10.1 - Week 10 - Applied - Theory

Objectives of this Applied Session

Objectives of this Applied Session

- To be able to implement and analyse hash tables
- To understand what makes a good or bad hash function

Following Along

Question *Submitted Oct 4th 2022 at 9:07:55 am*

Consider the following hash function:

which is used in a Hash Table which has linear probing implemented (with a standard probe step size of 1). Also consider the following names:

- Calculate and state the hash value of the above names. Explain and show how they are inserted into the table (using the same order of insertion). Using a piece of paper or excel spreadsheet helps here.
- State what happens when you search for the names Jim, Jon and Joe.

=69%19=12 (E at 12)

A = 65%19=8

T = 84%19 = 8

12 / 8 / 9 (Because 8 already have smtg, it'll go to 9) / 17 / 18 / 11 / 10 (8 is occupied, 9 is occupied) / 0 (17 is occupied, 18 is occupied, 19%19=0 so will go to 0) / 1 (17 is occupied, 18 is occupied, 0 occupied)

One by one find but there's no Joe

Peculiar Probing

In the code tab to the right, you've been given a slightly faulty version of a hash table with linear probing.

1. Fix the bugs, and make sure your hash table passes the tests starting with .
2. Add a new method to the class, so that your hash table can linear probe with a jump not equal to 1, with argument given as an argument. (For example, linear probing might check 5, 8, 11, 14...). Make sure it passes tests starting with .
   Advanced Questions:
3. Add a new method to the class, so that your hash table linear probes, like the class, but the jump size is decided by the first character in the string (For example, if the first character of the key is , then use a jump size of 1, if the first character of the key is , then use a jump size of 4)
4. Is still a functional hash table? If not, what issues can arise? If so, what are the benefits/downsides of using this over standard linear probing?

Good Hash Bad Hash

Question 1 *Submitted Oct 4th 2022 at 9:14:16 am*

I'm creating a hash table which will be used by a library, so that books can be accessed by their ISBN code

For example, you might want to insert:

- The book "A Brief History of Time", with key
- The book "The Art of Computer Programming", with key
- The entire "🅱ᴇᴇ Movie" script, with key

For this hashtable, I've decided to use linear probing, with hash function:

Is this hash function well suited for the problem at hand? What issues might I encounter?

2 - key value

5 - remainder

PROBLEM = 2. only take first value so first 2 books will collide

Question 2 *Submitted Oct 4th 2022 at 9:18:31 am*

I'm creating a lookup structure to store all my poems to show on my blog, here's a quick sample:

> Week 10 - Noskcaj Renreog
>
> Roses are Red,
> Violets are Blue,
> I can't wait,
> To learn hashtables with you!

I write a poem almost every week (and only ever 1 per week), so the urls for each poem are accessed by the week number:

https://blog.noskcaj.gov/poem-week-10

So I've decided to use the following hash function:

Is this hash function well suited for the problem at hand? Is this the best structure to represent this data?

It will work with big table size with no collision

Split is O(N) time complexity where N is length of key

PROBLEM = need to keep calling hash to do anything

CAN DO BETTER = normal array (Complexity probably O(1))

Deleting from Hashtables is scary

Suppose we have the hashtable from the "Following Along" quiz question, and have inserted everyone's names.

1. Discuss, in , what issues can arise when deleting a key from a hashtable (and nothing is done to remedy this). Give an example removal, using the hashtable above, to demonstrate this.
2. Copy your hashtable implementation from the previous coding task to , and add a new method which deletes the element in the hashtable, taking appropriate measures to ensure that the hash table still functions, and raises a if the is not present in the table.

More Mysteries

Question 1 *Submitted Oct 4th 2022 at 9:55:24 am*

Consider a hash table implemented with Linear Probing and maximum capacity **TABLESIZE=29**, which is ~3/4ths full. Assume you decide to resize and rehash, in order to reduce the table size.

Please answer the following questions:

- What size you would choose to reduce the load factor below 0.5?
- Why would we want to reduce the load factor below 0.5?
- What reasonable potential factors might affect your choice?

- **What size you would choose to reduce the load factor below 0.5?**
- **Why would we want to reduce the load factor below 0.5?**
- **What reasonable potential factors might affect your choice?**

22 items; load factor 0.5 means half full, half empty

roughly 45 but should be prime num so 47 is the next prime number and hence TABLE-SIZE = 47

Question 2 *Submitted Oct 4th 2022 at 9:49:13 am*

Suppose we are using the following mystery hash function to hash a list of unbounded numbers:

Where the value of is not known before picking the tablesize but it is known that , and is an integer.

We'll call a hashtable achievably perfect, if for any index in the hashtable, we can provide an input , where (There are no unhashable positions in the table).
What values of guarantee an achievably perfect hash table, no matter the value of a?

- [ ] 4
- [x] 7
- [ ] 9

- [ ] 16
- [x] 23