Lecture 11

Hashing

Runtime analysis

Linear Search

Add(**O**(1))

Remove(**O**(n))

Find(**O**(n))

Runtime analysis

Binary Search

Add(**O**(n))

Remove(**O**(log n))

Find(**O**(log n))

Binary Search

```
bool binarysearch(int key,int array[], int low, int high){

  if(low > high) return false;

  int mid = (low + high)/2;

  if(key == array[mid]) return true;

  if(key > array[mid])return binarysearch(key,array,mid+1,high);

  else return binarysearch(key,array,low,mid-1);
}...
```

Binary Search

```
...
int main(){
  //sorted array
  int array[10] = {1,3,4,6,7,9,12,34,89,100};

  cout<<binarysearch(6,array,0,9)<<endl;
  return 0;

}
```

Runtime analysis

Binary Search Tree

Add(**O**(log n))

Remove(**O**(log n))

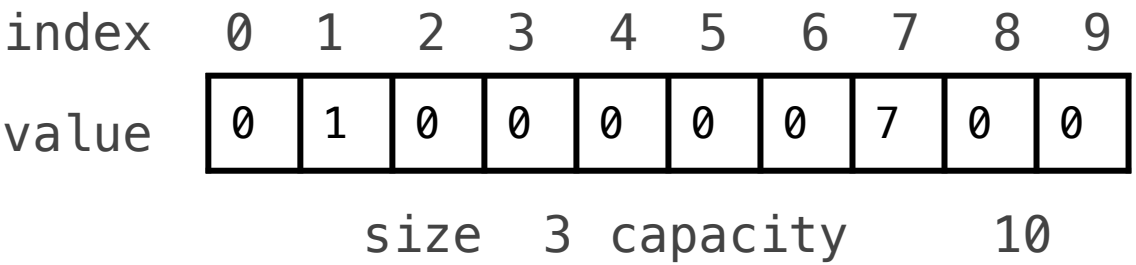
Find(**O**(log n))

Runtime analysis

Hash table

- Add(O(1))
- Remove(O(1))
- Find(O(1))

Hash Table (Linear Probing)




```
int hashCode(int num){
  return num%arraySize;
}
```

Hash Table (Separate Chaining)