



CSE 373: Hash Tables

Chapter 5



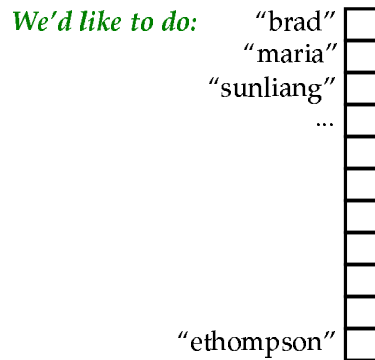
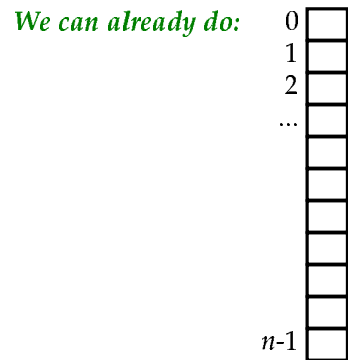
Motivation



Goal: The ability to store and retrieve information in $O(1)$ time

Current Solutions:

Hash Table Insight

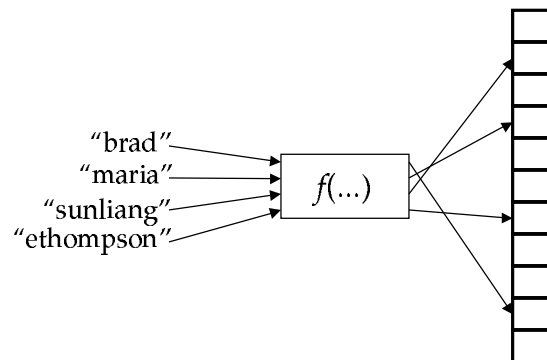


UW, Spring 1999

CSE 373 – Data Structures and Algorithms

Brad Chamberlain

Hash Table Approach

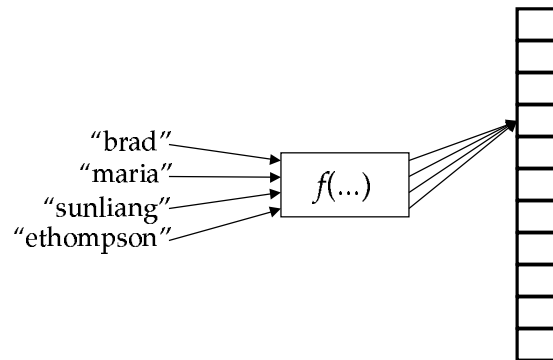


UW, Spring 1999

CSE 373 – Data Structures and Algorithms

Brad Chamberlain

Hashing Conflicts



UW, Spring 1999

CSE 373 – Data Structures and Algorithms

Brad Chamberlain

Hashing Integers



What's a simple scheme for hashing integers?

```
int HashInteger(int key,int tablesize) {  
  
}
```

- advantages?
- disadvantages?

UW, Spring 1999

CSE 373 – Data Structures and Algorithms

Brad Chamberlain

General Strategies



- Selecting a good hash function often depends on the set of possible keys
- Using a hash table whose size is a prime number tends to help reduce conflicts

Hashing Strings



General approach:

- convert string to integer
- “mod” integer by table size

Naive approach:

```
int HashString(char *key,int tablesize) {
    int total=0;
    while (*key) {
        total += *key;
    }
    return total%tablesize;
}
```

Problem with Naive Approach



```
HashString("bat", n)
= HashString("tab", n);
= HashString("rad", n);
```

Total Number of Possibilities $\approx 127 \times 12 = 1524$

Useful Number of Possibilities $\approx 27 \times 12 = 324$

Probably not good if hash table size or number of keys is greater than this...

Improved Approach



Read string as base 27 number:

1 27 729

$$\text{b a t} = 2 \times 1 + 1 \times 27 + 20 \times 729 = 14,609$$

$$\text{t a b} = 20 \times 1 + 1 \times 27 + 2 \times 729 = 1,505$$

$$\text{r a d} = 18 \times 1 + 1 \times 27 + 4 \times 729 = 2,961$$

Advantages?

Disadvantages?

Other Ideas



Hash using only a subset of the characters...

- first three?
- last three?
- middle three?
- first, middle, last?
- etc.

Hash Function Design Goals



- Hash to all slots in your table
- Avoid collisions
- Hash as evenly as possible
- Hash as quickly as possible

(Again, note that much of this may depend on the set of possible keys...)

Harsh Hash Reality



- No matter how good your hash function is, collisions will probably occur
- Thus, we also need a collision resolution strategy...
 - separate chaining
 - resizable hash table
 - open addressing

Hash Table Operations



- Main Operations:
 - `void Insert(HashTable, HashType);`
 - `Position Find(HashTable, HashType);`
 - `void Delete(HashTable, Position);`
- Normal Creation/Deletion Operations
- No iteration, `FindMin()`/`Max()`, etc.
(why?)

Operator Analysis



Hash Table *List* *BST*

problem size

space

Insert()

Find()

Delete()

other

UW, Spring 1999

CSE 373 – Data Structures and Algorithms

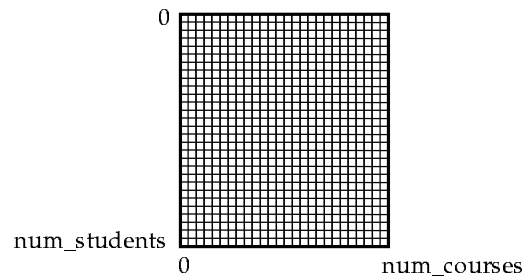
Brad Chamberlain

Flashback to Day 1



```
const int num_courses = 7000;  
const int num_students = 33000;
```

```
typedef int registry[num_students][num_classes];
```



UW, Spring 1999

CSE 373 – Data Structures and Algorithms

Brad Chamberlain