CPSC 420 Lecture 27: Today's announcements:

- HW4 is on Gradescope, due Mar 30, 23:59
- Examlet 4 on ~~Feb~~ April 5 in class. Closed book & no notes
- Reading: `https://student.cs.uwaterloo.ca/~cs466/Old_courses/F10/online_list.pdf` [by López-Ortiz]
  `https://courses.csail.mit.edu/6.897/spring03/scribe_notes/L5/lecture5.pdf` [by Demaine]
  `https://courses.csail.mit.edu/6.897/spring03/scribe_notes/L6/lecture6.pdf` [by Demaine]
- Reading: Ch.5 Hash Tables [Director's Cut by Erickson]

Today's Plan

- Online Algorithms
  - List Update
- Hashing

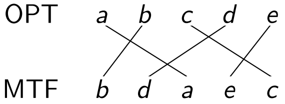
List Update: MTF versus Dynamic OPT

Theorem: For any sequence $s = s_1 s_2 \dots s_m$ of items to find,

$$\text{cost(MTF)} \leq 2\text{cost(OPT)}$$

where cost(A) is the cost (including paid swaps) of algorithm A on sequence s.

Proof: Let $\phi(i)$ be the number of inversions between the list orders of MTF and OPT after find(s_i).

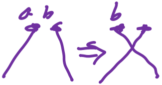


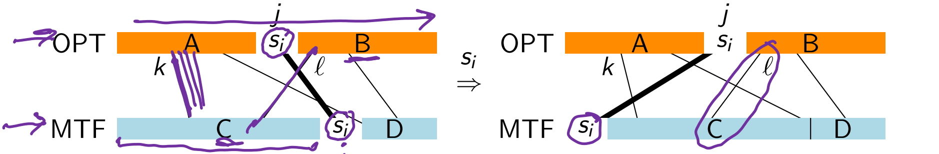
$$\phi(i) = 4$$

Let $c_i(A)$ be the cost of A on find(s_i).

We first show that $c_i(\text{MTF}) + \phi(i) - \phi(i-1) \leq 2c_i(\text{OPT}) - 1$.

MTF versus Dynamic OPT





k matches from A to C
ℓ matches from B to C

$\phi(i) - \phi(i-1) = k - \ell$

1. $c_i(\text{MTF}) = k + \ell + 1$
2. $\phi(i) - \phi(i-1) = \boxed{k - \ell}$
3. $c_i(\text{OPT}) = j + P(i) \geq k + 1 + P(i)$ where $P(i)$ is #paid swaps by OPT on ith find.
4. Each of $P(i)$ paid swaps increases $\phi(i)$ by ≤ 1.

So

$$c_i(\text{MTF}) + [\phi(i) - \phi(i-1)] \leq k + \ell + 1 + [\overbrace{k - \ell}^{2} + \overbrace{P(i)}^{4}]$$
$$= 2k + 1 + P(i) \overset{3}{\leq} 2c_i(\text{OPT}) - 1$$

MTF versus Dynamic OPT

Sum over all i to get: $\text{cost}(\text{MTF}) + \overbrace{\sum_{i=1}^{m}[\phi(i) - \phi(i-1)]}^{\text{difference in potential functions}} =$

$$\boxed{\sum_{i=1}^{m}(c_i(\text{MTF}) + [\phi(i) - \phi(i-1)])} \le \sum_{i=1}^{m}(2c_i(\text{OPT}) - 1)$$

$$= 2\text{cost}(\text{OPT}) - m$$

$$\le 2\text{cost}(\text{OPT})$$

Since $\sum_{i=1}^{m}[\phi(i) - \phi(i-1)] = \underbrace{\phi(m)}_{\ge 0} - \underbrace{\phi(0)}_{0} \ge 0$, we're done. $\square$

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u-1\}$ of possible keys to a slot (index from 0 to $m-1$) in a hash table (array) of size m.

What's a good hash function?

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u - 1\}$ of possible keys to a slot (index from 0 to $m - 1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u - 1\}$ of possible keys to a slot (index from 0 to $m - 1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?
  Impossible! $|U|/m$ must hash to the same slot.

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u-1\}$ of possible keys to a slot (index from 0 to $m-1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?
  Impossible! $|U|/m$ must hash to the same slot.
- SHA-3 cryptographic hash function?

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u-1\}$ of possible keys to a slot (index from 0 to $m-1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?
  Impossible! $|U|/m$ must hash to the same slot.
- SHA-3 cryptographic hash function?
  Expensive to compute. Secure? Any fixed hash function (used for all hash tables) can be studied to find many colliding keys.

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u-1\}$ of possible keys to a slot (index from 0 to $m-1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?
  Impossible! $|U|/m$ must hash to the same slot.
- SHA-3 cryptographic hash function?
  Expensive to compute. Secure? Any fixed hash function (used for all hash tables) can be studied to find many colliding keys.

  Choose hash function at random from a large set H.
- Keys spread evenly through hash table? $\Pr_{h \in H}[h(x) = i] = \frac{1}{m}$?

Hashing

A **hash function** maps keys from a universe $U = \{0, 1, \ldots, u-1\}$ of possible keys to a slot (index from 0 to $m-1$) in a hash table (array) of size m.

What's a good hash function?

- No two keys map to the same slot?
  Impossible! $|U|/m$ must hash to the same slot.
- SHA-3 cryptographic hash function?
  Expensive to compute. Secure? Any fixed hash function (used for all hash tables) can be studied to find many colliding keys.

Choose hash function at random from a large set H.

- Keys spread evenly through hash table? $\Pr_{h \in H}[h(x) = i] = \frac{1}{m}$?
  $H = \{\text{const}_i | 0 \leq i < m\}$ where $\text{const}_i(x) = i$ satisfies this, but is **bad**.

Universal Families of Hash Functions

A family of hash functions H (that map $U \to \{0, 1, \ldots, m-1\}$) is **universal** if for all distinct keys $x, y \in U$

$$\boxed{\Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}.}$$

Example

Let $h_{a,b}(x) = ((ax + b) \bmod p) \bmod m$ where p is a prime bigger than any key.

$$H = \left\{ h_{a,b} | a \in \{1, 2, \ldots, p-1\}, b \in \{0, 1, \ldots, p-1\} \right\}$$

Universal Families of Hash Functions

Let $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$ where p is a prime bigger than any key.

$$H = \{h_{a,b} | a \in \{1,2,\ldots,p-1\}, b \in \{0,1,\ldots,p-1\}\}$$

Theorem: $\boxed{H \text{ is universal}}$

$p = 6$

Proof: Choose any $x \neq y$.

Let $r = (ax+b) \bmod p$ and $s = (ay+b) \bmod p$.

$2 \cdot \underset{x}{2} \quad 2 \cdot \underset{y}{5}$

1. $r \neq s$ since

$4 = 10 \pmod 6$

$$x \neq y \Leftrightarrow x \neq y \pmod p \overset{a \neq 0}{\underset{p \text{ prime}}{\Leftrightarrow}} ax \neq ay \pmod p \Leftrightarrow r \neq s$$

2. For $x \neq y$, every pair (a,b) gives different (r,s) with $r \neq s$

since we can solve for (a,b) given (r,s).

b a

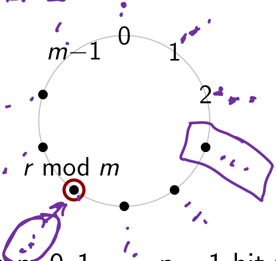
$\#(a,b)$ pairs $= \#(r,s)$ pairs $= p(p-1)$ so choosing (a,b) uniformly at random yields uniform random (r,s).

For a given value of r, of the $p-1$ possible values of s (since $s \neq r$), how many have $s = r \pmod m$?

H is universal (cont.)

For a given value of r, of the $p-1$ possible values of s (since $\boxed{s \neq r}$), how many have $s = r \pmod{m}$?

take p values mod m where do they go? 0, ... p-1



At most $\lceil \frac{p}{m} \rceil$ values s from $0, 1, \ldots, p-1$ hit this spot; minus 1 since $s \neq r$ (and r does hit this spot). Thus,

3. $\Pr_{h \in H}[h(x) = h(y)] = \Pr_{a,b}[r = s \pmod{m}] \leq \frac{\lceil p/m \rceil - 1}{p-1} \leq \frac{(p-1)/m}{p-1} = \frac{1}{m}$

□