Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Searching

Quick Recap of Relevant Topics

IIT Bombay

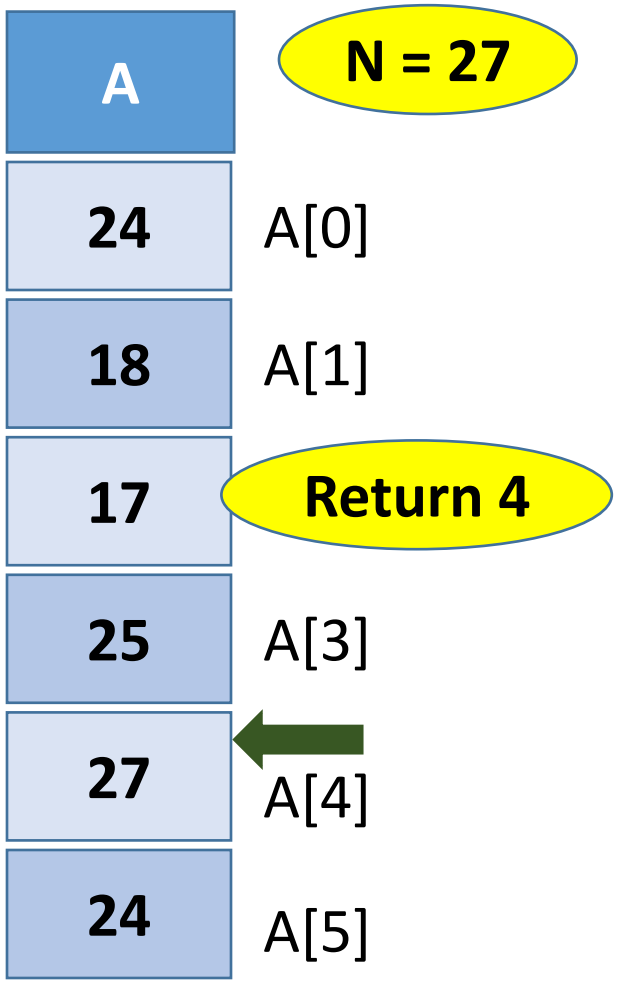
- Sorting integers
  - Selection sort
  - Merge sort
  - Counting “basic” steps in sorting an array
- Sorting strings and other data types
  - Same techniques apply
  - Appropriate comparison operator needed

Overview of This Lecture

IIT Bombay

- Searching integers
  - Linear search
  - Binary search
- Searching strings and other data types

The Searching Problem



Given an array A of integers and a candidate integer N

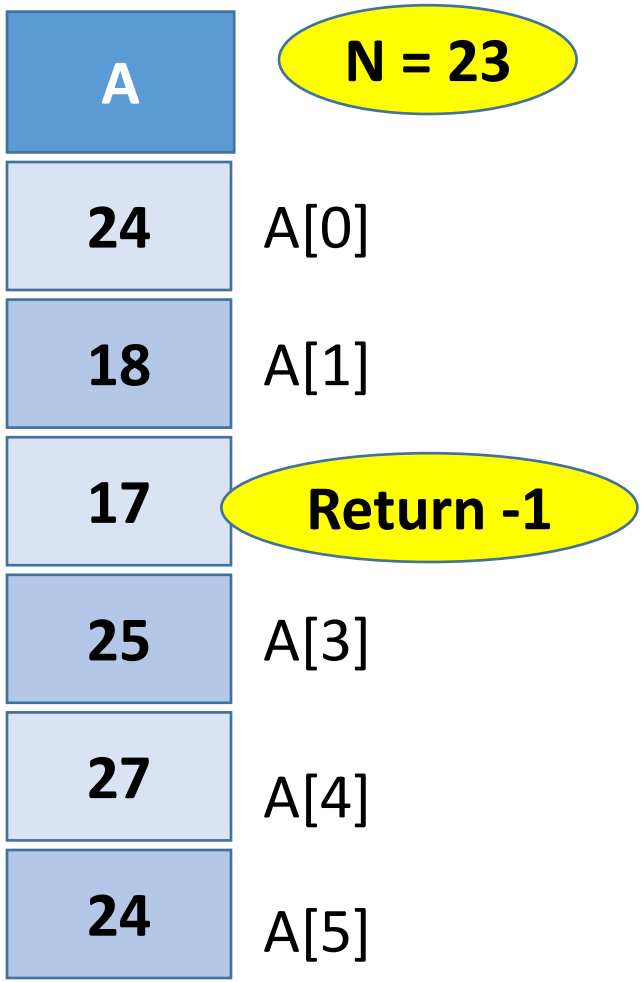
Is N present in A? **Yes/No question**

More interesting

If N is present in A, return its index in A else return -1

[Yes/No answer can be derived]

The Searching Problem



Given an array A of integers and a candidate integer N

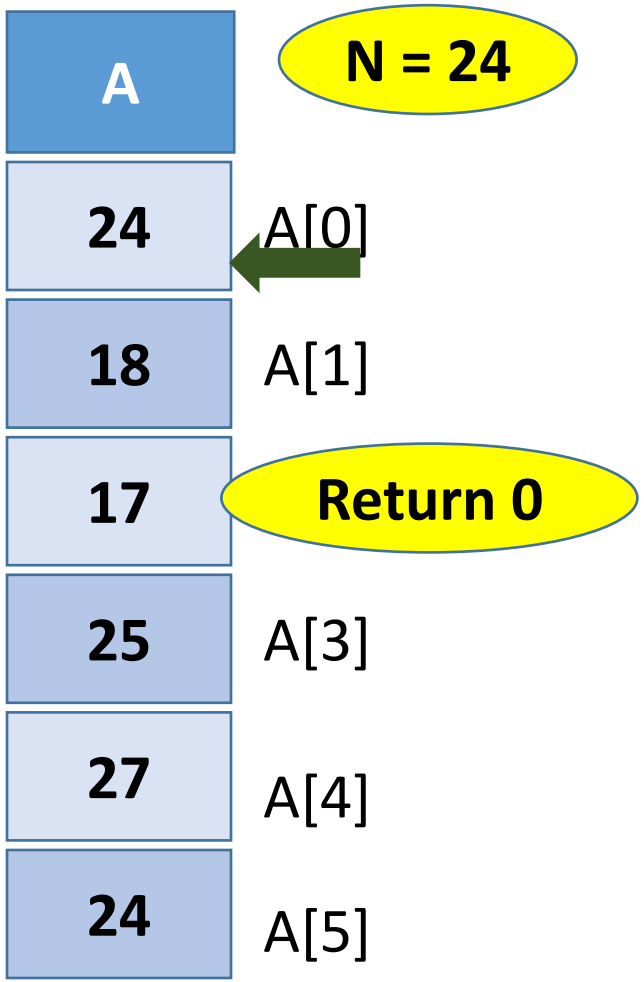
Is n present in A? **Yes/No question**

More interesting

If n is present in A, return its index in A else return -1

[Yes/No answer can be derived]

The Searching Problem



Given an array A of integers and a candidate integer N

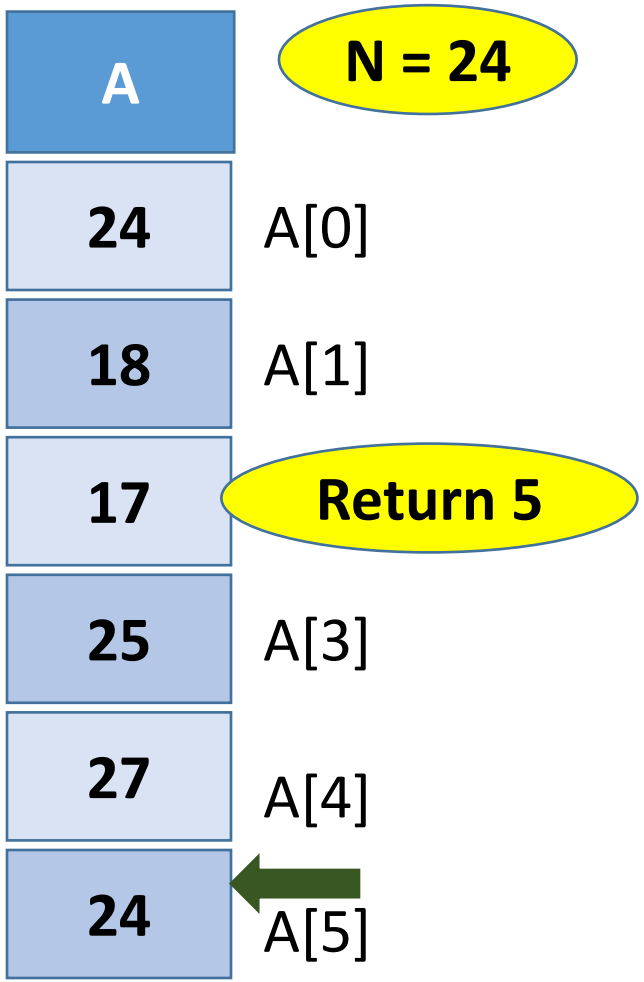
Is n present in A? **Yes/No question**

More interesting

If n is present in A, return its index in A else return -1

[Yes/No answer can be derived]

The Searching Problem



Given an array A of integers and a candidate integer N

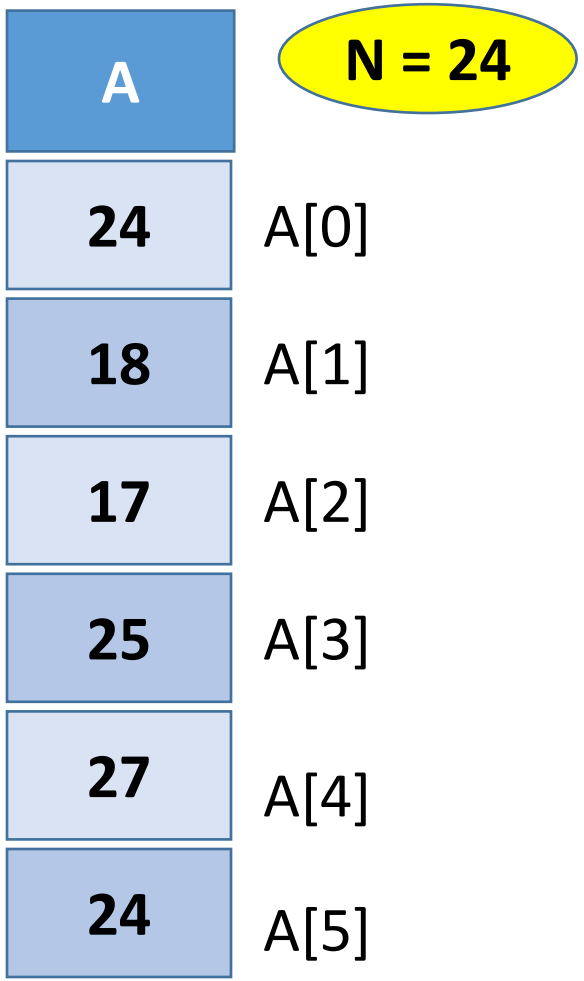
Is n present in A? **Yes/No question**

More interesting

If n is present in A, return its index in A else return -1

[Yes/No answer can be derived]

The Searching Problem



Given an array A of integers and a candidate integer N

In case of multiple matches, index of any one can be returned

n A

else return -1

[Yes/No answer can be derived]

Linear Search

- Check each element of the array
- Stop on finding first match and output index
- If array exhausted and no match found, return -1

Linear Search in C++

IIT Bombay

```
int main() {
  int i, n, A[100];  // Declarations
  cout << "Give size of array: "; cin >> n; // Read and validate inputs
  if ((n > 100) || (n <= 0)) { cout << "Invalid input!" << endl; return -1; }
  cout << "Give " << n << " positive integers in array." << endl;
  for (i = 0; i < n; i++) {cin >> A[i]; } // Read elements of array A
   … Code to search …
  return 0;
}
```

Linear Search in C++

IIT Bombay

```
int main() {
   … Declarations, reading inputs and validation …
   int srchElement, index;
   do {
      cout << “Give element to search (-1 to exit): “; cin >> srchElement;
      if (srchElement == -1) break;
      index = linearSearch(A,  n,  srchElement);
      if (index == -1) { cout << srchElement << “ not present!” << endl;}
      else { cout <<  srchElement << “present at index “ << index << endl;}
   }  while (true);
   return 0;
}
```

Linear Search in C++

IIT Bombay

```
int main() {
  … Declarations, reading inputs and validation …
  int srchElement, index;
  do {
    cout << “Give element to search (-1 to exit): “; cin >> srchElement;
    if (srchElement == -1) break;
    index = linearSearch(A,  n,  srchElement);
    if (index == -1) { cout << srchElement << “ not present!” << endl;}
    else { cout <<  srchElement << “present at index “ << index << endl;}
  }  while (true);
  return 0;
}
```

Linear Search in C++

IIT Bombay

```
int linearSearch(int A[], int n, int srchElement) {
   int i;
   for (i = 0; i < n; i++) {
     if (A[i] == srchElement) { return i; }
   }
   return -1;
}
```

“Basic” Steps In Searching

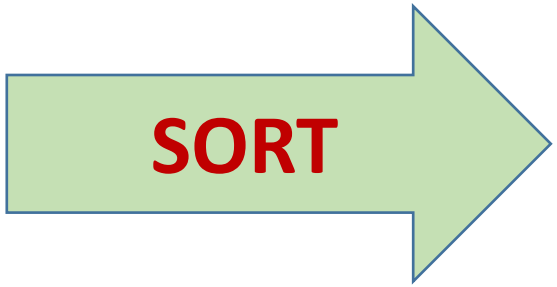
- Comparing an element of array A with the searched element and incrementing index
- Count of “basic” steps
  - At most n “basic” steps to search in an array of size n
- Can we do better?

The Searching Problem

IIT Bombay

A	
24	A[0]
18	A[1]
17	A[2]
25	A[3]
27	A[4]
24	A[5]

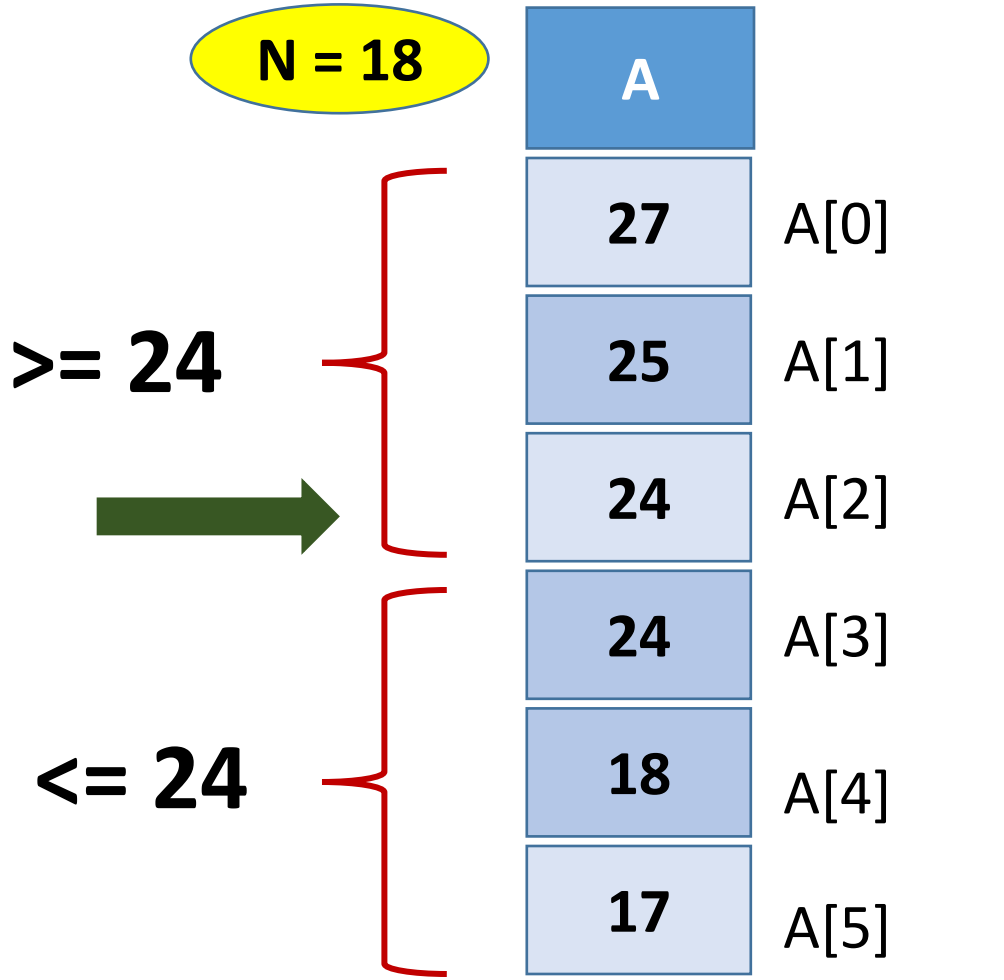
N = 18



A	
27	A[0]
25	A[1]
24	A[2]
24	A[3]
18	A[4]
17	A[5]

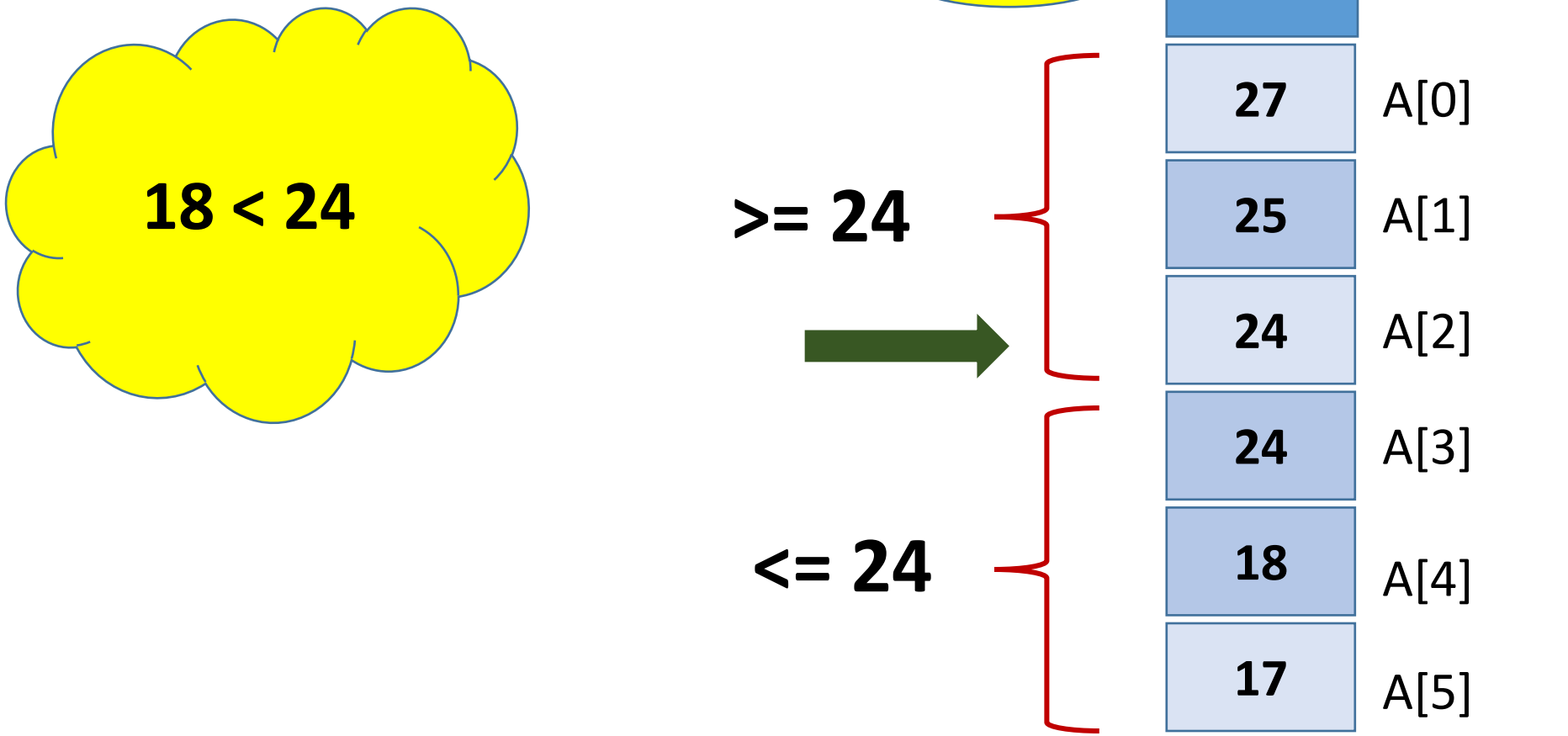
The Searching Problem

IIT Bombay



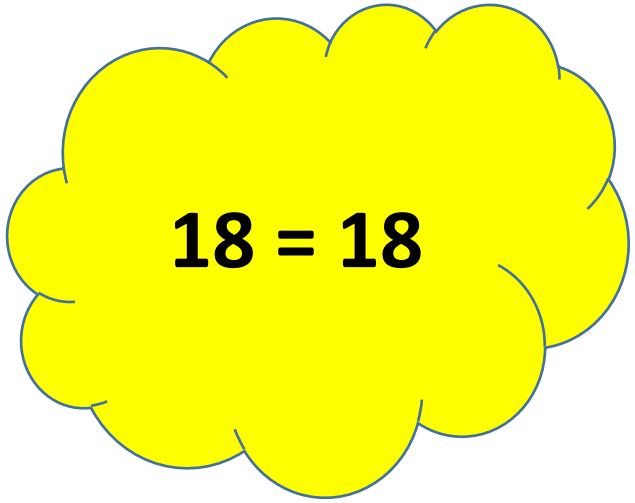
The Searching Problem

IIT Bombay

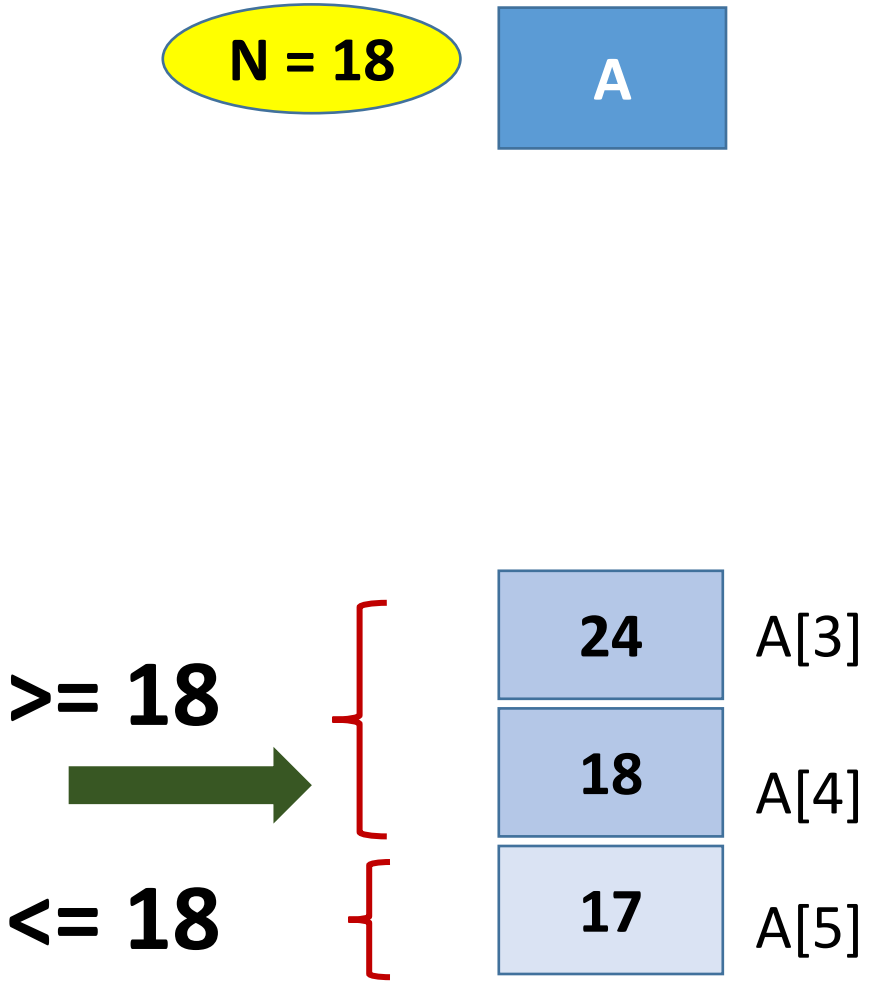


The Searching Problem

IIT Bombay



We are done!



General Idea

IIT Bombay

- Sort the array (A[0] … A[n-1]) in increasing order
- Check search element (m) with element at A[n/2]
- If m equals A[n/2], we are done (return n/2)
- If m < A[n/2], search for m in A[0] … A[n/2 - 1]
- If m > A[n/2], search for m in A[n/2] … A[n-1]
- If A has 1 element and m does not match that, return -1

Termination Case

Termination Case

Recursion

Recursion

General Idea

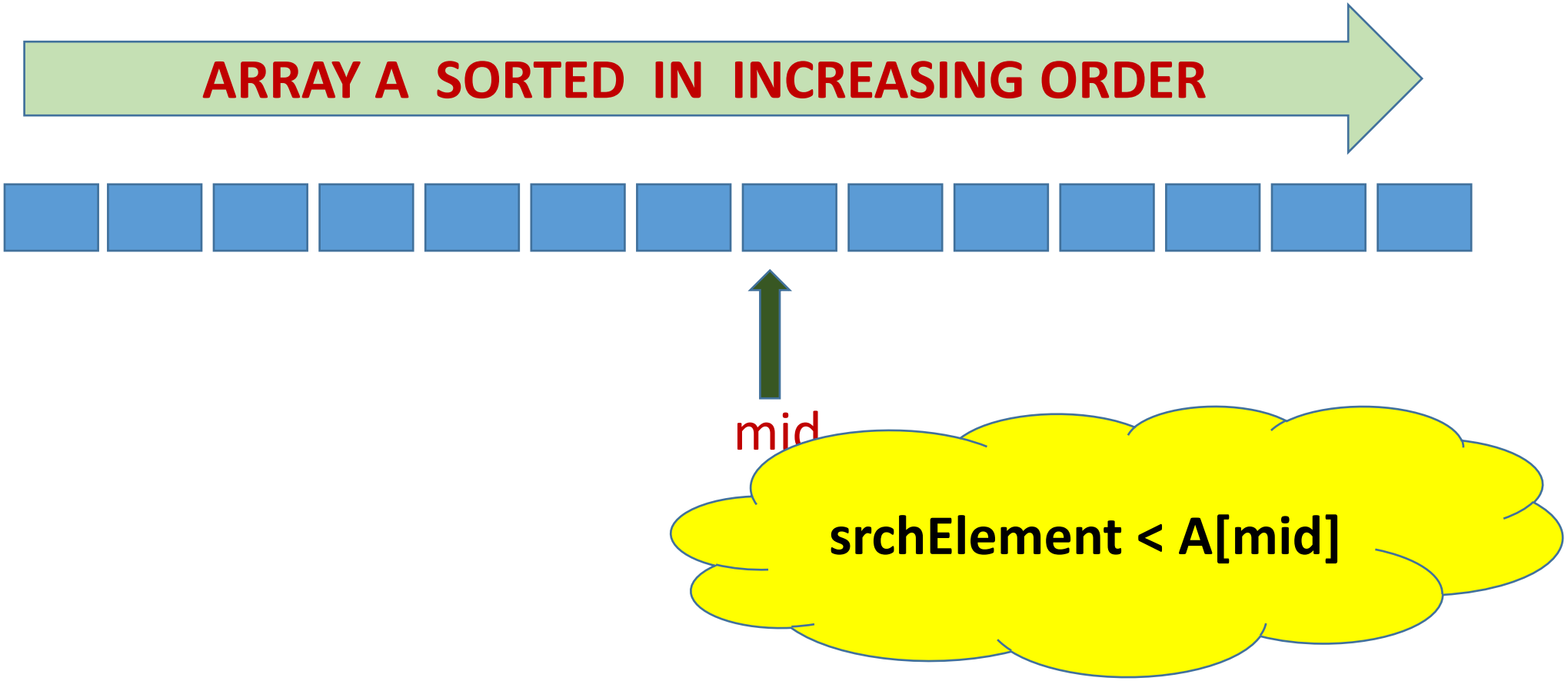
IIT Bombay

- Sort the array (A[0] … A[n-1]) in increasing order
- Check searc /2]
- If m equals
- If m < A[n/
- If m > A[n/]
- If A has 1 el , return -1

BINARY SEARCH

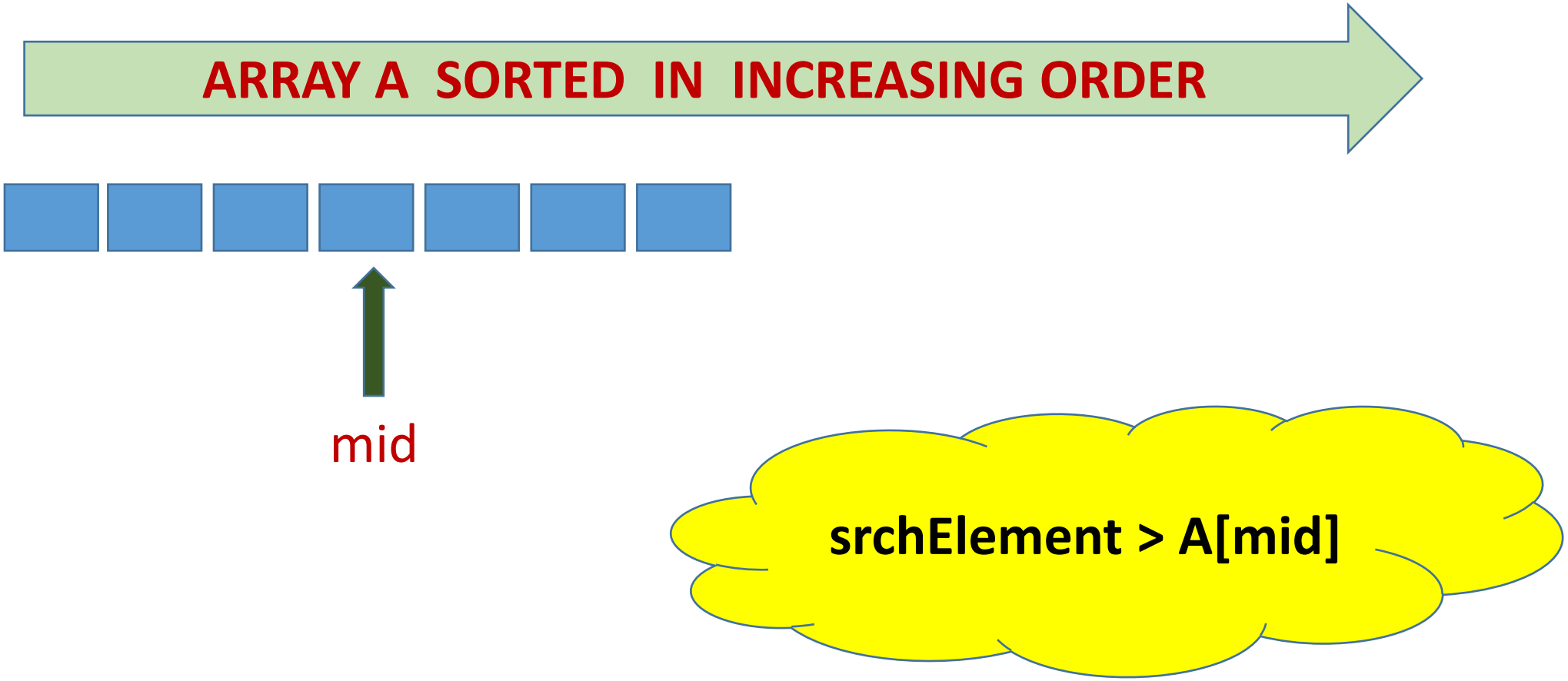
Binary Search In Action

IIT Bombay



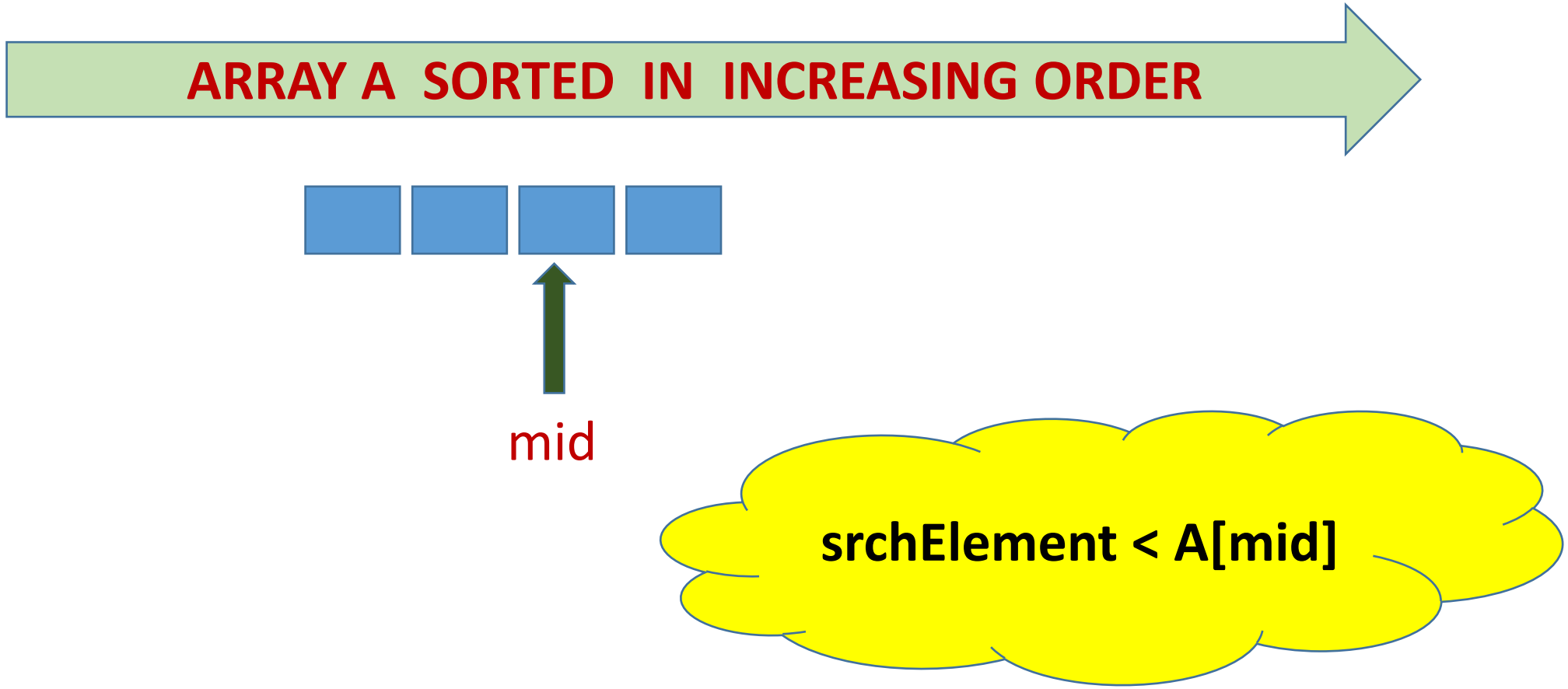
Binary Search In Action

IIT Bombay



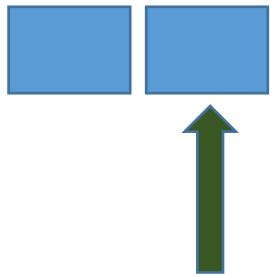
Binary Search In Action

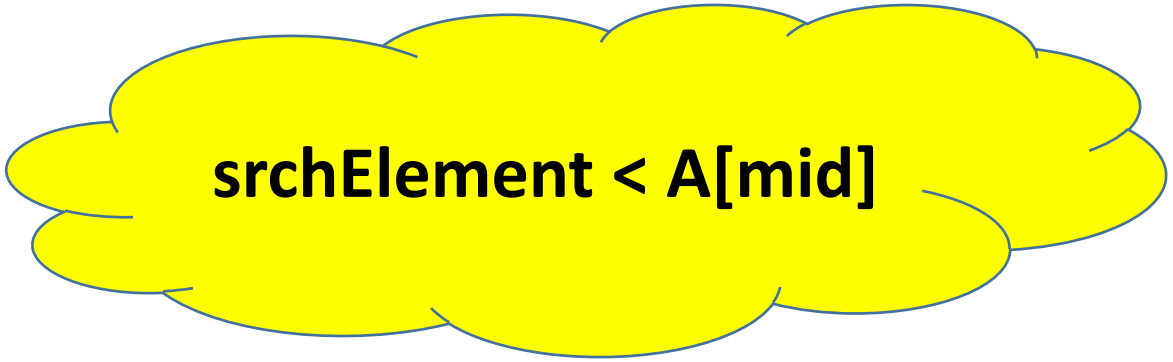
IIT Bombay



Binary Search In Action

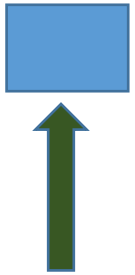
IIT Bombay



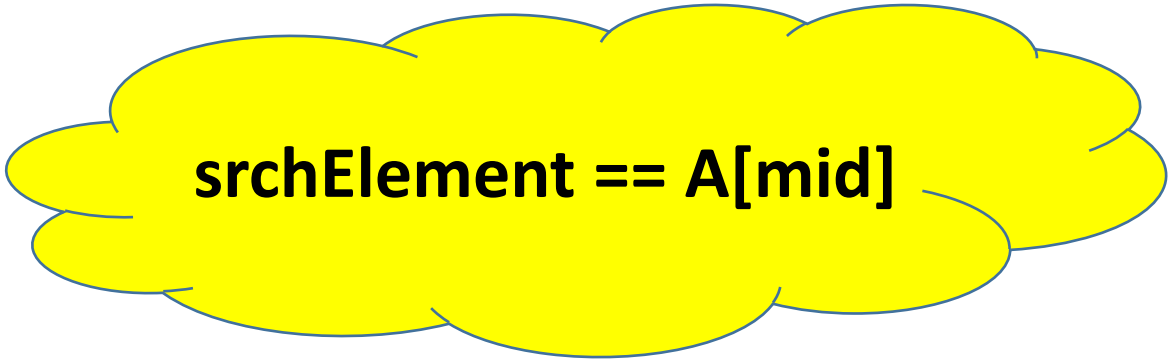


Binary Search In Action

IIT Bombay



mid



Binary Search in C++

IIT Bombay

```
// PRECONDITION:  A[start] … A[end – 1] sorted in increasing order
int binarySearch(int A[], int start, int end, int srchElement) {
   if (end == start + 1) { // Array A has exactly 1 element
       if (A[start] == srchElement) { return start; }
      else { return -1; }
   }
   int mid = (start + end)/2;
   if (A[mid] == srchElement) { return mid; }
   else {  if (A[mid] < srchElement) { return binarySearch(A, mid, end, srchElement); }
            else  { return binarySearch(A, start, mid, srchElement); }
   }
}
// POSTCONDITION: If srchElement in A[start] … A[end-1], return its index, else -1
```

Binary Search in C++


```
// PRECONDITION:  A[start] … A[end – 1] sorted in increasing order
int binarySearch(int A[], int start, int end, int srchElement) {
  if (end == start + 1) { // Array A has exactly 1 element
      if (A[start] == srchElement) { return start; }
     else { return -1; }
  }
  int mid = (start + end)/2;
  if (A[mid] == srchElement) { return mid; }
  else {  if (A[mid] < srchElement) { return binarySearch(A, mid, end, srchElement); }
          else  { return binarySearch(A, start, mid, srchElement); }
  }
}
// POSTCONDITION: If srchElement in A[start] … A[end-1], return its index, else -1
```

Binary Search in C++

IIT Bombay

```
// PRECONDITION:  A[start] … A[end – 1] sorted in increasing order
int binarySearch(int A[], int start, int end, int srchElement) {
  if (end == start + 1) { // Array A has exactly 1 element
     if (A[start] == srchElement) { return start; }
    else { return -1; }
  }
  int mid = (start + end)/2;
  if (A[mid] == srchElement) { return mid; }
  else {  if (A[mid] < srchElement) { return binarySearch(A, mid, end, srchElement); }
          else  { return binarySearch(A, start, mid, srchElement); }
  }
}
// POSTCONDITION: If srchElement in A[start] … A[end-1], return its index, else -1
```

What About Searching Other Data Types

- Exactly same technique
- Sort array
  - Use an appropriate comparison operator
  - lexEarlier(s1, s2) for strings
  - Custom comparison operator for other data types
- Use same comparison operator for searching
  - Decide which half of array to recurse on based on output of comparison operator

Count Of "Basic" Steps

IIT Bombay

- Let T_n be maximum count of steps when searching in array of size n
- $T_n = T_{n/2} + 1; \quad T_1 = 1$
- Solution: $T_n \approx \lceil \log_2 n \rceil$

Summary

IIT Bombay

- Searching in an array of integers
  - Linear search
  - Binary search
  - Sorting helps searching
  - Counting “basic” steps in searching
- Searching in an array of other data types