Hash Tables

Hash tables are generally made up of three components:

1. A **hash function**, $\boldsymbol{f(k)}$ or $\boldsymbol{h(k)}$. The hash function transforms a **key** into an integer (its index in array)
2. An **array** (the hash *table*)
3. The set of **values** associated with each **key.**

Assume $\boldsymbol{n}$ is the number of elements in the hash table, and $\boldsymbol{m}$ is size allocated to the underlying array

Goals of hash function:

•

•

Challenges:

Load factor: $\alpha = \frac{\#\ hashed\ items}{table\ size} = \frac{n}{m}$

Two strategies to handle collisions:

Separate chaining:

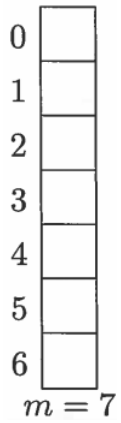
Open Addressing:

Separate Chaining example:

Given a hash function $h(k) = k \bmod m$

Insert the following keys into the hash table:

$8, 40, 36, 67, 22, 25, 70$



Separate Chaining:

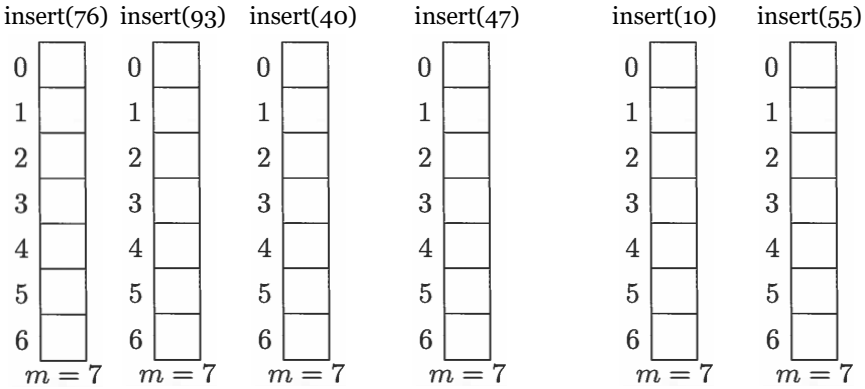
Pros:

Cons:

Open Addressing

Linear Probing: $h(k, i) = (hash(k) + i) \bmod m$

Open Addressing Exercise:



Searching for a key

Algorithm Find(k):
Input: the target key to search the hash table for
Output: the entry with the given key; null if key not found

$i \leftarrow h(k)$ % m
for $i \leftarrow 1$ to m **do**
  $e \leftarrow$ data[i]
  if e is *null* **then**
    return *null*
  end if
  if e's key is equal to k **then**
    return e
  end if
  $i \leftarrow (i + 1)$ % m
end for
return *null*
end

Search example:

Find entries with the following keys:

93, 8, 47, 12

Linear Probing:

Pros:

Cons:

Primary Clustering

Description:

Proposed solution: