EECS 233 SI Session 13
Leader: Bertram Su
October 31, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

1. Analyze hashing insertion code
2. Analyze deletion in hash tables

Foundations:

1. Examine the follow hash table from the last SI session. What value would be stored in the blank spot?

0	1	2	3	4	5	6
Apple F	Bear F	~~Bat~~ ~~T~~ Barn F	Boat ~~F~~ T	null	Baron F	Goof F

2. Assume there is a hash table that uses linear probing with h1 = first letter's order in the alphabet (a=0, b=1, c=2, etc). Given the table above, where would Barn be stored?

3. Redraw the entry square after removing boat. Should other entries be changed?

3
Boat
T

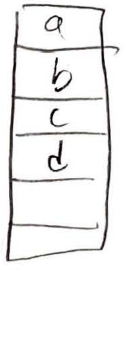
4. Clearly Baron is in this hash table. If I search for Baron, can I find it? Why or why not?

No, 4 is null

5. After using my hash table for a while, my load factor is high. What should I do? What does a high load factor possibly imply performance?

Rehash w/ larger table

Possible to have poor performance if collis ↑



$\frac{\text{items}}{\text{space}}$

Exercises:

6. Finish the following Entry class for an entry in a hash table that holds integers

```
public class Entry{

        int data;
        boolean remove;

}
```

7. Finish the following probe method to find an open index. This should look very familiar

```
private int probe(int key){
        int i = h1(key);
        int j = h2(key);
        int iterations = 0;

        //I want to keep going until I find an open space
        while(table[i] != null && table[i].removed == false){
                i = (i+j) % table.size; //this is how it was chosen to increase i in lecture
                iterations++;
                if(iterations ≥ table.size)
                        return -1;
        }
        return i;
```

From Here
Key = data
data = key

8. Finish the following find ~~Key~~ data method.

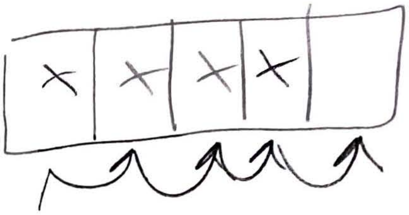
```
find~~Key~~data(int ~~key~~ data){ data
        int i = h1(~~key~~ data);
        int j = h2(~~key~~ data);
        int iterations = 0;

        while(table[i] != null)
                if(table[i].removed ==false && table[i].data==data)
                        return i;
                i = (i+j)%tableSize;
                iterations++;
                if(iterations ≥ tableSize)
                        return -1;
        }
}
```

9. Finish the following code for deletion

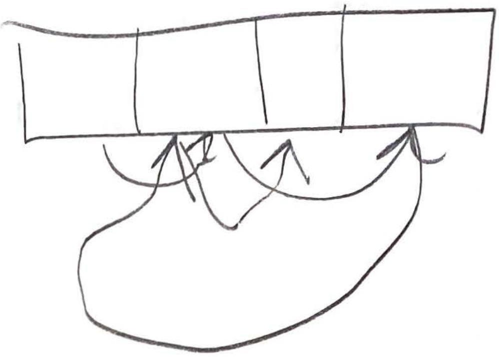
```
public void delete(int key)
        int i = find Key (key);
        if(i == -1 )
                return;
        else table[i].removed = true ;

}
```



10. finish the following code for insertion.

```
public void insert(int k){
        int i = probe (key);
        if(i == -1 )
                //throw an error
        else{
                if(table[i] != null ){
                        table[i].data = k;
                        table[i].removed = false;
                else
                        table[i] = new Entry(k) ;
}
```



Summary

11. The Red Cross is holding a silent auction online. The item is given to the highest bidder. What type of data structure should we use to store the data and what would our key be?

Max heap

12. Finish the chart for the Big O of an array and a hash. Why would we ever use a hash table?

Operation	**Array**	**Hash**
Search	n	n
Insertion	n	n
Deletion	n	n

you're probably not dealing w/ worst

Upcoming Events and Suggestions for Further Study:

Events:

- Next SI session is Sunday from 1:00 to 2:30 at Sears 336

Further Study:

- bigocheatsheet.com
    - A great graph that visualizes the big o complexity chart. It also has the big O time of data structures and algorithms that we will cover in the future.
- https://www.geeksforgeeks.org has nice hashing examples and code