Lecture 9, Thursday 5/3/01

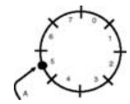
## Choosing Hash Functions

- Mostly black magic…
  division method: h(k)=k mod m
  - » Do not use $m=2^p$ (will not use all the bits)
  - » choose m=prime not too close to power of 2 or 10.
- Multiplication method: $h(k)=\lfloor m(kA-\lfloor kA\rfloor)\rfloor$
  - » choose $m\neq 2^p$, 0 < A < 1, not too close to 0 or 1.
  - » If $m=2^p$ , then all we do is scramble by multiplication, and choose p bits to the left of binary point.
  - » Another explanation: consider going from k-1 to k.

82

## More multiplicative method

- Example: m=8:
  - » each time k incremented:
    - go A around the circle,
    - Read off sector number.
  - » Note what happens if A=.5 or $1/2^p$.




83

## Universal Hashing

- Biggest problem with hash function:
  There is always an adversarial sequence that "kills" it !
- Can not choose truly random function - m to the power of keys different functions. Too much storage !!!
- We need a small family H of hash functions, such that, for any input, only small percentage of these functions are "killed".
- Existence of such family ? Size ?
  First, lets look at properties: What if h() is truly random ?
  Then:

$$\Pr[h(x)=h(y)]=\sum_{i=1}^{n}\Pr[h(x)=h(y)=i]=m\frac{1}{m^2}=\frac{1}{m}$$

84

## Universal Hashing

- Assume we found a family H that satisfies the requirement
  that if h ∈ H is chosen at random,
  then, for any x&y: $\Pr[h(x)=h(y)]=\sum_{i=1}^{n}\Pr[h(x)=h(y)=i]=m\frac{1}{m^2}=\frac{1}{m}$

  (note that we are given x&y and h chosen independently of x&y)
- Claim: this property is good enough for our purposes !

$C_x =$ total # collisions with $x$ (random variable !)

$$\lambda_{yz}=\begin{cases}1 & if\ h(y)=h(z)\\ 0 & otherwise\end{cases}$$ indicator random variable

$$C_x=\sum_{\substack{y\in T\\ y\neq x}}\lambda_{xy}$$

By our assumption: $E[\lambda_{xy}]=1/m$

$$\Rightarrow\ E[C_x]=E\left[\sum_{\substack{y\in T\\ y\neq x}}\lambda_{xy}\right]=\frac{n-1}{m}\leq\alpha$$

← This is enough for 1+α expected performance

85

## Construction Universal Hash Functions

- Need: for any x&y, proportion of functions in H that map
  both x and y to the same slot is 1/m.
- Take m prime.

  $Input: x=\langle x_0,x_1,\cdots,x_r\rangle,\quad \forall i, x_i<m.$

  $Let\ \ a=\langle a_0,a_1,\cdots,a_r\rangle,\ a_i\in[0,m-1]$ chosen uniformly at random.
- Define a function for each possible choice of *a*.

  $$h_a(x)=\sum_{i=0}^{r}a_i x_i \bmod m$$
- Claim: the family H is universal.

86

## Proving Universality

- Total number of functions in H: $m^{r+1}$
- Given particular x and y, what proportion of these functions map h(x)=h(y) ? WLOG, assume $x_0\neq y_0$
- Choose $a_1$, $a_2$, $a_3$, etc first. There are $m^r$ choices.
  Now we need to choose $a_0$ , to make h(x)=h(y):

$$a_0(x_0-y_0)=-\sum_{i=1}^{r}a_i(x_i-y_i)\bmod m$$

$But\ x_0\neq y_0\ \Rightarrow (x_0-y_0)\ invertible \bmod m$

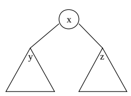
$\Rightarrow$ There is **only 1 solution**.

- Thus, total number of functions such that h(x)=h(y) is $m^r$ , exactly the right proportion.

87

# Binary Search Trees (Chapter 13)

- In addition to insert/delete:
  - » Heaps supported min/max.
  - » Hashing supported search.
  - » What if we want both min/max/search, and also pred/succ ?
- Binary Search trees:

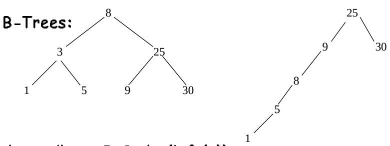



$\forall a \in$ left tree $key(a) \leq key(x)$
$\forall a \in$ right tree $key(a) \geq key(x)$

88

# Examples

- Legal B-Trees:




- In-Order walk: InOrder(left(x))
  print(x)
  InOrder(right(x))
- Note that given B-tree, can output sorted in O(n) time !
  Gives lower bound on constructing B-Tree.
  (Compare with Heap !)

89