CAS CS 365
Lec 17

Hash functions

1. Hash functions
    a. Hash
        i. Data struct
            1. HashMap, Dictionary in python
        ii. Is Q inside S? (subset question)
            1. Problem: need to compare every characters and there is redundancy
            2. Possible solution:
                a. hash(sum of ASCII code of S modulo by 100) and compare with hash of 4 characters in Q
                b. Removes some redundancy since H(AAAB) is similar to H(AAAC) → simply change B to C
    b. Setting: we have a large universe U of keys that we wish to map to a range [m] = {0,...,m-1}
    c. A truly random hash function h: U →[m] assigns an independent uniformly random variable h(x) to each key in x
    d. How many bits do we need to represent such a function
        i. $|U|*\log_2(m)$
        ii. Independence between the keys (assignment of key1 is not dependent upon assignments of other keys)
        iii. But need same space on the universe → does not make sense
    e. Why don't we use some of the bits of the keys?
        i. example) divide universe into 10 bits → 2^10 = 1024
        ii. No longer uniformly random
        iii. When the data are real, there are many similarities between them → many collisions (not a good hash function)
        iv. Good hash function should have the quality that even though the input is very similar, the hashed value must look very different
    f. A hash function h:U → [m] is a random variable in the class of all functions U → [m]
    g. Three important aspects of a hash function are:
        i. Space
            1. $|U| \log_2 m$ is prohibitive where universe U is big
        ii. Speed: time needed to calculate h(x) given x
            1. h(x) x in U should be fast

iii. scattering/properties of the random variable or Collisions
    1. As few as possible

h. Remark: These are not the only criteria, e.g., cryptographic hash functions
i. If Hash is cryptographic, it is usually slower
j. Family H of hash functions is uniform if choosing a hash function uniformly at random from H satisfies → use subsets (less space we need but we lose collision property)

$$\boxed{\Pr_{h\in\mathcal{H}}(h(x)=i)=\frac{1}{m}\text{ for all }i,x}$$

Family of all possible functions is prohibitive to use

k. Question: can you give a family of hash functions that is trivial yet uniform?
    i. $H = \{h_0, \ldots h_{m-1}\}$
    ii. $h_i(x) = i$ for all $i = 0\ldots m-1$ and any x
    iii. Absolute collision but it is uniformly random
l. A family of hash functions H is universal if for any two items in the universe (if there is two values that have the same hashed value, it should be pairwise independence)

$$\boxed{\Pr_{h\in\mathcal{H}}(h(x)=h(y))\le\frac{1}{m}\text{ for all }x\neq y}$$

m/m^2 = 1/m

m. A family H of hash functions is k-wise-independent if for any sequence of k disjoint keys and any sequence of k hash values (not necessarily disjoint)

$$\boxed{\Pr_{h\in\mathcal{H}}(h(x_1)=i_1,\ldots,h(x_k)=i_k)=\frac{1}{m^k}\text{ for all distinct }x_1,\ldots,x_k\text{ and }i_1,\ldots,i_k}$$

n. Remarks
    i. For any fixed key x, and h chosen uar from H, h(x) is a uar variable in [m]
    ii. For any k fixed distinct keys, the hashes are independent random variables
o. Suppose m = 1000. What do you think of the following hash function

$$h(x) = x \bmod 1000$$

    i. There is a pattern, not independent

2. Universal Hashing
    a. Let p be a large prime, p > [U]
    b. For any integers

$$a \in \{1,\ldots,p-1\}=[p]^{+},\ b\in\{0,1,\ldots,p-1\}=[p]$$

Define $h_{a,b}(x) = (ax+b) \bmod p \bmod m$

c. Let $\mathcal{H} = \{h_{a,b} \mid a \in [p]^+, b \in [p]\}$ be the set of all p*(p-1) such functions.

d. Theorem: H is universal

   i. If a and b are random, it is universal

   ii. $$\boxed{\Pr_{h \in \mathcal{H}}(h(x) = h(y)) \leq \frac{1}{m} \text{ for all } x \neq y}$$

   iii. Space required: a and b values → number of bits required for a and b → log(p^2-p) ~ log(p^2) = 2log(p)

e. Parameters a,b are also called salt

3. K-wise independent hashing

   a. More generally, to perform k-wise independent hashing, we generalize the previous idea to polynomials

   $$\boxed{h_{a_{k-1},...,a_0}(x) = \left(\sum_{i=0}^{k-1} a_i x^i \mod p\right) \mod m}$$

   h(x) = (a_{k-1}*x^k-1 + … + a_1x + a_0) mod p mod m

   b. What is the space requirement to store such a function?

      i. K numbers

   c. How do we decide the K value we need to achieve “enough randomness”?

      i. Depends on the actual problem

   d.