{P}[X > a \cdot \mathbb{E}(X)] \leq \frac{1}{a}$$

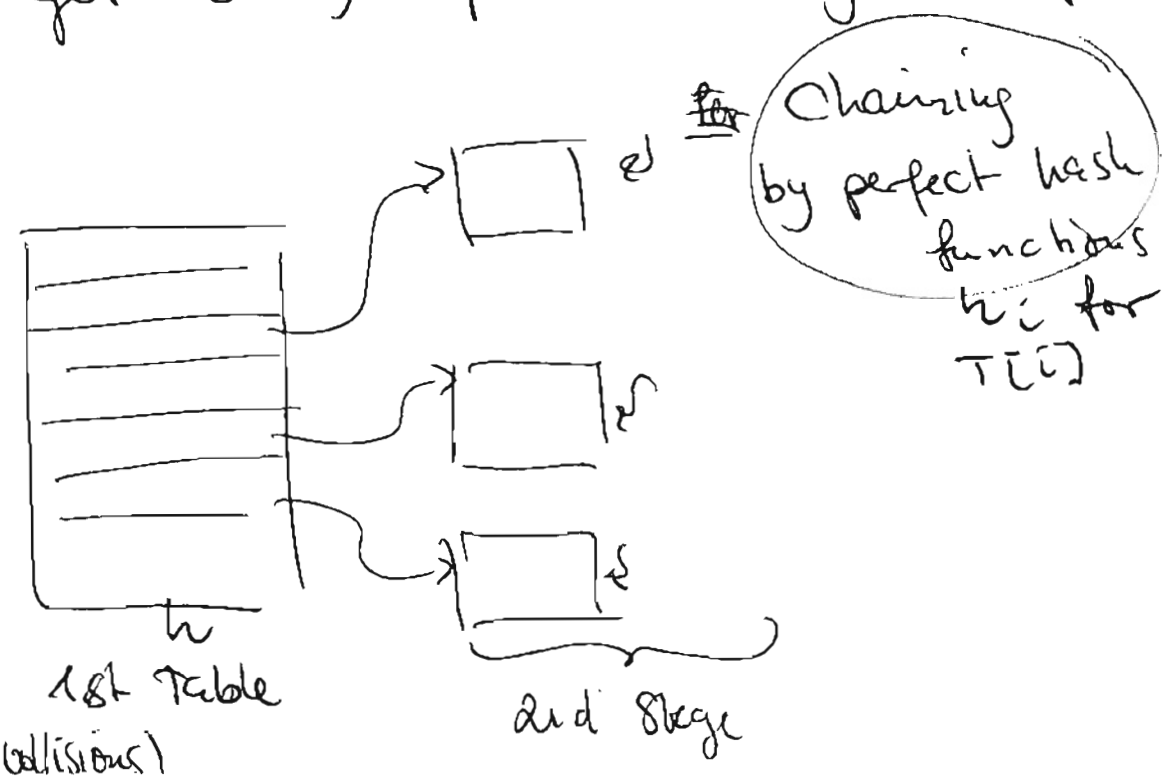
Thus: $\mathbb{P}[z(k) \geq 2\,\mathbb{E}(z(k))] \leq \frac{1}{2}$

$$\Rightarrow \mathbb{P}[z(k) > 1] \leq \frac{1}{2}$$

↳ We can find a "perfect" hash-function by picking $h \in \mathcal{H}$ randomly and check if h is perfect. If not, pick again.

Table size: $O(k^2)$

We can get $O(k)$ space using 2-stage hashing:



$\Rightarrow$ S is partitioned into $S_0 \dot\cup S_1 \dot\cup \dots \dot\cup S_{m-1}$

with: $S_i = \{ x \in S \mid h(x) = i \}$

$$Z(h) = \sum_{i=0}^{s-1} \left| \{ (x,y) \mid x,y \in S, x<y, h(x)=h(y)=i \} \right|$$

$$= \sum_{i=0}^{s-1} \binom{|S_i|}{2}$$

Hence $$\mathbb{E}\left(\sum_{i=0}^{s-1} |S_i|^2\right) = \mathbb{E}\left(\sum_{i=0}^{u} |S_i|\right) + 2\,\mathbb{E}\left(\sum_{i=0}^{u} \binom{|S_i|}{2}\right)$$

$$= u + 2\,\mathbb{E}\left(\sum_{i=0}^{s-1} \binom{|S_i|}{2}\right)$$

$$= u + 2\frac{u^2}{m} \qquad \text{(Due to (*))}$$

$$= 2u \quad \text{for } m = 2u$$

$2\binom{u}{2} = 2\frac{u(u-1)}{2} = u^2 - u$

$\Rightarrow O(u)$ Space

Due to Markov's Inequality:

$$\mathbb{P}\left(\sum |S_i|^2 \geq 4u\right) < \frac{1}{2}$$

$\Rightarrow$ (1) Choose h until $\sum |S_i|^2 < 4u$

$\Rightarrow$ (2) Chose $\{h_i\}$ until h_i perfect $\square$

Cuckoo-Hashing

[Pagh-Rodler 2001]

Def A class of hash-functions $\mathcal{H}$ is said to be (c,k)-universal, iff

$\forall\, x_1, x_2, \ldots, x_k \in \mathcal{U}$ (distinct)

$\forall\, a_1, a_2, \ldots, a_k \in T$ (distinct):

$$\#\left[h(x_1)=a_1 \wedge h(x_2)=a_2 \wedge \ldots \wedge h(x_k)=a_k\right] = c \cdot \frac{|\mathcal{U}|}{|T|^k}$$

For Cuckoo hashing we need 2 $(c, \log u)$ universal hashfunctions f, g

- f, g can be found (represented in $O(\log u)$ space and evaluated in $O(1)$ [Siegel '89]

- Idea Cuckoo hashing: insert $x \to T[f(x)]$, if there is some $y \in T[f(x)]$, insert y in $T[g(y)]$ or $T[f(y)]$
- Stop after $O(\log u)$ evictions

Find (x) : Check $T[f(x)]$ & $T[g(x)]$

Delete (x) : if $x \in T[f(x)]$ or $T[g(x)]$
remove x from the table

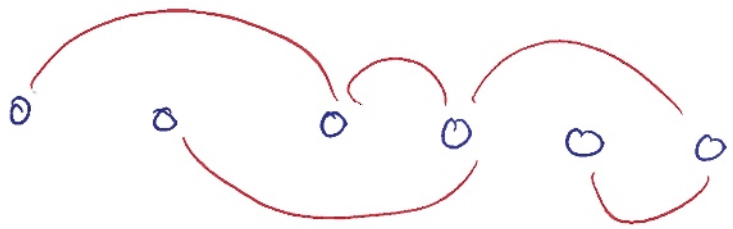
Insert (x) :

```
i ← 1
while (i ≤ maxloop)
    if T[f(x)] = empty
        then T[f(x)] ← x
             return
    else swap x ⇔ T[f(x)]
    if T[g(x)] = empty
        then T[g(x)] ← x
             return
    else swap x ⇔ T[g(x)]
    i ← i+1
Rehash with new f/g
insert(x)
```

maxloop = $O(\log n) = 6 \log n$

What is the running time of INSERT(x) ?

Visualization: Cuckoo-graph



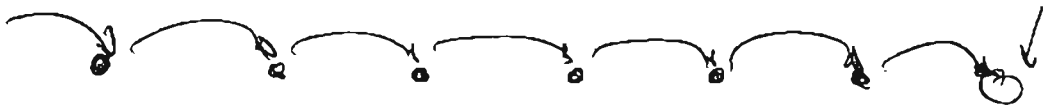
nodes = cells in T

edge $(i,j) \Leftrightarrow \exists x \in T : f(x)=i, g(x)=j$

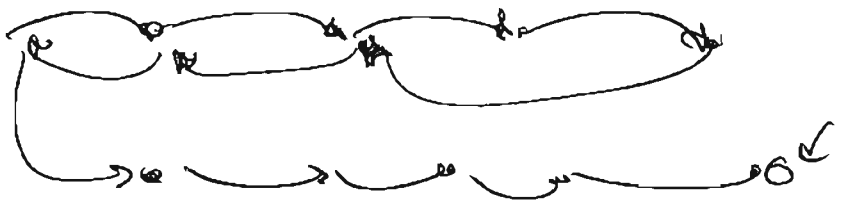
Let n be the # of elements in the table
Assume that $n \leq 4m$

- We have 3 types of situations when inserting

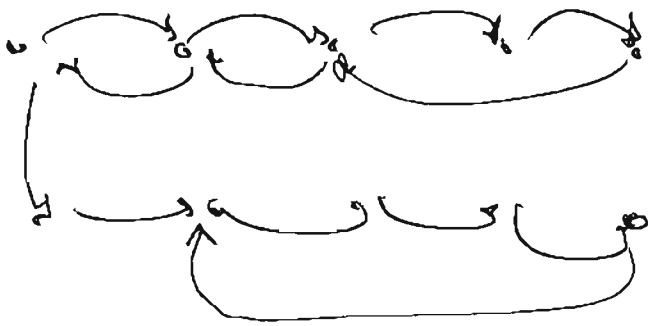
Case A: (Path)



Case B: (1 cycle)



Case C: (2 cycles)



no empty cell

Case A: ($c = 1$)

$$\mathbb{P}[T[f(x)] \text{ is occupied}]$$

$$= \mathbb{P}\left(\exists y \in S, y \neq x : f(x) = f(y) \text{ or } f(x) = g(y)\right)$$

$$= \sum_{\substack{y \in S \\ y \neq x}} \left(\mathbb{P}\left(f(x) = f(y)\right) + \mathbb{P}\left[f(x) = g(y)\right]\right) \leq 2 \cdot \frac{4c}{8n} = \frac{4c}{4n} = \frac{c}{2}$$

$$\mathbb{P}[2 \text{ evictions}] =$$

$$\sum_{y_1, y_2}' \mathbb{P}\left[f(x) = f(y_1) \wedge g(y_1) = f(y_2)\right]$$
$$+ \mathbb{P}\left[f(x) = g(y_1) \wedge f(y_1) = g(y_2)\right]$$
$$+ \mathbb{P}\left[f(x) = f(y_1) \wedge g(y_1) = g(y_2)\right]$$
$$+ \mathbb{P}\left[f(x) = g(y_1) \wedge f(y_1) = g(y_2)\right] \leq 4 \cdot \frac{c}{8n^2} \leq \frac{c}{4}$$

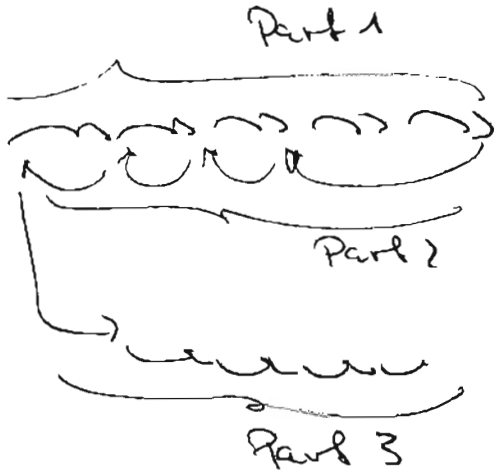
$$\mathbb{P}[t \text{ evictions}] \leq \frac{c}{2^t}$$

$$\Rightarrow \mathbb{E}(\text{evictions in case A}) = \sum_{t=0}^{\infty} (t+1)\frac{c}{2^t} \quad (\downarrow = 4 \cdot c) = O(1)$$

$$\mathbb{P}[\text{rehash in case A}) = 2^{-6\log n} \leq 2^{-\log n^2} = O\left(\frac{1}{n^2}\right)$$

Case B:

Observation



t steps

Because Part 2 ⊂ Part 1 : Part 1 or Part 3 $\geq t/3$

$\Rightarrow \mathbb{P}[\text{1 loop with } t \text{ steps}] <$

$\mathbb{P}[\text{path of length } t/3 \text{ from } x] = 2^{-\frac{t}{3}}$

$\Rightarrow \mathbb{E}(\text{running time case B}) \leq \sum_t (t+1)\, 2^{-\frac{t}{3}} = O(1)$

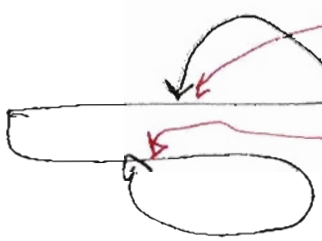
$\mathbb{P}(\text{reject}) = 2^{-\frac{6}{3}\log n} = O(\frac{1}{n^2})$

Case C (via counting)

How many 2-loop scenarios ~~length~~ t distinct ~~loops~~ nodes

: combinatorial 2-loop graphs: $\leq t^3$

(3 important positions : 1st jump source
1st jump target
2nd jump target)



How many ways to place a 2loop in the table? : $m^{t-1} = (4u)^{t-1}$ (position of x is assigned)

How many ways to place keys in the cells? : u^{t-1}

$\Rightarrow$ in total $t^3 u^{t-1} (4u)^{t-1}$ configurations

How many cuckoo graphs are there? (with t edges)

$\forall$ edge (u,v) $\exists y$: $f(y) = u$ $g(y) = v$

or $g(y) = u$ $g(y) = v$

$\Rightarrow$ $\forall$ edge : $\frac{m^2}{2} = \frac{(4u)^2}{2}$ choices

$\Rightarrow$ in total $\frac{(4u)^{2t}}{2^t}$

$\rightarrow \mathbb{P}[\text{we have case } C] = \frac{2^t}{(4u)^{2t}} \cdot t^3 u^{t-1} (4u)^{t-1}$ (with t elements)

$$= \frac{1}{8u^2} \frac{t^3}{2^{t-1}}$$

$O(1)$

$\Rightarrow \mathbb{P}[\text{we have case } C] \leq \sum_{t=2}^{\infty} \frac{t^3}{8u^2 2^{t-1}} = \frac{1}{8u^2} \sum_{t=2}^{\infty} \frac{t^3}{2^{t-1}}$

$= O(\frac{1}{u^2})$

$\Rightarrow \mathbb{P}[\text{rehash}] = O(\frac{1}{u^2})$ { long case A, B ; case C

$\Rightarrow \mathbb{P}[\text{rehash after } u \text{ insertions}] : O(\frac{1}{u})$

$\Rightarrow \mathbb{P}[\text{rehash is successful}] = 1 - O(\frac{1}{u})$

↳ successful after $O(1)$ tries

~~$\Rightarrow$ Rehashing takes $O(u)$ time in expectation~~

~~↳ amortizable over the~~

$\Rightarrow$ Case C: costs $O(\log u) + O(u)$

↑ maxloop steps ↑ rehash

$\Rightarrow$ In expectation: $O(\frac{1}{u^2} \cdot O(u)) = O(\frac{1}{u})$

$\Rightarrow$ Running time INSERT: $O(1) + O(\frac{1}{u}) = O(1)$

in expectation, amortized

□