CS2351
Data Structures
Lecture 18:
Hashing II
About this lecture

- The Hashing Problem
- Hash with Chaining
- Hash with Open Addressing

- Choosing a good Hash Function
  ** Universal Hash Function
Hash Function for Chaining
What is a good hash function?

• A good hash function should satisfy the simple uniform hashing assumption:

  1. Each element of $U$ is equally likely to be mapped into any of the $m$ entries
  2. Also, it is independent of where any other element is mapped to

• However, it is difficult to check, as we often don’t know the key distribution
What is a good hash function?

- Sometimes we do know ...

**Ex:** Suppose keys are random real numbers drawn independently and uniformly from [0,1)

⇒ The hash function

\[ h(k) = \lfloor km \rfloor \]

satisfies the simple uniform hashing
What is a good hash function?

• In practice, we use heuristics to create hash functions
  • May not satisfy simple uniform hashing, but performs well

• A general idea is to avoid the hash value to be dependent on the patterns that might exist in the key
The Division Method

- In division method we map key $k$ into one of the $m$ slots by:
  \[ h(k) = k \mod m \]

  \text{Ex: if } m = 12, k = 100 \implies h(k) = 4 \]

- Should avoid $m = \text{power of 2}$ (why?)

- A prime not close to power of 2 is usually a good choice

  \text{Ex: } n = 2000, \text{ we may choose } m = 701
The Multiplication Method

• In multiplication method we compute the hash value in 3 steps
  1. Fix a constant \( A \) from \((0,1)\)
  2. Multiply the key \( k \) with \( A \) and take the fractional part
  3. Multiply the fractional part with \( m \), and take the floor of the result

• In summary: \( h(k) = \lfloor m \{ kA \} \rfloor \)
  where \( \{ x \} \) denote the fractional part of \( x \)
The Multiplication Method

• Unlike the division method, we don’t need to avoid certain values of $m$ here.

• In fact, we often set $m$ to be a power of 2 (say $m = 2^p$) \(\Rightarrow\) easier computation.

Ex: Suppose the word size of our computer is $w$ bits.
If we further restrict $A$ to be a real of the form $s/2^w$ for some integer $s$, then ...
The Multiplication Method

Ex (cont):

Then to compute the desired hash value, we can:

1. Obtain $k \times s$ as a $2w$-bit integer
2. Retain the last $w$ bits of $k \times s$
3. Retain the first $p$ bits of the result of part 2

• In C: \[ h = (k \times s) >> (w - p); \]
Remark

• Knuth suggests
  \[ A = \left( \sqrt{5} - 1 \right)/2 = 0.6180339887\ldots \]
  is likely to work well

• Thus when \( w = 32 \), we try to choose
  \[ s = 2654435769 \]
  which is the integer closest to \( A \times 2^{32} \)
Remark

• Most hash functions assume the universe of keys to be integers
• If keys are not integers, we may convert them to integers
• Ex: Given a string \( pt \), we may look at it as a radix-128 integer
  \[ pt_{(128)} = 112 \times 128 + 116 = 14452 \]
• We shall assume all keys are integers
Hash Function for Open Addressing
What is a good hash function?

• In open addressing, our focus is to create hash function of the form $h(k, j)$ such that the values $h(k, 0), h(k, 1), \ldots, h(k, m-1)$ form a permutation of $[0, m-1]$

• We are going to describe three common techniques for creating such functions
  • Unfortunately, they don’t satisfy the uniform hashing assumption …
Linear Probing

• In linear probing we need an auxiliary hash function
  \[ h' : U \rightarrow \{ 0, 1, \ldots, m-1 \} \]

• Based on \( h' \), the desired hash function is simply:
  \[ h(k, j) = ( h'(k) + j ) \mod m \]

• Any disadvantage of this scheme?
Quadratic Probing

- In quadratic probing we also need an auxiliary hash function
  \[ h' : U \rightarrow \{ 0, 1, \ldots, m-1 \} \]
- Based on \( h' \), the desired hash function is:
  \[ h(k, j) = (h'(k) + aj + bj^2) \mod m \]
  for some fixed \( a \) and \( b \)
- We need to choose \( a \) and \( b \) carefully otherwise cannot get a permutation
Double Hashing

• In double hashing we need two auxiliary hash functions $h_1$ and $h_2$ where
  
  $h_1 : U \rightarrow \{ 0, 1, ..., m-1 \}$

• The desired hash function is:
  
  $h(k, j) = (h_1(k) + j \cdot h_2(k)) \mod m$

• We need $h_2(k)$ to be relatively prime to $m$
  
  • Method 1: $m = 2$ power, $h_2(k) = \text{odd}$
  
  • Method 2: $m = \text{prime}$, $0 < h_2(k) < m$
Double Hashing

Ex (Method 1):
\[ m = 65536 \]
\[ h_2(k) = (2 \times k) + 1 \]

Ex (Method 2):
\[ m = 701 \]
\[ h_2(k) = 1 + (k \mod 700) \]