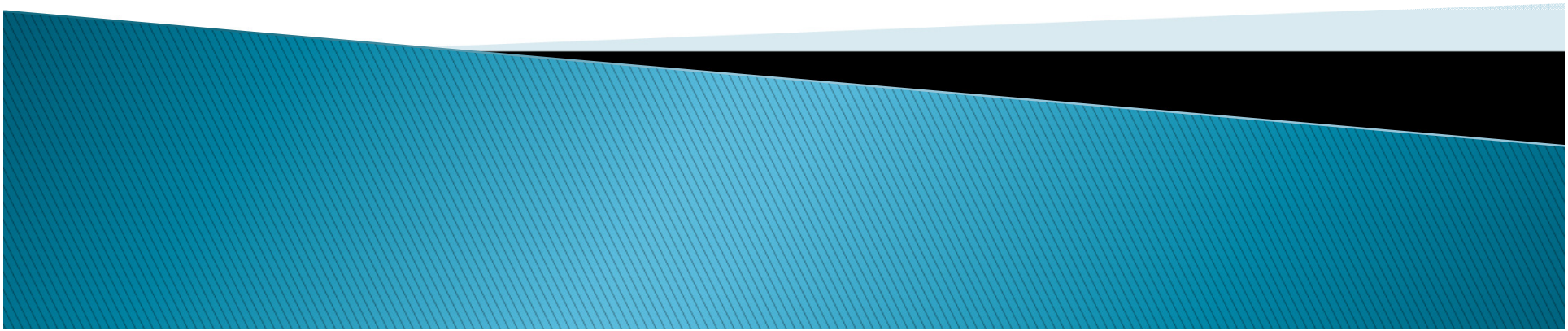
Concept of Hashing

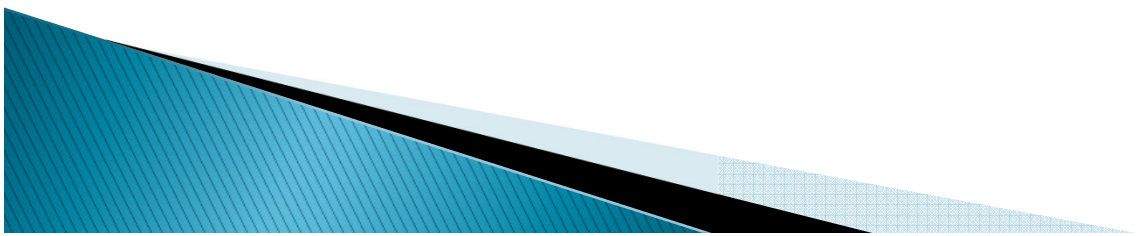
15-123

Systems Skills in C and Unix



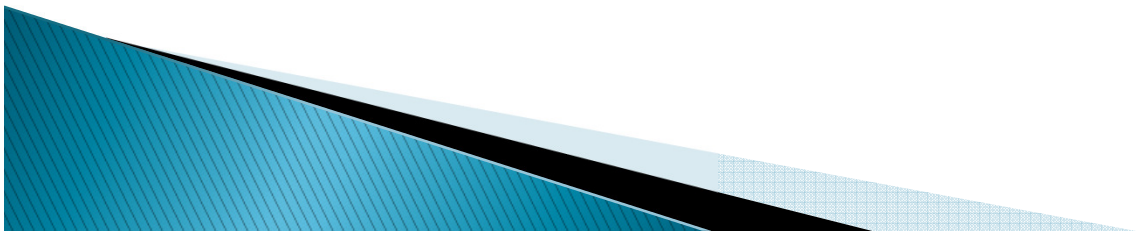
What is hashing

- Internet has grown to millions of users generating terabytes of content every day
- With such large data sets, how do we find anything?
- Two standard search techniques
  - Linear search – O(n)
  - Binary Search – O(log n)
- What if we need to find things even quicker?



Finding things in O(1)

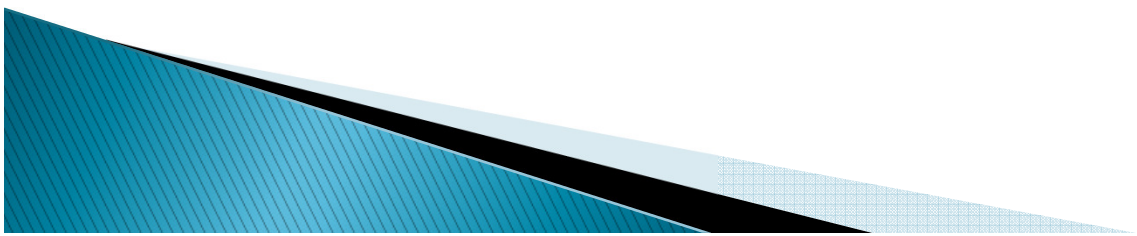
- Suppose our intent is to find an item in O(1)
  - That is, constant time or time does not depend on data size n
- In most cases, we only care about
  - Finding and retrieving things quickly
  - Updating and inserting things quickly
- We do not care about
  - Order statistics of the data



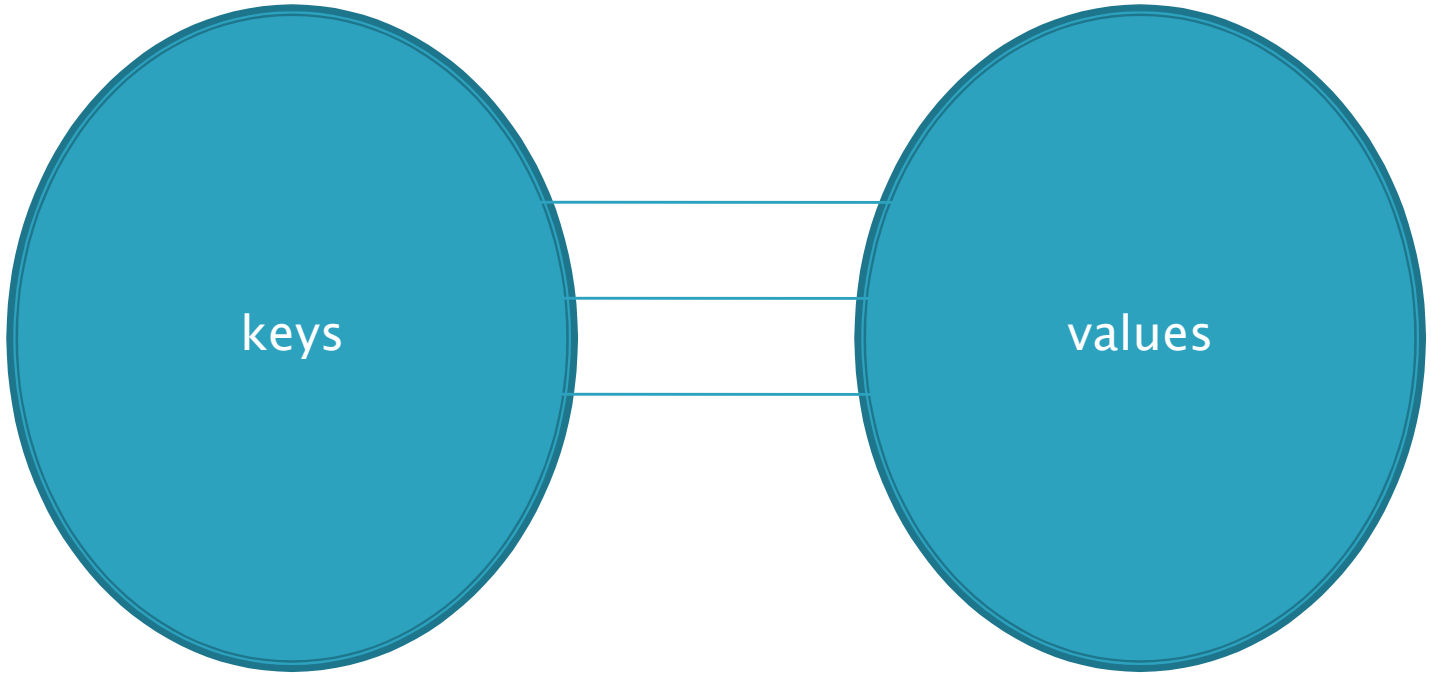
Finding things quickly

- Strategy – hashing

- Data Structure – hash table



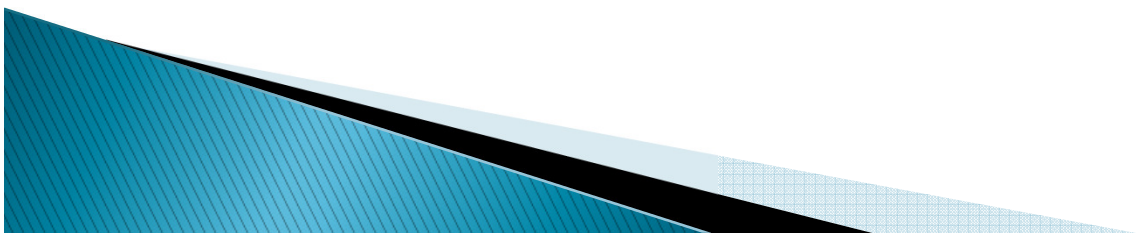
Maps



A relation between two sets defined by a simple function

Hash Function

- A hash function maps a key to a value
- Simplest Form
  - A[i] – a mapping of index (an integer) to a value
- The hash table idea is much more general
  - Keys don't have to be integers
  - H("guna") = "professor"
- If a hash function H can be defined, then information can be stored using (key,value) pairs

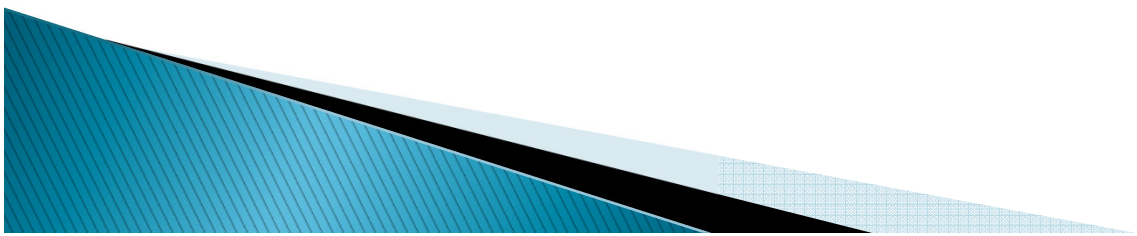


What makes a good hash function?

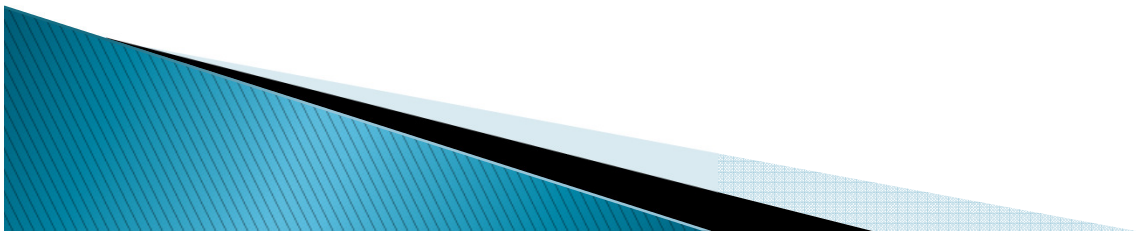
- A hash function must be
  - Easy to calculate
  - Must avoid “collisions”
- What do we mean by “easy to calculate”?
  - The cost of computing the hash value must be minimized
- What do we mean by “collisions”?
  - It is possible that two keys can map to the same value (unless you can come up with a perfect hash function)
  - Finding the perfect hash function is “hard”

Example

- Take a simple set of strings {“abc”, “bda”, “cad”}
- Define a hash function as follows
  - H(“abc”) = sum of the characters % 5
  - Where n = 5 is the table size
- Find H(“abc”), H(“bda”), H(“cad”)

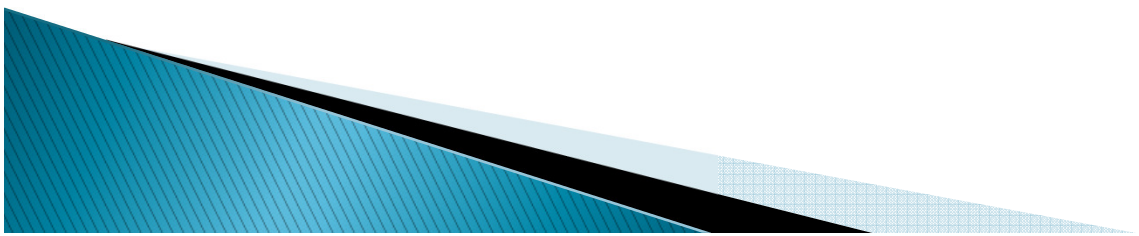


Storing the values



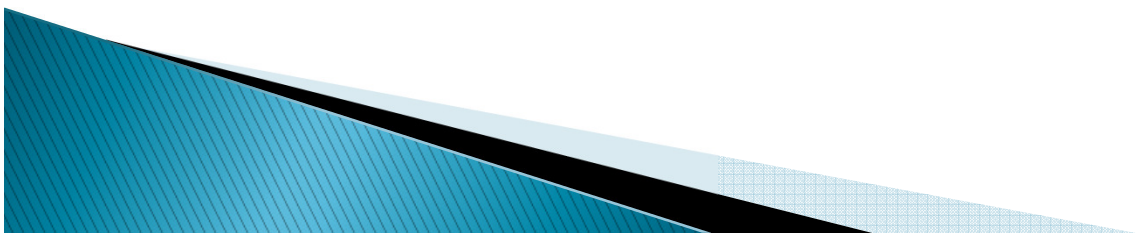
Questions

- What happens if “abc” and “bac” hash into the same location?
- How do we resolve it?
- Using a collision resolution strategy



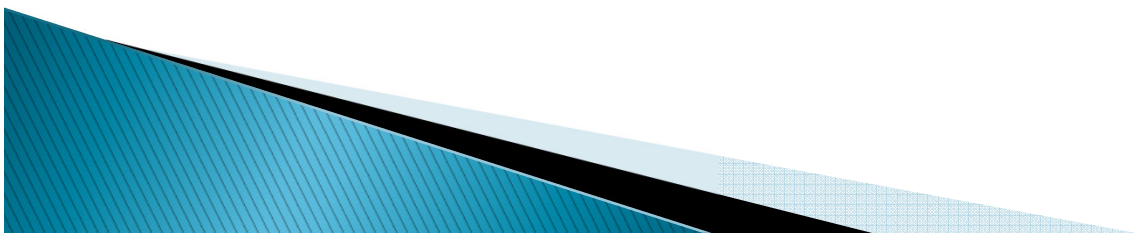
Using a better hash function

- H(s) = Σ sum of characters has too many collisions
- Define H(s) as a polynomial representation of characters of s



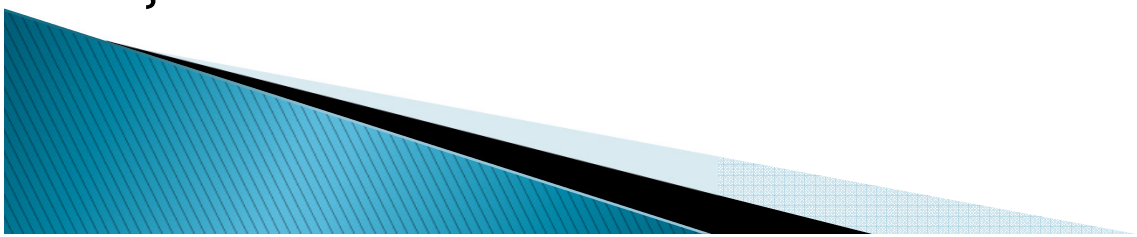
Making things more efficient

- How can we calculate H(s) more efficiently?



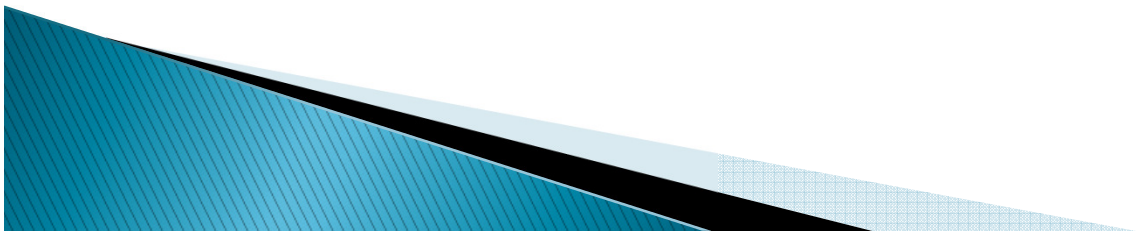
Good hash function

```
int hash_string(char* s, int n, int m) {
     int a = 1664525; int b = 1013904223;
    /* inlined random number generator */
    int r = 0x1337beef; /* initial seed */
    int h = 0; /* empty string maps to 0 */
    for (int i = 0; i < n; i++) {
            h = r*h + (int)s[i];
            r = r*a + b; linear congruential random no */
     }
     h = h % m; /* reduce to range */
     h += m; /* make positive, if necessary */
     return h;
}
```



Questions

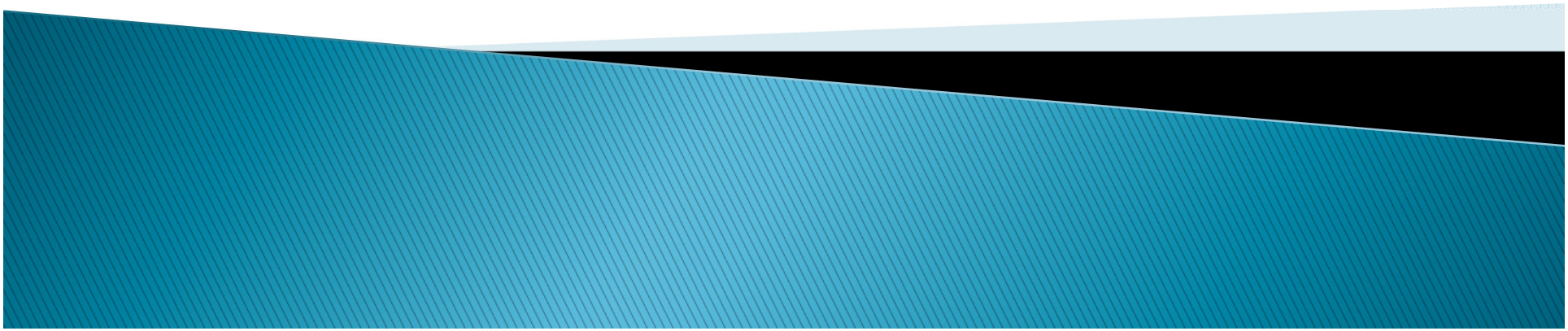
- Suppose we would like to hash 10000 keys, (each up to a 5 character string) into a hash table of size 12000. We use the function
  - H(string) = Σ sum of the characters of the string
- What would be the key distribution?



Collision Resolution

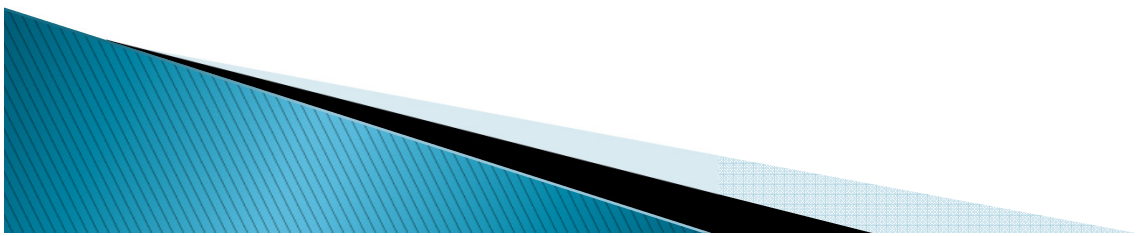
15-123

Systems Skills in C and Unix

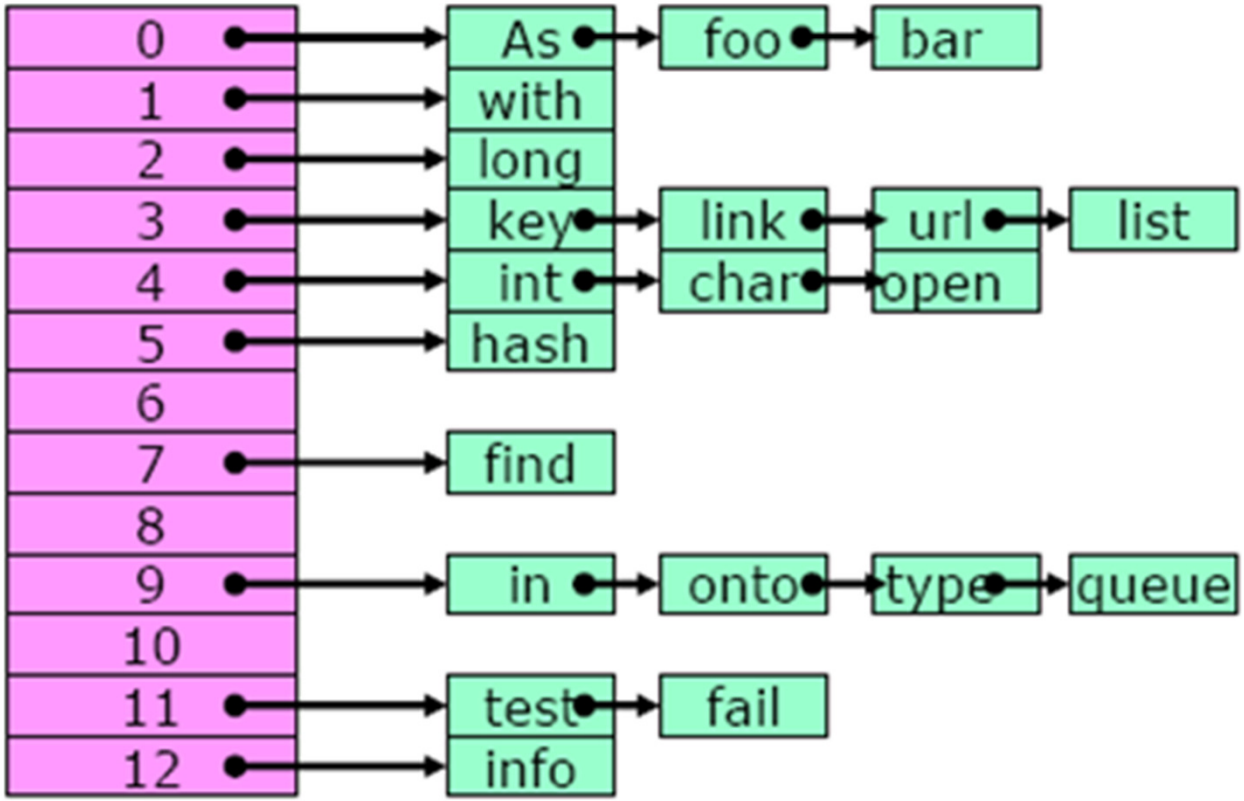


What is a collision

- A collision occurs when two keys map to the same location
- Why do collisions occur?
  - Mainly due to bad hash functions
  - Eg: imagine hashing 1000 keys, where each key is on average 6 characters long, using a simple function like H(s) = Σ characters

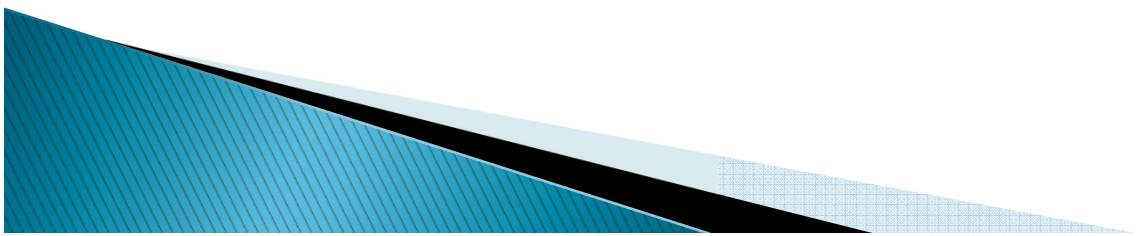


Separate Chaining



Separate Chaining

- Pros
  - No probing necessary
    - Each node has a place in the same hashcode
  - List gets never full
    - Performance can go down though
- Cons
  - Complicated implementation of array of linked lists
  - Still lots of collisions can create a “bad” hash table



Coding Examples