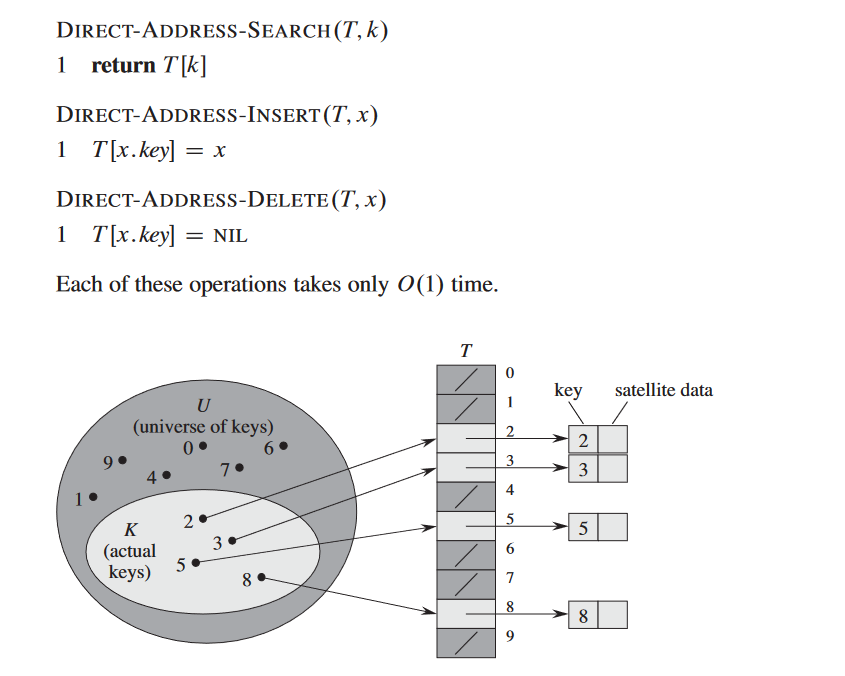
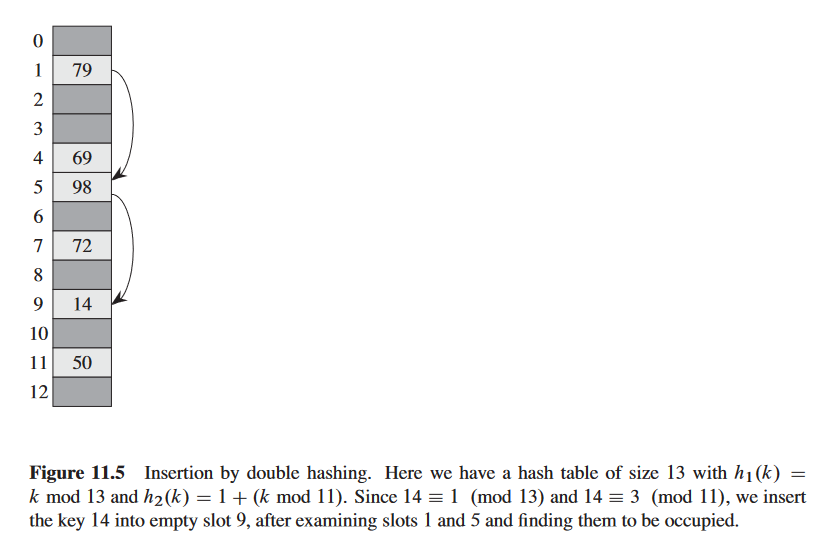
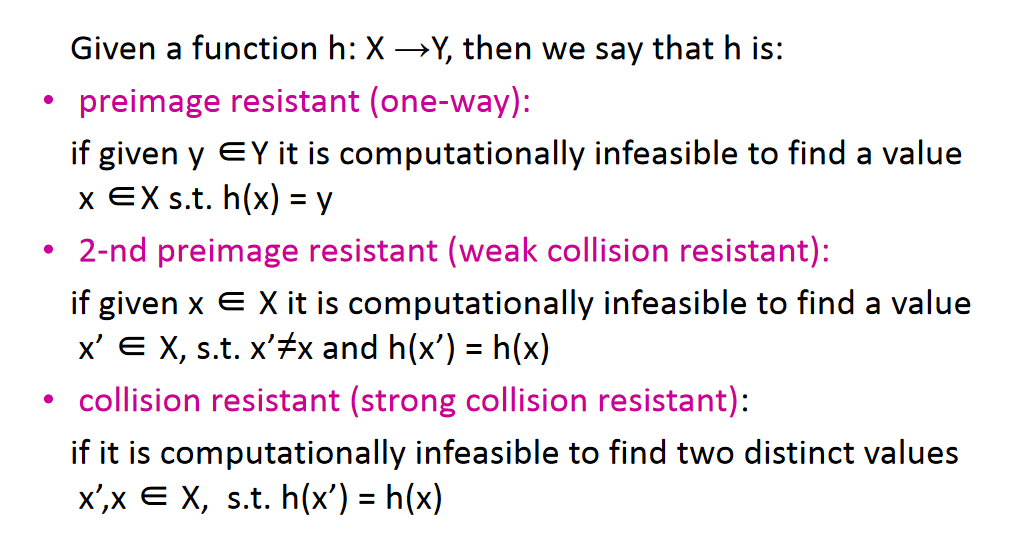
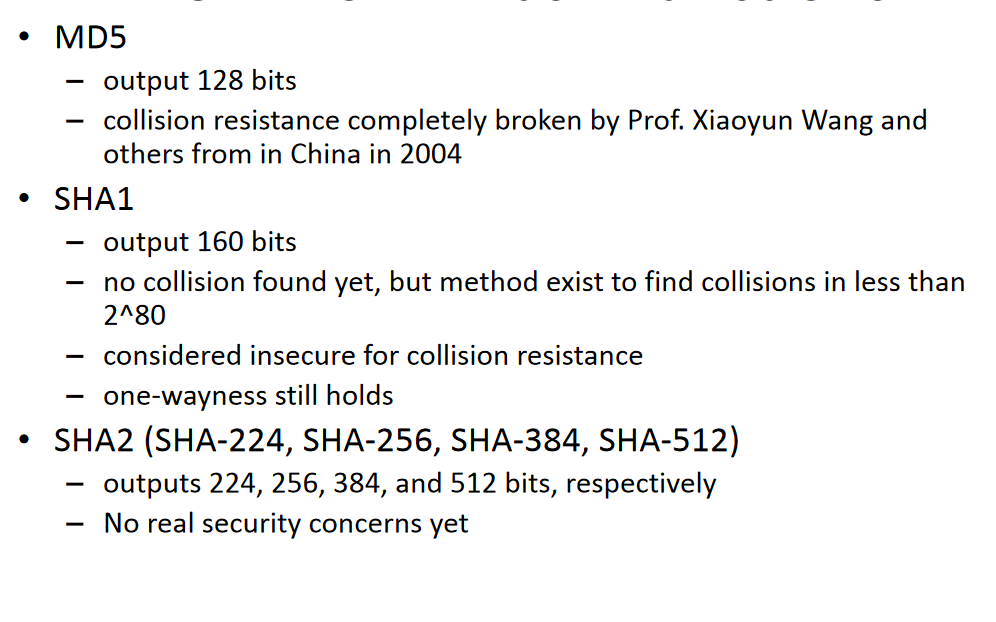
Midterm 2 - CPSC 535

# Hash Tables and Functions

1. Hash Tables Basics
   1. Is an Indexed structure array
   2. O(1) search time complexity
   3. Order matters!
   4. Functions
      1. Insert, search, delete
   5. Each element has key computed by hash function
   6. Hash table: domain of values
2. Hashing  
   
   1. Hash function – function used to map data of arbitrary size to data of fixed size
      1. Must map search key to index
      2. Values returned by hash functions are
         1. Values, codes, sums, or just hashes
      3. Only hints at records location but not the exact locale
      4. Compresses messages of arbitrary lengths to a fixed length
         1. Must be in O(N) time
      5. Merkle-Damgard transform is used to convert data to hash functions
      6. Example: hash function that strings integers in [0,2,32,-1]
      7. Has many-to-one function
      8. Provides time efficient solutions
   2. Good hash functions required to satisfy
      1. Determinism – value must always generate same hash value
      2. Uniformity – map expected inputs over output range (all same probability)
      3. Defined range – hash must have fixed size
         1. Achieved by breaking variable input into chunks of a specific size
      4. Data normalization – inputs that are considered equivalent must yield same hash value. Upper case ALL the letters!
      5. Non-invertible – not possible to reconstruct input datum x from hash value h(x) alone
3. Collision
   1. When different keys with same hash value exist, collision happens
   2. Solutions are chaining and open addressing
   3. Chaining

|  |  |
| --- | --- |
|  |  |

* + 1. Solution
       1. All elements that share same hash key placed on linked list
       2. Entry on hash table is either NIL or pointer to first element
    2. Analysis
       1. Input: n = number of leemnets on hash table  
           m = number of slots  
            = load factor
       2. Operations
          1. Insert = O(1)
          2. Deletion = O(1)
          3. Searching =  wosrt = O(N)
       3. By not using pointers, save memory used to increase number of slots in hash table
  1. Open Addressing (not on exam)
     1. Methods include
        1. Probing: linear, quadratic
           1. Linear: given key k, probe T[h’(k)] using hash function. We do this one by one  
              
           2. Quadratic: like linear except instead of one by one we go one by squared  
              
        2. Hashing: Double, universal, cuckoo
           1. Double:   
              
           2. Universal and cuckoo have their own explanations but, for know, we are only focused on three methods
  2. Cryptographic hash Functions
     1. These are the 3 security properites
     2. Collision resistance: cannot find two distinct values with same hash value thus H(x) = H(y)
     3. Hiding: if r chosen from pobaility, it is infeasible to find x
     4. Puzzle-friendly: for every output vlue y, if k is chosen, it is infeasible to find x H(k||x) = y in given amount of time
  3. Collision Resistnace
     1. 
     2. Examples of collision resistance
        1. Hash function with 256-bit output size
           1. Pick 2^256+1 distinct values
           2. Compute hashes
           3. Check if two outputs are equal
        2. At least one collision but to find it have to compute hash function 2^256 for worst-case and 2^128 for average-case
     3. Examples of hash function with hiding property
        1. High min-entropy = distriction is very spread out
        2. Nonce = value that can only be used once
        3. If nonce r is chosen uniformly among strings that eacha re 256 bits long, any string was strong with probability of 1/2^256 value
     4. Exmaples of puxxle friendly
        1. Seach puzzle:
           1. Has a hash function H with value r chosen from high min-entropy distribution nand target set Y, |Y| << |X|
           2. Solution is value x such that 
           3. Size Y determines how hard puzzle is
           4. Puzzle-friendly = no solving strategy is better than trying random values
     5. Well-known hash functions
        1. 

# Binary Trees and B-Trees

# Text-Search Data Structures and Tries

# Dynamic Programming

# Greedy Algos and Huffman Coding