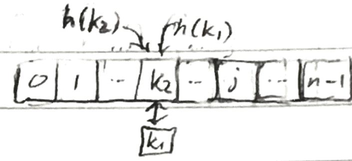
CSC263 Wednesday Lecture : Hashing　　　　　　　　　Jan 31, 2018

- Dictionary
  - S : n elements with keys from universe $U = \{0, 1, 2, 3, 4, \ldots, U\}$
  - OPS : insert, delete, search
  - Assumption that $|U|$ is small. Then...
    - → store S in a direct access table $T[0, 1, \ldots, U]$, use keys as index for location in array to store item
  - If $|U|$ is big (note: U is just the possible universe of keys, NOT the number of keys you need to lable items.)
    - → store S in a hash table T:



  - n : # of keys to be stored
  - m : size of the hash table
  - Hash function : $h(k) = i$, $h \in U \to i$: index of T
    - ↳ Hash function may map multiple elements from universe to index, causing collisions, solution: at index, store a linked list that acts as a stack. Putting second item pops it on top of the stack.
  - INSERT : $O(1)$
  - SEARCH (k) : $h(k) = i$, transverse linked list at $T[i]$ to find k
    - ↳ worst case : all n items at same index, transverse whole linked list of length n, $\therefore O(n)$
    - ↳ expected time is $\Theta(1)$ under the following assumptions
      1. SUHA : Simple Uniform Hashing Assumption;
         - the hashing of keys into slots has uniform probability distribution
         - $P(h(k) = i) = 1/m$; key hashings are independent events
      - then expected chain length after n insertions into m spaces on the hash table, $E(l_i) = n/m = \alpha$, load factor
      - then total chain length is n → $E(l_0 + l_1 + \ldots + l_{m-1}) = n$; $E(l_0) + \ldots + E(l_{m-1}) = n$; $E(n_i) = E(n_j)$; $m \times E(n_i) = n$
      - then SEARCH $\in O(\alpha + 1)$
      - if $n \in O(m)$ (ie n & m within constant factor) we can say SEARCH $\in \Theta(1)$
- defining a good hash function is an art