Hashing, Hashtables

Example

- 100,000 students with unique roll nos. (e.g. JEE).
- A common requirement is finding all students satisfying a certain condition. (Ex. with centre as Kanpur).

Solutions

- Store in an array
  - Without any organization.
  - Broken into several sub-sets.
- Store as <key, value> pairs.

Hashtables

- Dynamic set of <key, value> pairs with *search*, *insert* and *delete* operations.

  Ex 1. Compiler: Identifiers in a program

  Ex 2. Spell checking
- Generalization of an array. In array, access, insert, delete operations O(1). Access through an index, which acts like a key.
- Hashtable: operations are O(1) on average. Access directly through the key.

Implementation

- Store in array A using A[hash(key)].
  - Function hash – hashing function i.e. Key→Integer mapping.
  - hash(k1)=hash(k2) for two distinct keys k1, k2 – collision. Ideally collision should be 0.
  - Colliding keys stored as chains (i.e. lists) in the same location.
- Size of key space.

  Ex. If keys from (0, 1, ... m-1) then can be stored in direct-address-table. Not possible if key space very large but only small part is actually used. Then hash function used.

hash function

- hash: each key should equi-probably hash to any of the slots (say m slots) independent of the hash value of any other key.
- Keys usually interpreted as natural numbers. Any other type (e.g. string) mapped to natural nos.

Division method

- hash(k)=k mod m, where m is suitably chosen.
- Avoid $m=2^j$, since hash(k) is just the j lower order bits of k. Good choice: a prime not close to power of 2.

  Ex. 2000 elements, with collision list size limited to 3. Then m=701 is a good choice.

  Prime near 2000/3 but not close to power of 2. So hash(k)=k mod 701.
- Can waste space if m not suitable.