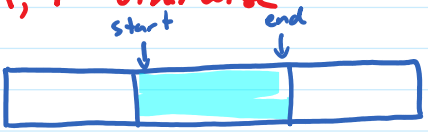
Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[\text{len}(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise

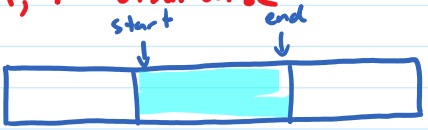
```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:
a) $0 \leq start \leq len(A)$
b) $-1 \leq end \leq len(A)-1$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

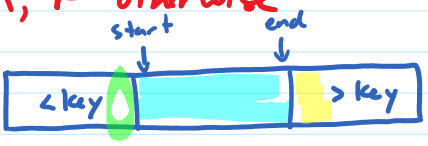
INVARIANT:

a) $0 \leq start \leq len(A)$
b) $-1 \leq end \leq len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e) $end - start + 1 \leq len(A) - n$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[\text{len}(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

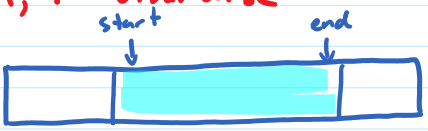
c) $key > A[start-1]$

d) $key < A[end+1]$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

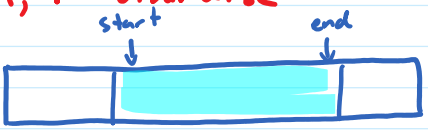
d) $end = len(A)-1$ or $key < A[end+1]$

e)

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

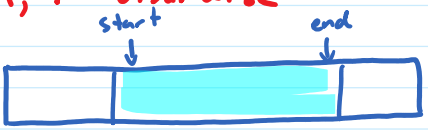
a) $0 \leq start \leq len(A)$
b) $-1 \leq end \leq len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e)

Basis ($n = 0$): $start = 0$ and $end = len(A)-1$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$
b) $-1 \leq end \leq len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e)

Basis ($n = 0$): start = 0 and end = len(A) − 1

$0 \leq len(A)$ $0 \leq len(A)$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \le start \le len(A)$

b) $-1 \le end \le len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Basis ($n = 0$): $start = 0$ and $end = len(A)-1$

$0 = start = 0 \le len(A)$

$0 \le len(A)$

$-1 \le len(A)-1 = end = len(A)-1$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \le start \le len(A)$

b) $-1 \le end \le len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Basis ($n = 0$): $start = 0$ and $end = len(A)-1$

$0 = start = 0 \le len(A)$

$0 \le len(A)$

$-1 \le len(A)-1 = end = len(A)-1$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \leq end$ and $A\left[\left\lfloor \frac{start+end}{2} \right\rfloor\right] \neq key$

$0 \leq start_{na} \leq len(A) - 1$

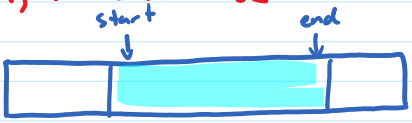
$0 \leq end_{na} \leq len(A) - 1$

Binary Search

```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise



INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \leq end$ and $A\left[\left\lfloor \frac{start+end}{2} \right\rfloor\right] \neq key$

$0 \leq start_{old} \leq len(A)-1$

$0 \leq end_{old} \leq len(A)-1$

$0 \leq start_{old} + end_{old} \leq 2 \cdot (len(A)-1)$

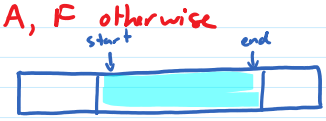
$0 \leq \left\lfloor \frac{start_{old} + end_{old}}{2} \right\rfloor \leq len(A)-1$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise

```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```



INVARIANT:

a) $0 \le start \le len(A)$
b) $-1 \le end \le len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e)

Induction: Suppose INV is true after n iterations and $start \le end$ and $A[\lfloor \frac{start+end}{2} \rfloor] \neq key$

$start_{old} \le end_{old}$

$2 \cdot start_{old} \le start_{old} + end_{old} \le 2 \cdot end_{old}$

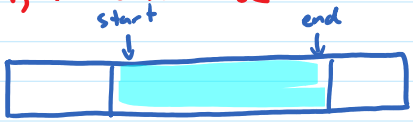
$0 \le start_{old} \le \lfloor \frac{start_{old} + end_{old}}{2} \rfloor \le end_{old} \le len(A)-1$

Binary Search

```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise



INVARIANT:

a) $0 \le start \le len(A)$

b) $-1 \le end \le len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \le end$ and $A[\lfloor \frac{start+end}{2} \rfloor] \ne key$

$$start_{old} \le end_{old}$$

$$2 \cdot start_{old} \le start_{old} + end_{old} \le 2 \cdot end_{old}$$

$$0 \le start_{old} \le \underbrace{\left\lfloor \frac{start_{old} + end_{old}}{2} \right\rfloor}_{mid} \le end_{old} \le len(A)-1$$

Binary Search

```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \leq end$ and $A\left[\left\lfloor \frac{start+end}{2} \right\rfloor\right] \neq key$

$$start_{old} \leq end_{old}$$

$$2 \cdot start_{old} \leq start_{old} + end_{old} \leq 2 \cdot end_{old}$$

$$0 \leq start_{old} \leq \underbrace{\left\lfloor \frac{start_{old} + end_{old}}{2} \right\rfloor}_{mid} \leq end_{old} \leq len(A)-1$$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \leq end$ and $A\left[\left\lfloor \frac{start+end}{2} \right\rfloor\right] \neq key$

2 cases: i) $key < A[end_{new}+1]$ $\quad end_{new} = mid - 1$

$0 \leq mid \leq len(A)-1$

$-1 \leq end_{new} \leq len(A)-2 \leq len(A)-1$

ii) $key > A[mid]$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e)

Induction: Suppose INV is true after n iterations and $start \leq end$ and $A\left[\left\lfloor \frac{start+end}{2} \right\rfloor\right] \neq key$

2 cases: i) $key > A[mid]$ $\quad start_{new} = mid + 1$ $\quad$ so $key > A[start_{new} - 1]$

$0 \leq mid \leq len(A)-1$

$1 \leq mid+1 \leq len(A)$

$0 \leq 1 \leq start_{new} \leq len(A)$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \le start \le len(A)$
b) $-1 \le end \le len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e) $end - start + 1 \le len(A) - n$

Termination: when $n = len(A)$

$end - start + 1 \le len(A) - n = len(A) - len(A) = 0$

$end - start + 1 = 0$

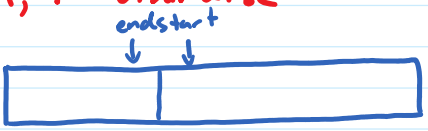
$start = end + 1$

$start > end$

Binary Search

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \leq start \leq len(A)$

b) $-1 \leq end \leq len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e) $end - start + 1 \leq len(A) - n$

Postcondition: 2 cases: i) start > end

$A[start-1] < key < A[end+1]$

ii) $start \leq end$ and $A[\lfloor \frac{start+end}{2} \rfloor] = key$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise


```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \le start \le len(A)$
b) $-1 \le end \le len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e) $end - start + 1 \le len(A) - n$

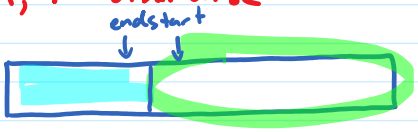
Postcondition: 2 cases: i) $start > end$ $A[0] \le \cdots \le A[start-1] < key < A[end+1] \le \cdots \le A[len(A)-1]$

ii) $start \le end$ and $A[\lfloor \frac{start+end}{2} \rfloor] = key$

Binary Search

PRE: $A[0] \le A[1] \le \cdots \le A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise




```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

INVARIANT:

a) $0 \le start \le len(A)$

b) $-1 \le end \le len(A)-1$

c) $start = 0$ or $key > A[start-1]$

d) $end = len(A)-1$ or $key < A[end+1]$

e) $end - start + 1 \le len(A) - n$

Postcondition: 2 cases: i) $start > end$

$start - 1 \ge end$

$\underbrace{A[0] \le \cdots \le A[start-1]}_{A[end] \text{ is in here}} < key < A[end+1] \le \cdots \le A[len(A)-1]$

$key \ne A[0], \ldots, A[len(A)-1]$

should return F

ii) $start \le end$ and $A\left[\left\lfloor \frac{start + end}{2} \right\rfloor\right] = key$

Binary Search

```
binarySearch(A, key)
  start ← 0
  end ← len(A) - 1;
  while (start <= end and A[(start + end) / 2] != key) {
      mid = (start + end) / 2;
      if (key < A[mid]) {
          end ← mid - 1;
      }
      else {
          start ← mid + 1;
      }
  }
  if (start > end) return false;
  else return true;
```

PRE: $A[0] \leq A[1] \leq \cdots \leq A[len(A)-1]$ (A is sorted)

POST: returns T if key is in A, F otherwise

INVARIANT:
a) $0 \leq start \leq len(A)$
b) $-1 \leq end \leq len(A)-1$
c) $start = 0$ or $key > A[start-1]$
d) $end = len(A)-1$ or $key < A[end+1]$
e) $end - start + 1 \leq len(A) - n$

Postcondition: 2 cases: i) $start > end$

$start - 1 \geq end$

$A[0] \leq \cdots \leq A[start-1] < key < A[end+1] \leq \cdots \leq A[len(A)-1]$

($A[end]$ is in here)

$key \neq A[0], \ldots, A[len(A)-1]$

should return F

ii) $start \leq end$ and $A[\lfloor \frac{start + end}{2} \rfloor] = key$

so key is at index mid - should return T