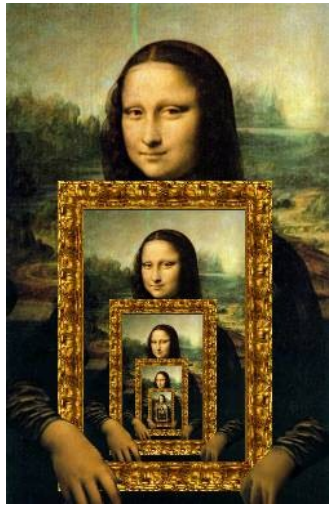
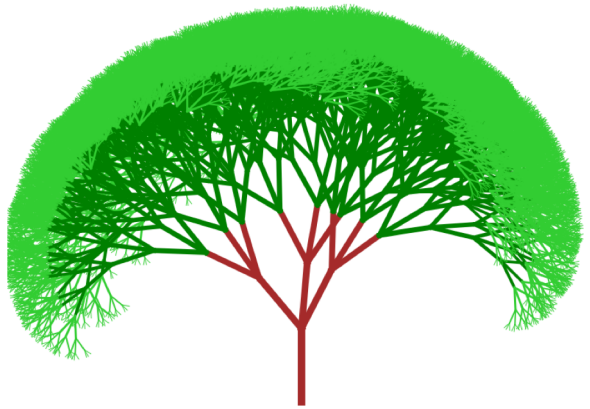
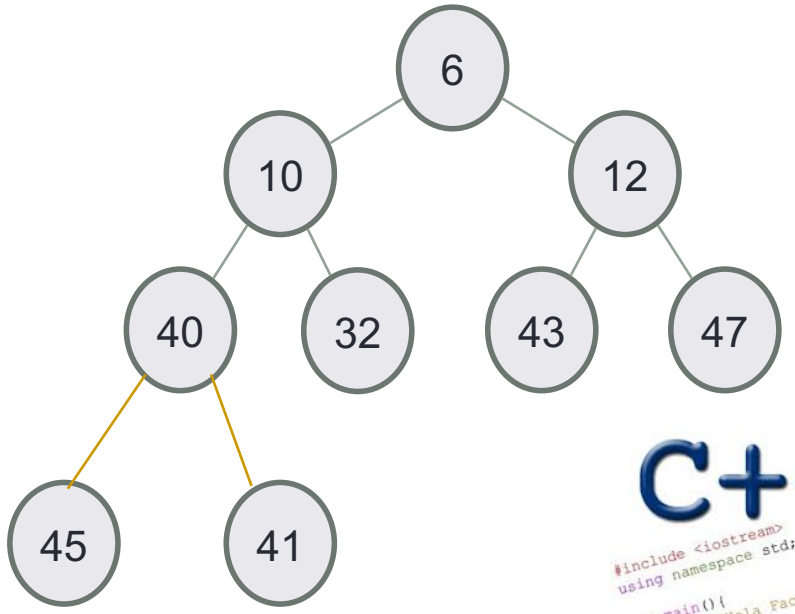
MORE ON RECURSION



Problem Solving with Computers-I





COMPUTER SCIENCE
UNDERGRADUATE AFFAIRS
COMMITTEE

PRESENTS

Speed Advising

Date: Friday, December 6, 2019
Location: 1132 Harold Frank Hall
Time: 10:00 AM - 1:00 PM

Refreshments will be provided

Final Exam: Monday 12/09, noon-3:00p, Embarcadero Hall

Final Exam Review Session:

Day: Friday (12/06)
Time: 5:00p - 7:00p
Location: Phelps 3536

Thinking recursively !

```
int fac(int N){

    if (N <= 1)
        return 1;

    else{
        int rest = fac(N-1);
        return rest * N;
}
```

Base case

Recursive case

Human: Base case and 1 step

Computer: Everything else

Thinking recursively !

```
int fac(int N){

    if (N <= 1)
        return 1;

    else
        return fac(N-1)* N;
}
```

Base case

Recursive case (shorter version)

Human: Base case and 1 step

Computer: Everything else

Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```

```
cout<<fac(1);
```

Result: 1

The base case !

Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```

fac(5)

Behind the curtain…

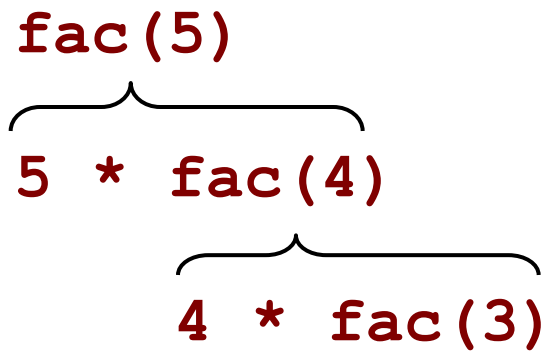
```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```

fac(5)

5 * fac(4)

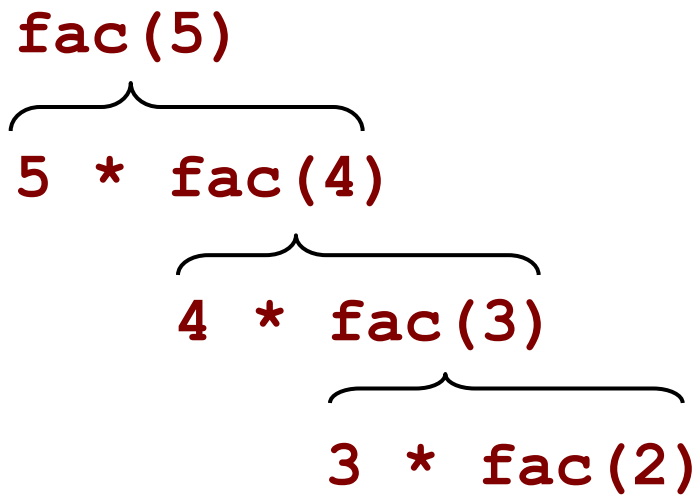
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



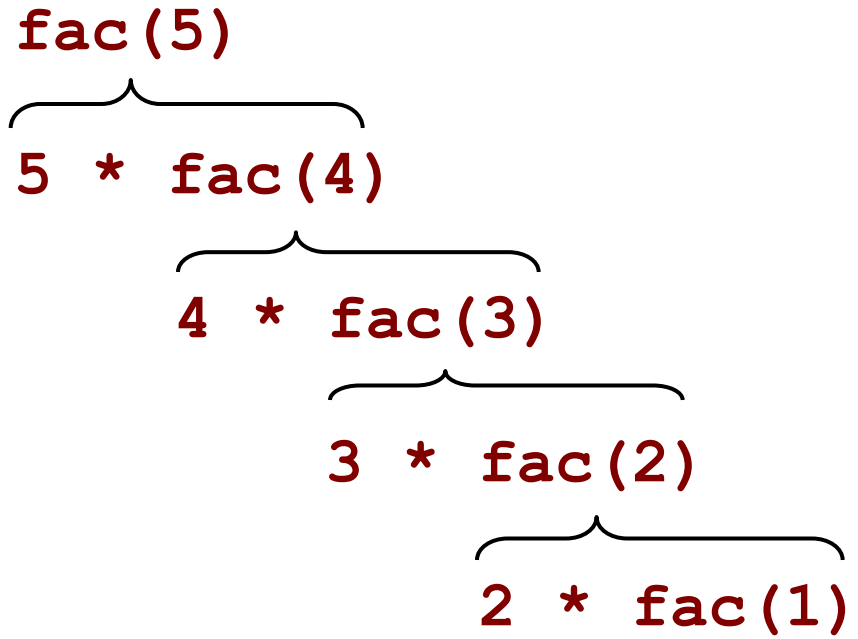
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



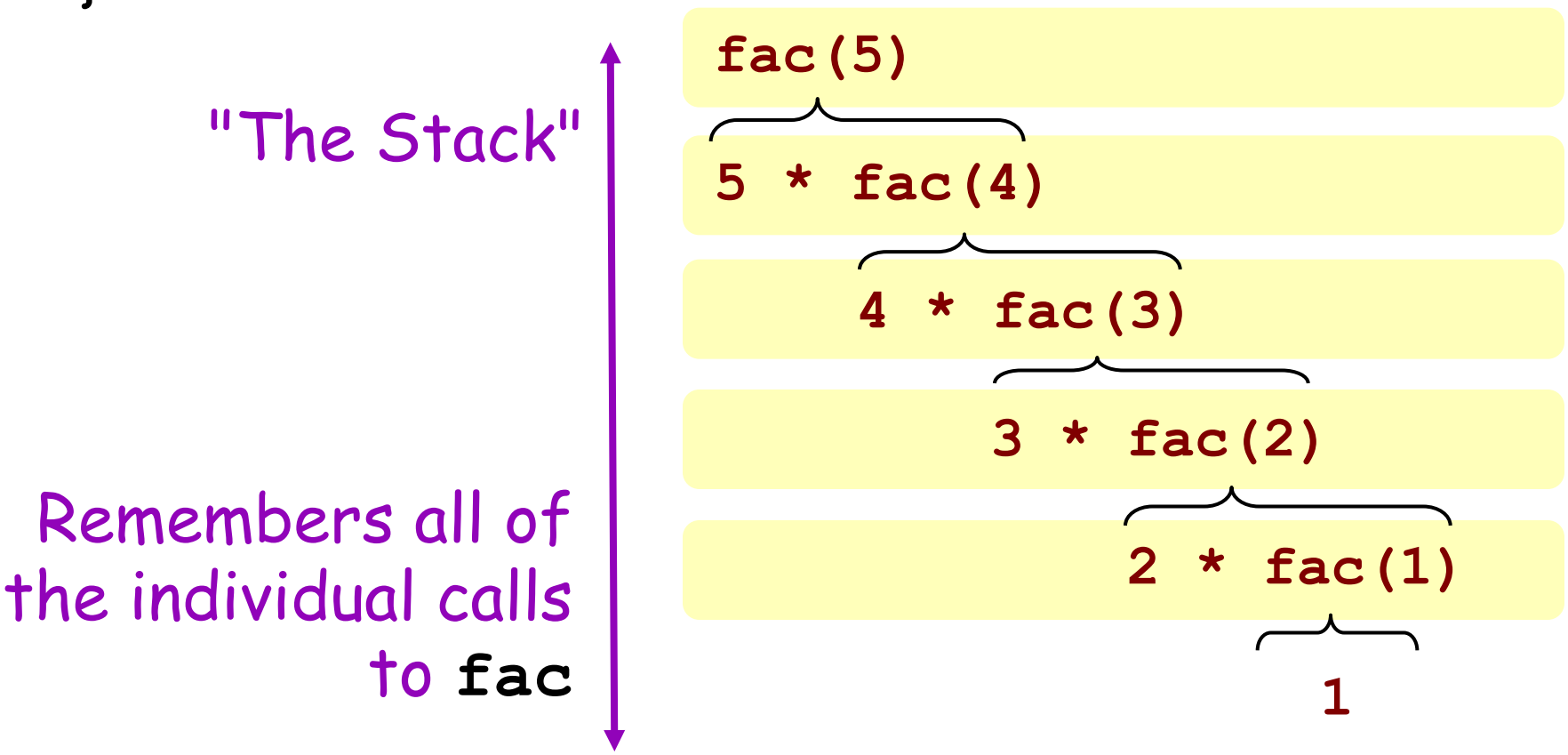
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



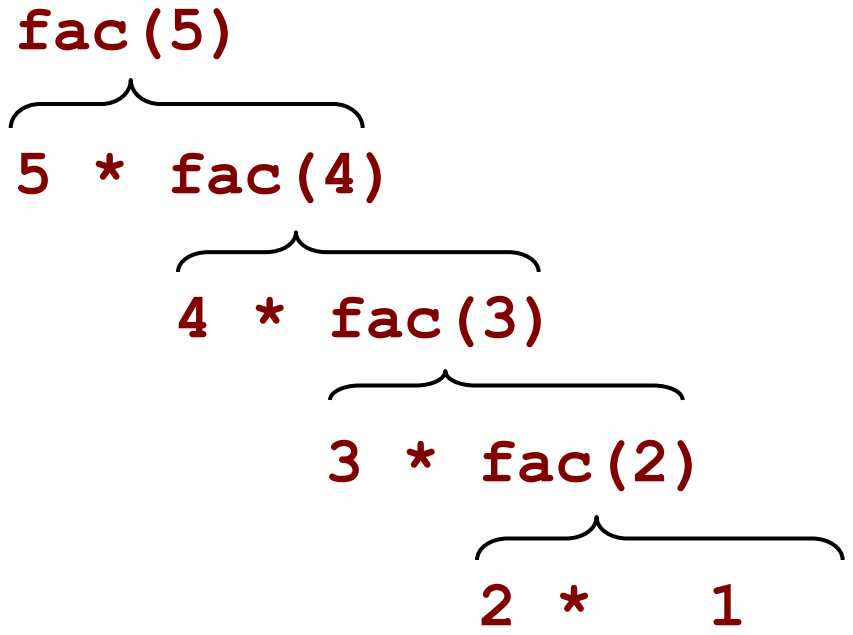
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



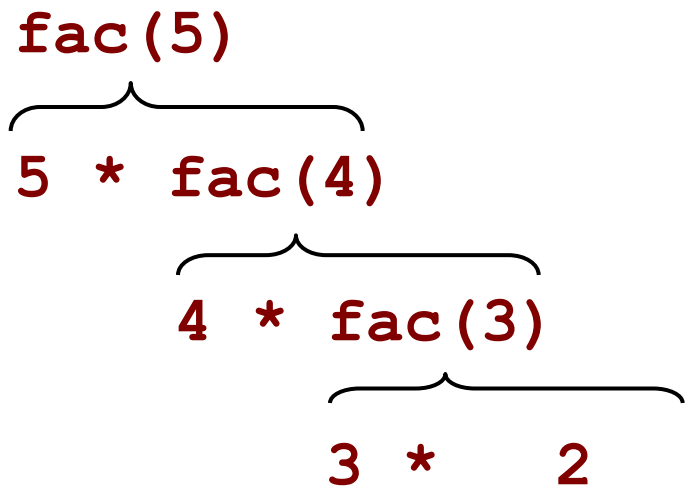
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



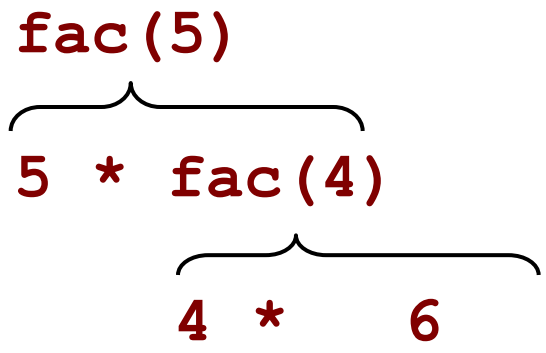
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



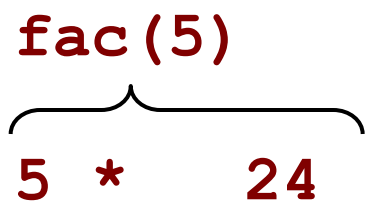
Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```



Behind the curtain…

```
int fac(int N){

    if (N <= 1)
        return 1;
    else
        return N * fac(N-1);
}
```

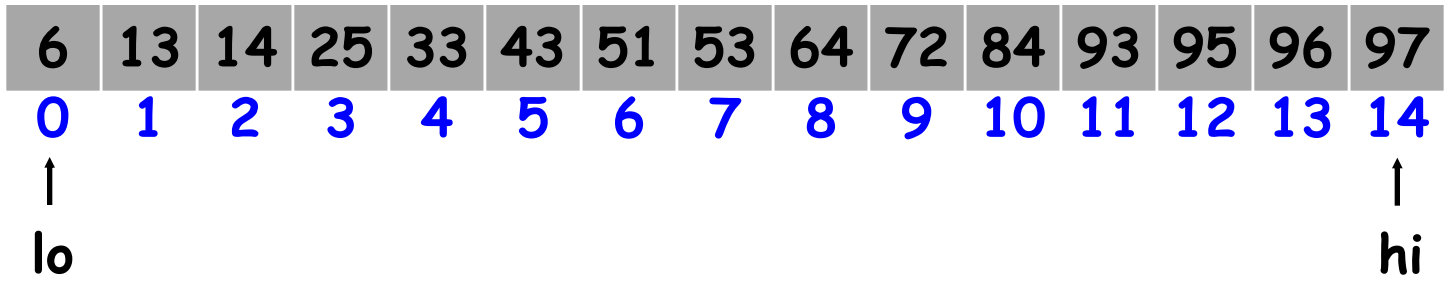
fac(5)

Result: 120

Binary Search: Efficient search in a sorted array

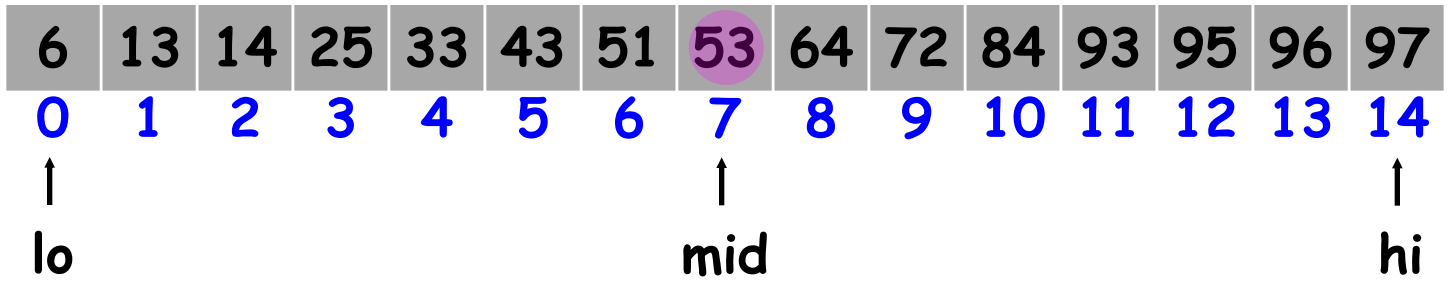
- Binary search. Given `value` and sorted array `v[]`, find index `i` such that `v[i] == value`, or return -1 indicating that no such index exists.
- Invariant. Algorithm maintains `v[lo]` ≤ `value` ≤ `v[hi]`.

- Ex. Binary search for 33.



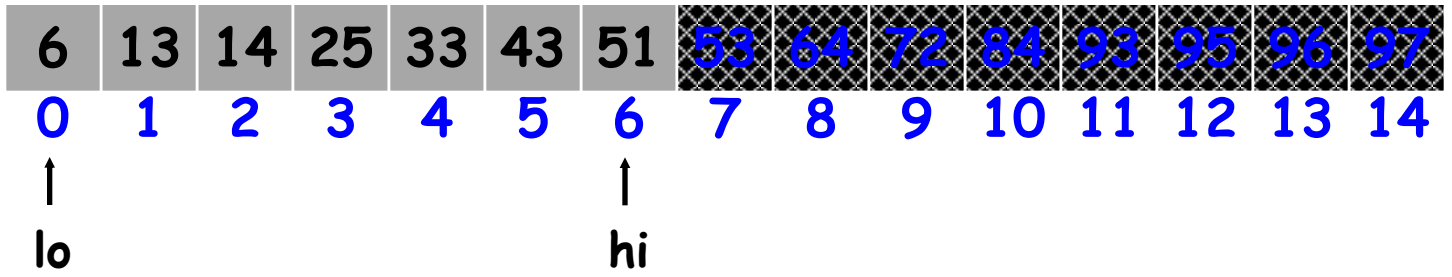
Binary Search

- Ex. Binary search for 33.



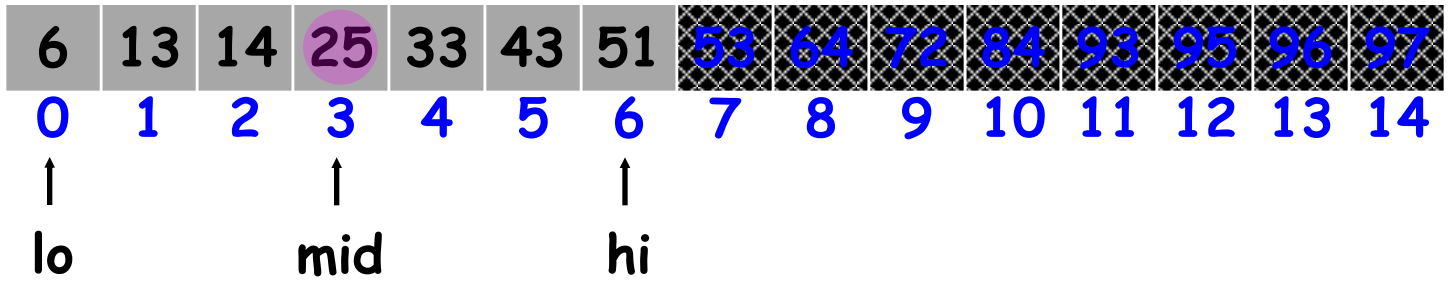
Binary Search

- Ex. Binary search for 33.



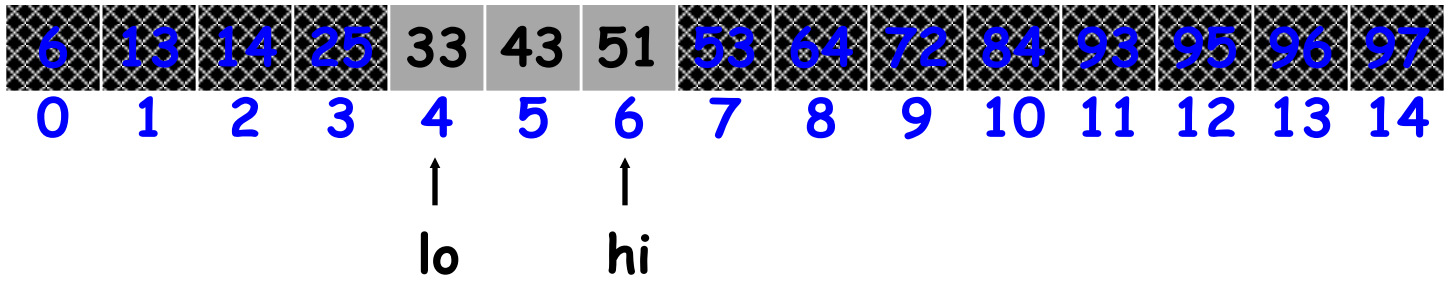
Binary Search

- Ex. Binary search for 33.



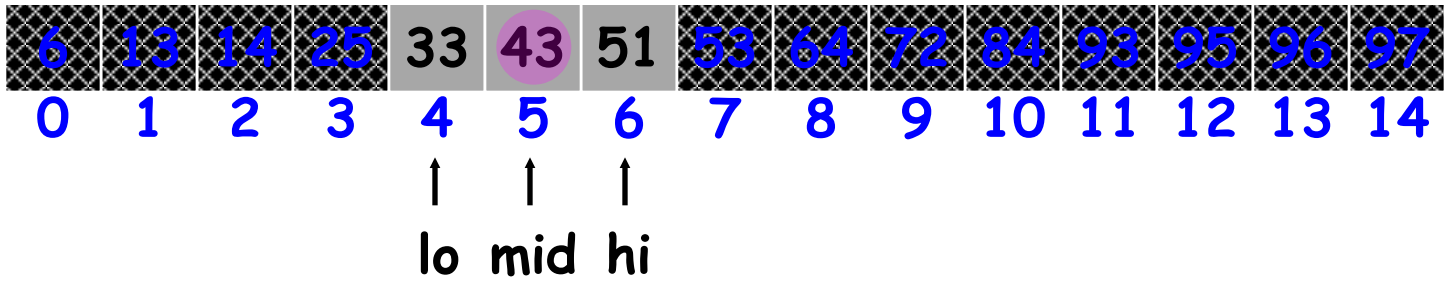
Binary Search

- Ex. Binary search for 33.



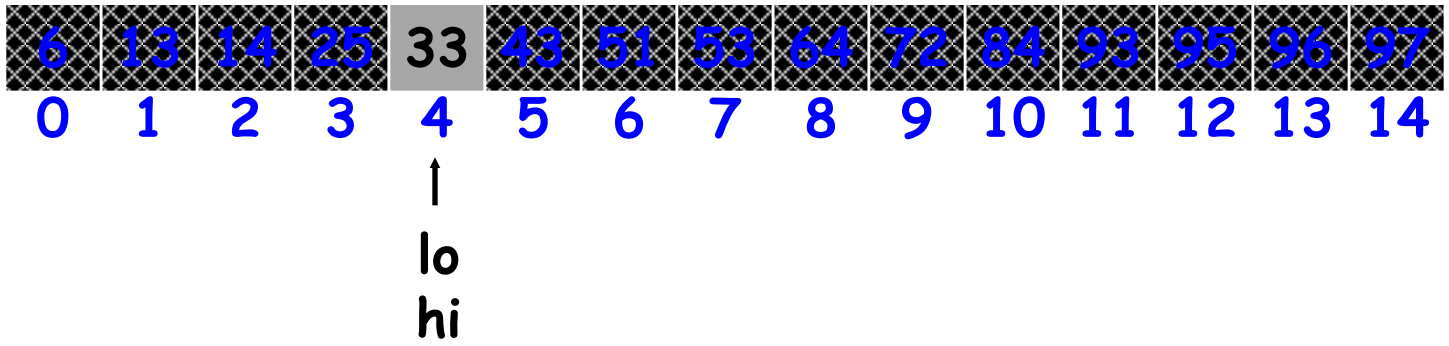
Binary Search

- Ex. Binary search for 33.



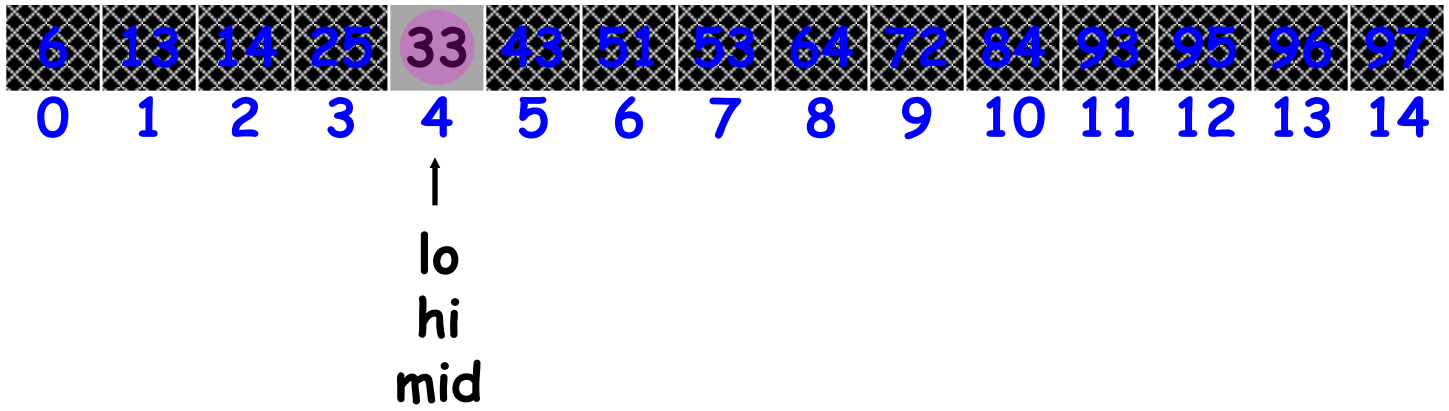
Binary Search

- Ex. Binary search for 33.



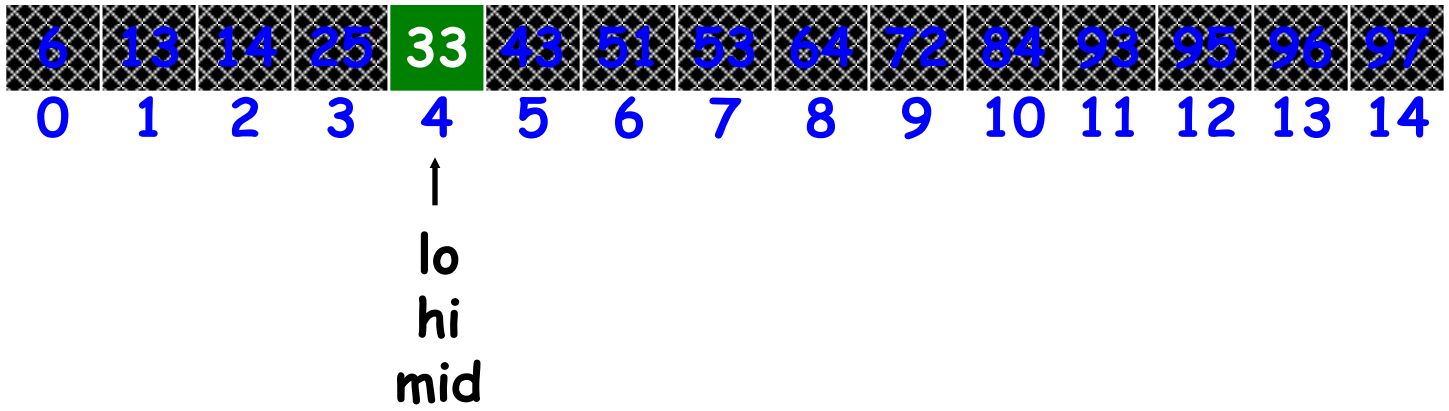
Binary Search

- Ex. Binary search for 33.



Binary Search

- Ex. Binary search for 33.



Write the recursive implementation of Binary search

```
int binarySearch(int v[], int value, int lo, int hi);
```

Which of the following is a valid base case?

```
int binarySearch(int v[], int value, int lo, int hi){
```

A:

```
if(hi<=lo){
    return -1;
}
```

B:

```
int mid = (lo + hi)/2;
if(v[mid] == value){
  return mid;
}
```

C: Both A and B

Fill in the blanks

```
int binarySearch(int v[], int value, int lo, int hi){
     if(hi < lo)
          return -1;
     int mid = (lo + hi)/2;
     if(v[mid] == value)
          return mid;
     if(v[mid] < value){
          binarySearch(v, value, _____, hi);
     }

}
```

A: lo
B: mid - 1
C: mid
D: mid + 1
E: hi

Searching a linked list

Given a linked list, implement a recursive search function

- Return true if a given value is present in the linked list
- Otherwise return false

Recursive function to free nodes in a linked list

Given a linked list, implement a recursive function to delete all the nodes in the linked list

Is this a correct implementation?

A: Yes

B: No

```
int binarySearch(int v[], int value, int lo, int hi){
     if(hi<lo)
          return -1;
     int mid = (lo + hi)/2;
     if(v[mid] == value)
          return mid;
     if(v[mid] < value){
          binarySearch(v, value, mid + 1, hi);
     }else{
          binarySearch(v, value, lo, mid - 1);
      }
}
```