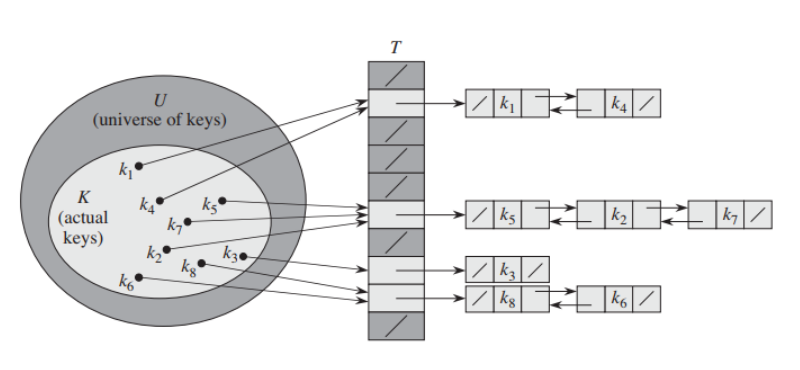
PROBABILISTIC ANALYSIS AND HASHING

Recitation Week 4

Hash Tables



- Many applications require some concept of a dictionary
  - A dictionary stores elements and can grow (or shrink) over time
    - Search
    - Insert
    - Delete
- Some dictionaries, such as the hash table use addresses to expedite this process
  - A hash function h takes the universe of possible keys and associates them with a value using some function
  - A key k is then stored at location $h(k)$ in some auxiliary array
    - The array is usually much smaller than the universe of keys – Collisions
  - If two elements would be stored at the same location, we chain
    - So each index of the auxiliary array points to a linked list
  - The performance of an algorithm using a hash table is directly related to the length of the chains

WARM-UP – HASHING AND RANDOMNESS

- TWO PROBLEMS TO THINK ABOUT
- WHAT IF OUR HASHING FUNCTION GOES WRONG?
  - SUPPOSE n ITEMS HAVE BEEN INSERTED INTO A HASH TABLE WITH m SLOTS. WHAT IS THE WORST-CASE TOTAL RUNNING TIME OF SEARCHING FOR EACH OF THESE ITEMS?

- "FINDING A 1"

  INPUT: A HALF FULL HASH TABLE WITH UNKNOWN HASHING FUNCTION

  OUTPUT: THE INDEX OF A USED HASH TABLE ENTRY

PROPOSE A DETERMINISTIC ALGORITHM AND ANALYZE ITS WORST-CASE RUNNING TIME AND THEN PROPOSE A RANDOMIZED ALGORITHM THAT YOU THINK WOULD BE BETTER

RANDOMIZED ANALYSIS AND HASHING

- IN HASHING, WE ARE CONCERNED WITH COLLISIONS AS IT SLOWS DOWN HASH TABLE OPERATIONS AND FUNCTIONALITY
- SAY WE HAVE A UNIFORM HASHING FUNCTION AND WE INSERT UNIFORMLY AND INDEPENDENTLY GENERATED ITEMS INTO OUR HASH TABLE WITH m SLOTS. HOW MANY INSERTIONS ARE REQUIRED BEFORE WE EXPECT A COLLISION?
  - GUESS?
- CAN WE MODEL THIS USING INDICATOR RANDOM VARIABLES AND COMBINE THEM IN SOME WAY?
- $X_{ij} = I\{$ITEMS i AND j HASH TO THE SAME LOCATION$\}$
- LET X BE THE SUM OVER i AND j. WHAT DOES X REPRESENT?

X is the number of pairs of elements with the same hash

$$X = \sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}$$

How can we use this random variable to solve our problem?

EXPECTING A COLLISION

- $X_{ij} = I\{$ITEMS i AND j HASH TO THE SAME LOCATION$\}$
- $X = \{$THE NUMBER OF PAIRS OF ELEMENTS WITH THE SAME HASH$\}$

$$E[X] = E\left[\sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}\right]$$

$$= \sum_{i=1}^{k} \sum_{j=i+1}^{k} E[X_{ij}]$$

$$= \binom{k}{2} \cdot \frac{1}{m}$$

$$= \frac{k(k-1)}{2m}$$

What does $E[X_{ij}]$ equal?

When this exceeds 1, the expected number of pairs is greater than 1.
$k = \sqrt{2m} + 1$

HOW MANY ELEMENTS WITH A GIVEN HASH?

- Recall the chaining paradigm uses a linked list at each element of the hash table. Collisions are resolved by appending the element to the end of the linked list.
- We would like to have very short linked lists for chaining to work well
- Suppose have a uniform hashing function for a hash table with m slots. We use chaining to resolve collisions. If n elements are generated independently and uniformly at random and inserted into our hash table, what is the expected length of the linked list at table entry 3?
  - What are our indicator variables? What are we looking to solve?
  - $X_j = I\{$The j-th element hashes to entry $3\}$
  - Now we can combine our indicator variables using Linearity of expectation
  - $E\left[\sum_{j=1}^{n} X_j\right] = \sum_{j=1}^{n} E\left[X_j\right] = \frac{n}{m}$

Chain Hashing and Probabilistic Analysis

- Say we have a uniform hashing function and we insert uniformly and independently generated items into our hash table with m slots. We resolve collisions with chaining.
- How many elements do we need to hash, in expectation, so that there is at least one element in hash position 3?
- We know that, in expectation, $\frac{n}{m}$ elements are hashed to position 3 when we insert n elements. Can we use this result to solve our desired problem?
- $\frac{n}{m} \geq 1 \Rightarrow n \geq m$

- Follow-up Question: How many elements do you need to insert, in expectation, so that there is at least one element at every hash position?

FULL HASH TABLES USING CHAINING

- MAIN QUESTION: IF WE USE CHAIN HASHING, HOW MANY ELEMENTS DO YOU NEED TO INSERT, IN EXPECTATION, SO THAT EVERY HASH TABLE LOCATION IS USED?
  - HINTS
  - WHEN SELECTING A RANDOM VARIABLE(S), THINK: HOW CAN I COMBINE THEM? WHERE IS THERE ANY RANDOMNESS?
  - WHAT'S THE PROBABILITY OF HASHING THE FIRST INPUT TO A "NEW" LOCATION?
  - ONCE THE FIRST DISTINCT LOCATION HAS BEEN USED, WHAT'S THE PROBABILITY OF HASHING TO A NEW LOCATION ON THE FIRST SUBSEQUENT PULL?
- $X_i = I\{$THE i-TH DISTINCT HASH TABLE INDEX IS FILLED ON A THROW (GIVEN THAT $i-1$ ENTRIES WERE FILLED PRIOR)$\}$
- $E[X_i] = \frac{m-i+1}{m}$
- WHAT DOES THIS MEAN ABOUT THE EXPECTED NUMBER OF INSERTS NEEDED TO USE THE i-TH DISTINCT HASH ENTRY AFTER HAVING USED $i-1$ ENTRIES PRIOR?

Full Hash Tables Using Chaining

- $X_{ik} = I\{$A new hash table entry is used on the k-th input after having used i-1 distinct entries$\}$
- $E[X_{ik}] = \frac{m-i+1}{m}$
- Let Y_i be the number of inserts required to use the i-th distinct entry (after having used $i-1$ distinct entries prior). How does Y_i relate to X_{ik}?
  - $E[Y_i] = \frac{1}{E[X_{ik}]}$
- To find the total insertions needed to fill all m distinct hashes, we can sum over the Y_i.

$$E\left[\sum_{i=1}^{m} Y_i\right] =$$

$$\sum_{i=1}^{m} E[Y_i] =$$

$$\sum_{i=1}^{m} \frac{m}{m-i+1} =$$

$$\Theta(m \cdot \log(m))$$