CS1100 – Introduction to Programming

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lecture 25

Searching and Sorting

Searching for an element

15	8	3	12	30	7	9	17	32	19

Searching for an element

15	8	3	12	30	7	9	17	32	19

Task: Search for a key in the array.

Searching for an element

15	8	3	12	30	7	9	17	32	19

Task: Search for a key in the array.

```
int n, key;
for (int i = 0; i < n; i++) {
    if (A[i] == key) {
       printf("Found %d at index %d\n", key, i);
       break;
    }
}
```

Searching for an element

15	8	3	12	30	7	9	17	32	19

Task: Search for a key in the array.

```
int n, key;
for (int i = 0; i < n; i++) {
    if (A[i] == key) {
        printf("Found %d at index %d\n", key, i);
        break;
    }
}
```

- A linear pass over the array.
- Can this be avoided?

Searching for an element

15	8	3	12	30	7	9	17	32	19

Task: Search for a key in the array.

```
int n, key;
for (int i = 0; i < n; i++) {
    if (A[i] == key) {
        printf("Found %d at index %d\n", key, i);
        break;
    }
}
```

- A linear pass over the array.
- Can this be avoided?
    - No! if the input has no other assumptions.
    - Yes! for example if input is sorted.

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Binary Search

- Check the middle element. If found, break.

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Binary Search

- Check the middle element. If found, break.
- Else decide which part of the array is relevant and repeat.

Can be done since array is sorted!

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Binary Search

- Check the middle element. If found, break.
- Else decide which part of the array is relevant and repeat.

Can be done since array is sorted!

```
int n, key; scanf("%d", &key);
start = 0; end = n-1;
while (______) {
    mid = _______;
    if (A[mid] == key) {
       printf("Found at %d", mid);
       break;
    }
    if (A[mid] > key) {
       start = ______;
    } else {
       end = ______;
    }
}
```

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Binary Search

- Check the middle element. If found, break.
- Else decide which part of the array is relevant and repeat.

Can be done since array is sorted!

```
int n, key; scanf("%d", &key);
start = 0; end = n-1;
while (______) {
    mid = (start + end) / 2;
    if (A[mid] == key) {
       printf("Found at %d", mid);
       break;
    }
    if (A[mid] > key) {
       start = mid+1;
    } else {
       end = mid-1;
    }
}
```

Searching for an element in a sorted array

32	30	19	17	15	12	9	8	7	3

Task: Search for a key in a sorted array.

Binary Search

- Check the middle element. If found, break.
- Else decide which part of the array is relevant and repeat.

Can be done since array is sorted!

```
int n, key; scanf("%d", &key);
start = 0; end = n-1;
while (start <= end) {
    mid = (start + end) / 2;
    if (A[mid] == key) {
       printf("Found at %d", mid);
       break;
    }
    if (A[mid] > key) {
       start = mid+1;
    } else {
       end = mid-1;
    }
}
```

Linear Search versus Binary Search

Exercise for you:

- Take a large sorted array.
- Use linear search to find an element not present in the array.
  needs n comparisons
- Use binary search to find the same element in the array.
  count and print the number of comparisons
  check performance in terms of time

Coding Binary Search

```c
#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {
  if (high >= low) {
    int mid = low + (high - low) / 2;

    // If found at mid, then return it
    if (array[mid] == x)
      return mid;

    // Search the left half
    if (array[mid] > x)
      return binarySearch(array, x, low, mid - 1);

    // Search the right half
    return binarySearch(array, x, mid + 1, high);
  }
  return -1;
}

int main(void) {
  int array[] = {3, 4, 5, 6, 7, 8, 9};
  int n = sizeof(array) / sizeof(array[0]);
  int x = 4;
  int result = binarySearch(array, x, 0, n - 1);
  if (result == -1)
    printf("Not found");
  else
    printf("Element is found at index %d", result);
}
```