Admin: PS #2 out

Reading: CLRS 11.1-3, 17

Outline:
- Computing a hash function
- Resizing a hash table
- Robin-Karp string-matching & “rolling hashes”
How to compute \( h(x) \)?

Lots of ways; here's one that's good

Assume \( x \) is an integer

Let \( m \) be hash table size

Let \( p \) be prime, \( p \geq m \) (ok if \( p = m \) if prime)

Pick \( a \): \( 0 < a < p \) \{ can choose randomly \}

Pick \( b \): \( 0 < b < p \)

Let

\[
 h(x) = \left( (a \cdot x + b) \mod p \right) \mod m
\]

Not needed if \( p = m \)

Example:

\[
 m = 1,000,000 \\
 p = 1,000,003 \\
 a = 314159 \\
 b = 271828
\]

If \( x = \text{"ATTGCATA"} \) treat as base-4 integer

If \( x = \text{"weather"} \) treat as base-26 integer

Note: can compute \( x \mod p \) as first step: \( h(x) = h(x \mod p) \)

Note: if \( p \) reasonably large, can use same \( a, b, p \)
with tables of different size \( m \)

See text for other methods (division method, multiplication method).
resizing a hash table (ref. chapter 17) 6.006
(also applies to resizing arrays in general...) rivest
imagine we are using chaining for l6.3
collision resolution 9/22/08

\[ m \text{ slots} \]

\[ m^2 \]

\[ n \text{ keys arranged in } m \text{ lists (some are empty)} \]

\[ \text{average list length} = \text{load factor} = \alpha = n/m \]

\[ \Rightarrow \text{ want } m = \Theta(n) \text{ at all times} \]

\[ \Rightarrow \text{ don't know how large } n \text{ might get to... what } m \text{ to use?} \]

\[ \text{... } m \text{ too large: costly to create, wasteful} \]

\[ \text{... } m \text{ too small: slow to search, as lists get long} \]

\[ \Rightarrow \text{ want to dynamically adjust } m \text{ as appropriate} \]

\[ \alpha \geq 4/5: \text{ double table size} \quad \text{so } \alpha = 2/5 \text{ afterwards} \]

\[ \alpha? \quad 4/5 \quad 2/5 \quad 2/5 \quad 2/5 \]

\[ \Rightarrow 2/5 \leq \alpha < 4/5 \text{ always (assuming we are} \]

\[ \text{always inserting, never deleting)} \]
Analysis

"Amortized analysis"

- cost of one insert can be large, since we might have to copy entire table!

- So, let's look at cost \( T(n) \) for a sequence of \( n \) inserts; then \( T(n)/n \) is "average" (or amortized) cost per insert.

Suppose \( m=5 \) initially

\[
\begin{align*}
T(1) &= 1 \\
T(2) &= T(1) + 1 = 2 \\
T(3) &= 3 \\
T(4) &= 3 + 1 + 4 = 8 \quad \text{(copy 4)} \\
T(5) &= 9 \\
T(6) &= 10 \\
T(7) &= 11 \\
T(8) &= 11 + 1 + 8 = 20 \quad \text{(copy 8)} \\
\end{align*}
\]

if \( n=2 \):

\[
T(n) = n + (4+8+16+\ldots+n) 
\leq n + 2n = 3n
\]

so average cost per insert is \( \leq 3 \) per insert.

(worst-case in an amortized sense: \( n \) inserts never take more time than \( 3n \))

Or:

When we insert an element, we pay 1 now, and set aside 2 "units of work" to do later. (savings)

When savings account is big enough to copy entire table over, do it! (Illustrate)
What about deletions?

If $\alpha \leq \frac{1}{5}$, halve table size (so $\alpha$ becomes $2/5$).

Can do both insertions & deletions; amortized cost per operation is still $\leq 3$.

(Example: if we decrease $n$ from $m \cdot \frac{2}{5}$ to $m \cdot \frac{1}{5}$, we have done $m/5$ deletions. We pay $m/5$ for those deletions and put $2 \cdot (m/5)$ in the bank. That more than pays for putting those remaining $m/5$ elts in a new, smaller table.)

Exercise: why don't we halve table size when $\alpha < \frac{2}{5}$? (instead of $\frac{1}{5}$)?
String-matching (Robin-Karp) & "rolling hash"  

Given: pattern $P[1..m]$ \{of chars\} \text{ text } T[1..n]$ \text{ does } P \text{ occur in } T? 

e.g. find "ATG" (corresponds to "start codon") in "AATCGC...

Idea:
1. compute hash$(P)$
2. for each length-$m$ window of $T$
   (i.e. for each $T[i..i+m-1]$)

\[
\text{A A T C G C}
\]

compute hash$(T[i..i+m-1])$ & compare to hash$(P)$

if $=$: check to see if they really match
if $\neq$: move on to next $i$

Want hash function s.t. we can go from one window to next easily! Want to be able to compute $T[i+1..i+m]$

Easily, given $T[i..i+m-1]$

"rolling hash"
How?

(continuing DNA example) \([A=0, C=1, G=2, T=3]\) 

pick prime \(p = 1009\); hash = string \(\mod p\)

suppose \(m = 9\)

\[
T[i...i+8] = \text{CTATTACGT}
= 130330123_y \quad \text{(base 4)}
= 184091_{10} \quad \text{(base 10)}
= 453 \mod p
\]

what is effect of dropping high-order \(C\)?

\(C\) in high order has value 1 \(\cdot 4^8\)

\[
= 1 \cdot 960 \mod p \quad \overset{\text{precompute!}}{\longrightarrow}
\]

so hash (TATTACGT) = 453 - 960
\[
= 502 \mod p
\]

What is effect of shifting left & appending "G" on right?

multiply by 4 & add 2

hash (TATTACGT6) = 4 \cdot 502 + 2
\[
= 1001 \mod p
\]

one subtract
one multiply \} to move over to next window
one add