



# CS 225

## Data Structures

*March 13 – BTree Analysis*

*G Carl Evans*



## BTree Analysis

The height of the BTree determines maximum number of \_\_\_\_\_ possible in search data.

...and the height of the structure is: \_\_\_\_\_.

**Therefore:** The number of seeks is no more than \_\_\_\_\_.

*...suppose we want to prove this!*



## BTree Analysis

In our AVL Analysis, we saw finding an upper bound on the height (given  $n$ ) is the same as finding a lower bound on the nodes (given  $h$ ).

We want to find a relationship for BTrees between the number of keys ( $n$ ) and the height ( $h$ ).



# BTree Analysis

## **Strategy:**

We will first count the number of nodes, level by level.

Then, we will add the minimum number of keys per node (**n**).

The minimum number of nodes will tell us the largest possible height (**h**), allowing us to find an upper-bound on height.



# BTree Analysis

The minimum number of **nodes** for a BTree of order  $m$  **at each level:**

root:

level 1:

level 2:

level 3:

...

level  $h$ :



# BTree Analysis

The **total number of nodes** is the sum of all of the levels:



# BTree Analysis

The **total number of keys:**



## BTree Analysis

The **smallest total number of keys** is:

So an inequality about  **$n$** , the total number of keys:

Solving for  **$h$** , since  **$h$**  is the number of seek operations:





# BTree Analysis

Given  $m=101$ , a tree of height  $h=4$  has:

Minimum Keys:

Maximum Keys:



# Hashing



# Hashing

## Goals:

We want to define a **keyspace**, a (mathematical) description of the keys for a set of data.

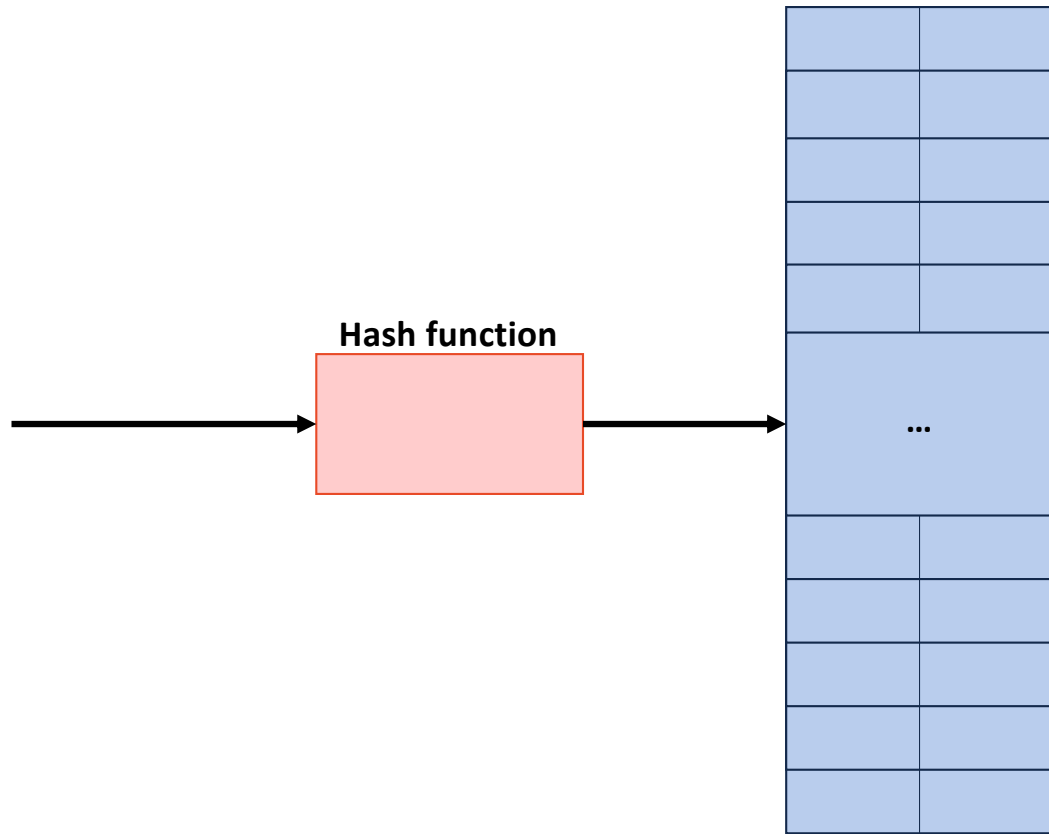
...use a function to map the **keyspace** into a small set of integers.



# Hashing

Locker Number	Name
103	
92	
330	
46	
124	

# Hashing



# A Hash Table based Dictionary

## Client Code:

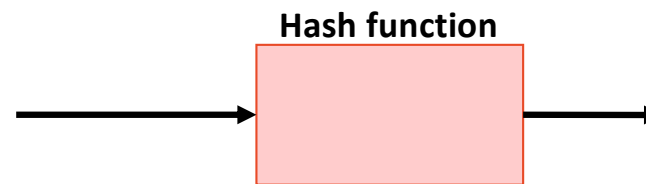
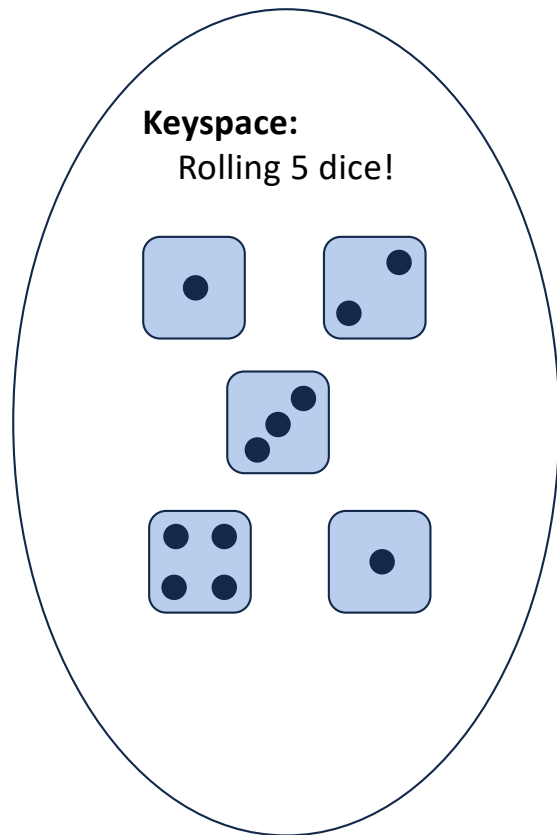
```
1 Dictionary<KeyType, ValueType> d;  
2 d[k] = v;
```

A **Hash Table** consists of three things:

- 1.
- 2.
- 3.



# A Perfect Hash Function



Key	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	