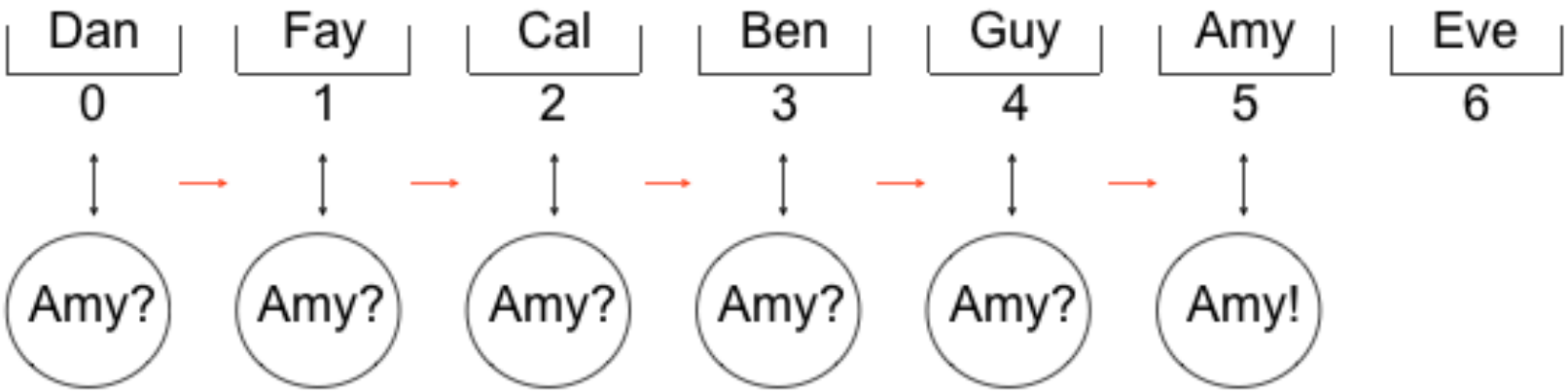
Searching

- Sequential Search
- Binary Search

Sequential Search

- Scans the list comparing the target value to each element.



Sequential Search

- List does not need to be sorted.
- Worst case: n comparisons to find target or **O(n)** (where n = total number of elements).
- Average case: about n / 2
- Best case: ????
- n comparisons are needed to establish that the target value is not in the array.

Binary Search

- The elements of the list must be arranged in ascending (or descending) order.
- The target value is always compared with the middle element of the remaining search range.
- We must have random access to the elements of the list (think: array or **ArrayList**).

Binary Search

Recursive method:

```
public Integer binarySearch (Integer [ ] arr, Integer value,
                             int left, int right)
{
   if (right < left)
      return -1;         // Not found

   int mid = (left + right) / 2;
   int compare = value.compareTo(arr[mid]); // costly!!!

   if (compare == 0)    // easy comparison
      return mid;

   else if (compare < 0)
      return binarySearch (arr, value, left, mid - 1);

   else      //  if ( compare > 0)
      return binarySearch (arr, value, mid + 1, right);
}
```

Binary Search

Iterative method:

```
public Integer binarySearch (Integer [ ] arr, Integer value,
                             int left, int right)
{
   while (left <= right)
   {
       int mid = (left + right) / 2;
       int compare = value.compareTo(arr[mid]);

       if ( compare == 0 )      // value = arr[mid]
          return mid;

       else if ( compare < 0 )  // value < arr[mid]
          right = mid - 1;

       else                     //  value > arr[mid]
          left = mid + 1;
    }
    return -1;  // Not found
}
```

Binary Search

- A "divide and conquer" algorithm.
- Works very fast: only 20 comparisons are needed for an array of 1,000,000 elements; (30 comparisons can handle 1,000,000,000 elements; etc.).
- We say that this is an **O(log n)** algorithm.
- Max number of comparisons = $\lfloor \log_2 n \rfloor + 1$

Binary Search

Practice: Search for “Cal”

Amy **Ben** **Cal** **Dan** **Eve** **Fay** **Guy**