Hash Tables

Problem: Storing a large number of elements (e.g. dictionary, symbol table)

Operations: Insert, Search, [Delete]

Solution: Use a linked list
Insert = $\Theta(1)$
Search = $\Theta(n)$
Delete = $\Theta(n)$

Better Solution

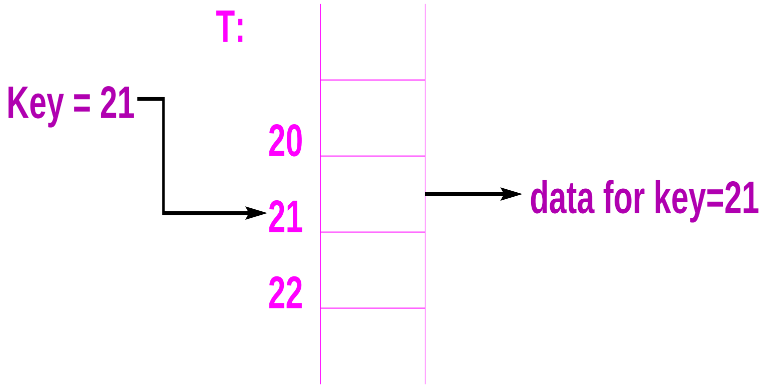
Better Solution: ____________________
$\Theta(1)$ operations
$\Theta(n)$ memory (at least)

Direct-Address Tables

If the number of possible keys is _______ and they are ________, then the table can be a BIG array.

Let the universe of m possible keys be U = {0, 1, .., m-1}.

Direct-Address Table T[0,..,m-1] is an array. Each slot (array element) corresponds to a unique key.



Operations

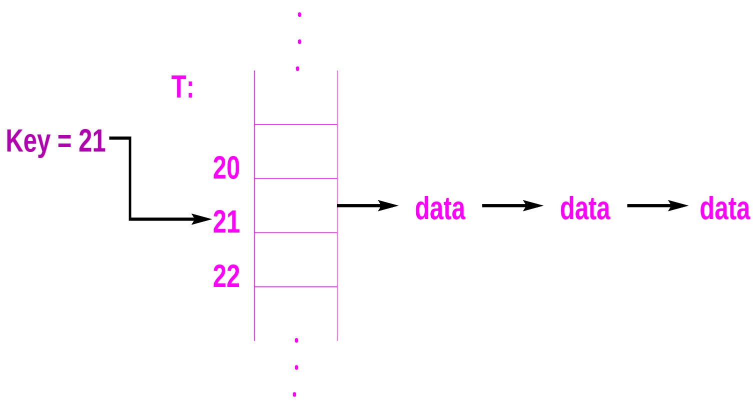
```
Insert(T,x)
   T[key(x)] = x

Search(T,x)
   return(T[key(x)])

Delete(T,x)
   T[key(x)] = NIL
```

What if the keys are not unique?

Solution 1: Insert implies Replace

Solution 2: ____

If we assume a uniform distribution over keys, a $\Theta(1)$ search is maintained.



If we can maintain $\Theta(1)$ performance for multiple entries for the same key, perhaps we can do the same while mapping multiple keys into the same array element.

In other words, use **Hash Tables**.

Hash Tables

Problem with Direct Addressing: ______________________________

For example, consider a compiler symbol table. Symbols here are up to 30 alphabetic characters.

$|U| = 26 \,.\, 26 \,.\, 26 \,.\, \ldots \,.\, 26 = 26^{30} = 2\text{x}10^{42}$ bits.
Note that 1 gigabyte is only 10^9 bits.

Let K = set of actual keys occurring.
For large |U|, |K| is typically $<<$|U|.

Define Table T of size |K|
(T is a **hash table**, where we have chopped up U).

Analysis

Memory: $\Theta(|K|)$
Performance: $\Theta(1)$ average case, $\Theta(n)$ worst case

Instead of key k being stored in slot T[k], it is now stored in slot __________.

The function h(k) is the hash function.
The value of h(k) is the hash value of key k.

Example

Consider an example where |U| = 100, $|K|$ = 10, and h(k) = k mod 10.

U = {0,..,99}

Problem

Collisions: Two keys hash to the same slot.
Reduce collisions by using a _______________ hash function.
However, collisions are still possible.

Collision Resolution by Chaining

Data corresponding to keys with same hash values are stored in a linked list (as shown in the figure above).

$$\text{Insert} = \Theta(1)$$

$$\text{Search} = \Theta(l)$$

where l is the length of the chain

$$\text{Delete} = \Theta(l)$$

for a singly-linked list

Analysis of Chaining

Let the **load factor** α be calculated as number of keys stored / number of slots = n/m. For our earlier example, $\alpha = \frac{100}{10} = 10$.

α represents the ______________________ of the chain.

The performance of Search is relative to the performance of the hash function computation and the length of the chain, or $\Theta(1 + \alpha)$, both for successful and unsuccessful searches.

Thus, if m is proportional to n, then α is a constant, and all operations are $\Theta(1)$.

Question:

Would it help to keep chains sorted?

In this case,

$$\text{Insert} = \Theta(1+\alpha)$$
$$\text{Search} = \Theta(1+\alpha)$$
$$\text{Delete} = \Theta(1+\alpha)$$

Asymptotically, ___________ This reduces constant on search, but increases constant for Insert. Delete is the same as before.

Basically, ___.

Hash Functions

Good Hash Functions:

- If key distribution P is known, then the hash function should satisfy

$$\sum_{k:\, h(k)=j} P(k) \;=\; \frac{1}{m} \quad \text{for } j = 0, .., m-1$$

- Heuristics
  - Design hash function such that similar keys map to different slots (e.g. name1, name2)
  - Hash value should be independent of data patterns

Division Method

$$\text{h(k) = k mod m}$$

k is a natural number
m is the number of slots

Choice of m

m should not be a power of _, because h(k) would be the p lowest-order bits of k ($m = 2^p$)

avoid powers of ___ for decimal keys, because not all digits will be used

good values include primes not too close to powers of 2

Example

n=100, want $\alpha = 3$

Ideally, m = 33 (not prime, so try m = 31).

However, 31 is close to $32 = 2^5$, so try m = 29 or m = 37 (select m = 37).

h(k) = k mod 37

Multiplication Method

h(k) = $\lfloor m(kA \ mod \ 1) \rfloor$, where 0<A<1

(kA mod 1) returns the ______________ part of kA.

In this case the choice of m is less critical. Typically choose a power of _ to simplify arithmetic.

However, the choice of A does matter. A recommendation is to use

$$A = \frac{\sqrt{5}-1}{2} = 0.6180339887...$$

The worst choice is __________, because in this case every key hashes to $\lfloor \frac{m}{2} \rfloor$ or 0.

Universal Hashing

Any fixed hash function will have $\Theta(n)$ worst case time.

Choose hash function ____________, independent of the keys to be stored.

Choice at ____________ prevents worst case behavior on multiple runs.

Suppose we want the hash function to uniformly distribute hash values over the hash table of size m.

Given h(x), we want P(h(x) = h(y)) = ____.

Universal Hash Functions

We want to select from a set of hash functions H with reasonable certainty that the above property is true.

Thus, the number of functions |f| in H such that h(x) = h(y) for x,y $\in$ U must satisfy

$$\frac{|f|}{|H|} = \frac{1}{m} \longrightarrow |f| = \frac{|H|}{m}$$

Definition: A ____________ collection of hash functions H contains exactly $|H|/m$ hash functions such that h(x) = h(y) for x,y ∈ U.

$$h_a(x) \;=\; \sum_{i=0}^{r} a_i x_i \; mod \; m$$

where key x = $< x_0, x_1, .., x_r >$ is decomposed into r+1 bytes
a = $< a_0, a_1, .., a_r >$, each chosen randomly from {0, 1, .., m-1}.

$H \;=\; \cup_a \{h_a\}$ is a universal collection of hash functions.

Thus, we want to randomly select "a" each time.

Open Addressing

All elements are stored in the hash table (no pointers).

If a hash slot is full, then _______ other slots using the ____________________ until a slot is found or no slot can be found (overflow).

The hash function now becomes ________, where i ranges over {0,1,..,m-1}.
h(k,i) returns the ith probe in the probe sequence.

The entire probe sequence must be a permutation of {0,1,..,m-1}.

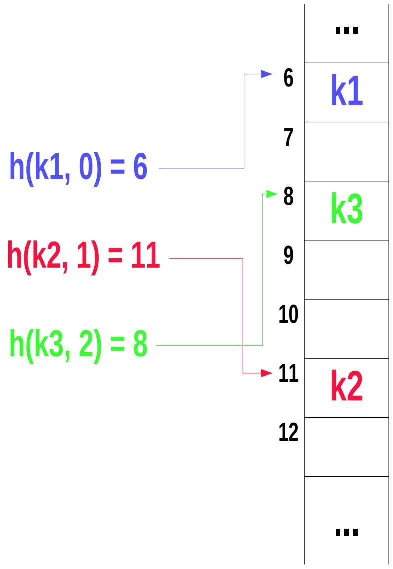
Pseudocode

```
Insert(T,k)
   i = 0
   repeat
```

```
        j = h(k,i)
        if T[j] = NIL
        then T[j] = k
             return j
        else i = i + 1
    until i = m
    error "hash table overflow"
```

Pseudocode

```
Search(T,k)
    i = 0
    repeat
        j = h(k,i)
        if T[j] = k
        then return j
        else i = i + 1
    until (T[j] = NIL) or (i = m)
    return NIL
```



Delete(k,i) is more difficult, because replacement by NIL may break a possible probe sequence.

Solution: replace deleted key by special symbol. However, in this case search time no longer depends on α.

Solution: use __________ when deletions are required.

Generating Probe Sequence

Uniform Hashing: Each key is equally likely to generate any of the m! permutations.

This is difficult in practice.

Linear Probing

Given an ordinary hash function h(k): h(k,i) = ____________________.

Sequence:

h(k)
h(k) + 1
h(k) + 2
...
m-1
0
1
2
...
h(k) - 1

There are only m ($<< m!$) possible sequences, but these are simple to compute.

Problem with Linear Probing:

Primary Clustering.

Long sequences of filled slots increase search and insert time.

Long sequences are more likely to get even longer.

Quadratic Probing

$$h(k,i) = (h(k) + c_1 i + c_2 i^2) \bmod m$$

- Only certain combination of c_1, c_2, and m use the entire hash table.
- $h(k_1,0) = h(k_2,0)$ implies $h(k_1,i) = h(k_2,i)$. This leads to secondary clustering.

- There are only m ($<< m!$) distinct probe sequences.

Example

$$h(k_1,i) = (h(k) + i + i^2) \bmod m, \; c_1 = c_2 = 1$$

In this example, the probe sequence is

h(k)
h(k) + 2
h(k) + 6
h(k) + 12
...

What if m = 20?

Double Hashing

$$h(k,i) = (h_1(k) + i\, h_2(k)) \bmod m$$

where h_1 and h_2 are auxiliary hash functions.

- If $h_2(k)$ and m have a common divisor, then not all of the table is probed.
- Let $m = 2^p$ and $h_2(k)$ = odd number
- m = prime number, $h_2(k) \in \{0, 1, ..., m-1\}$.
  For example, $h_1(k)$ = k mod m
  $h_2(k)$ = 1 + (k mod m') where m' = m - 1.
- Since each pair $h_1(k)$, $h_2(k)$ yields different probe sequences, the number of sequences is $\Theta(m^2)$, which is closer to ideal.

Example

Given input (9371, 3723, 9873, 9769, 8679, 1239, 4584), and a hash function h(x) = x mod 10, show the resulting open-addressed hash table using

1. linear probing

```
  +----+
0 |8679|  h(9371, 0) = 1
  +----+
1 |9371|  h(3723, 0) = 3
  +----+
2 |1239|  h(9873, 0) = 3 COLLISION!  h(9873, 1) = 4
  +----+
3 |3723|  h(9769, 0) = 9
  +----+
4 |9873|  h(8679, 0) = 9 COLLISION!  h(8679, 1) = 0
  +----+
5 |4584|  h(1239, 0) = 9 COLLISION!  h(1239, 1) = 0 COLLISIO
  +----+     h(1239, 2) = 1 COLLISION!  h(1239, 3) = 2
6 |    |
  +----+  h(4584, 0) = 4
7 |    |
  +----+
8 |    |
  +----+
9 |9769|
  +----+
```

2. double hashing with hash function h2(x) = (x mod 5)

   Note that 10 is a multiple of 5, so this is not an effective choice for a secondary hash function.

```
           h(9371, 0) = 1 + 0 = 1
  +----+
0 |    |  h(3723, 0) = 3 + 0 = 3
  +----+
1 |9371|  h(9873, 0) = 3 + 0 = 3 COLLISION!
  +----+     h(9873, 1) = ((3 + 1*(9873 mod 5)) mod 10) = 3
2 |    |
  +----+  h(9769, 0) = 9 + 0 = 9
3 |3723|
  +----+  h(8679, 0) = 9 + 0 = 9 COLLISION!
4 |4584|     h(8679, 1) = ((9 + 1*(8679 mod 5)) mod 10) = (9
  +----+     COLLISION!
5 |1239|     h(8679, 2) = ((9 + 2*(8679 mod 5)) mod 10) = (9
  +----+
6 |9873|  h(1239, 0) = 9 + 0 = 9 COLLISION!
  +----+     h(1239, 1) = ((9 + 1*(1239 mod 5)) mod 10) = (9
7 |8679|     COLLISION!
  +----+     h(1239, 2) = ((9 + 2*4) mod 10) = (9 + 8) mod 1
8 |    |     h(1239, 3) = ((9 + 3*4) mod 10) = (9 + 12) mod
  +----+     h(1239, 4) = ((9 + 4*4) mod 10) = (9 + 16) mod
9 |9769|
  +----+  h(4584, 0) = 4 + 0 = 4
```

Analysis of Open Addressing

Let n be the number of elements in the table,
m is the size of the table.

$$n \leq m$$

$$\alpha = \underline{\hspace{2em}} \leq 1$$

Assume uniform hashing (each sequence is equally likely).

Theorem 12.5

The expected number of probes in an unsuccessful search is at most 1/(1-α).

For example, if the table is half full, $\alpha = 0.5$, the number of probes is _.

If the table is 90% full, $\alpha = 0.9$, the number of probes is ___.

If α is constant, the performance of an unsuccessful search is ______.

Corollary 12.6

On average, the number of probes for Insert is $\leq$ 1/(1-α).

Theorem 12.7

The expected number of probes in a successful search is at most

$$\frac{1}{\alpha} ln \frac{1}{1-\alpha} + \frac{1}{\alpha}.$$

For example, if the table is half full, $\alpha = 0.5$, the expected number of probes is ____.

If the table is 90% full, $\alpha = 0.9$, the expected number of probes is ____.

If α is constant, the performance of a successful search is ______.

Applications