CSE 142
Programming I

Linear and Binary Search

Searching

- Problem: given an array, find a particular value
- This is a common problem
- What might make this easier to solve?

Linear Search

- Our first idea will be to simply look at each element in turn until we find what we're looking for.
- This is linear search

Linear Search

```
int search(int toFind, int a[], int start,
  int end){

  int i;

  for (i=start; i<=end;i++){
     if (a[i] == toFind) return i;
  }

  return -1;  /* not found */
}
```


Linear Search

- Say we have the array

```
int foo = {1, 7, 2, 10, -3, 0, 1, -2}
```

- What do we get when we call
  search(1, foo, 0, 7);
  search(7, foo, 2, 6);

How Long Does This Take?

- Assume that each element occurs only once
- On average we have to look at half of the elements before we find the right one
- If we double the number of elements, it doubles the time
  - ➤ "linear time"
- What if we had 1,000,000 elements?

Can We Do Better?

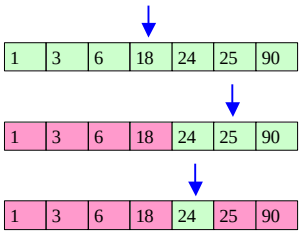
- Imagine looking for someone in the phone book
- Do you start at the beginning and look at every entry as you go?
- What is special about the phonebook?

Binary Search

- If the data is sorted we can use binary search
- Idea: See check the middle element to decide which half to look at. Repeat the process on each half.

Binary Search Strategy:

Look for 24



Let's Write the Code:

```
int bsearch(int toFind, int a[],
  int begin, int end);
```


Binary Search

```
int bsearch(int toFind, int a[], int left,
            int right){

  int mid;
  while (left <= right){
      mid = (left + right) / 2;
      if (a[mid] == toFind) return mid;
      if (toFind < a[mid]) right = mid - 1;
      else left = mid + 1;
  }
  return -1;
}
```


Is This Actually Better?

- If we have 1,000,000 elements, we only need to loop about 20 times!
- For the time to double, we need to square the number of elements!
  - ➤We call this "logarithmic growth"
  - ➤How many loops to search 1,000,000,000,000 elements?