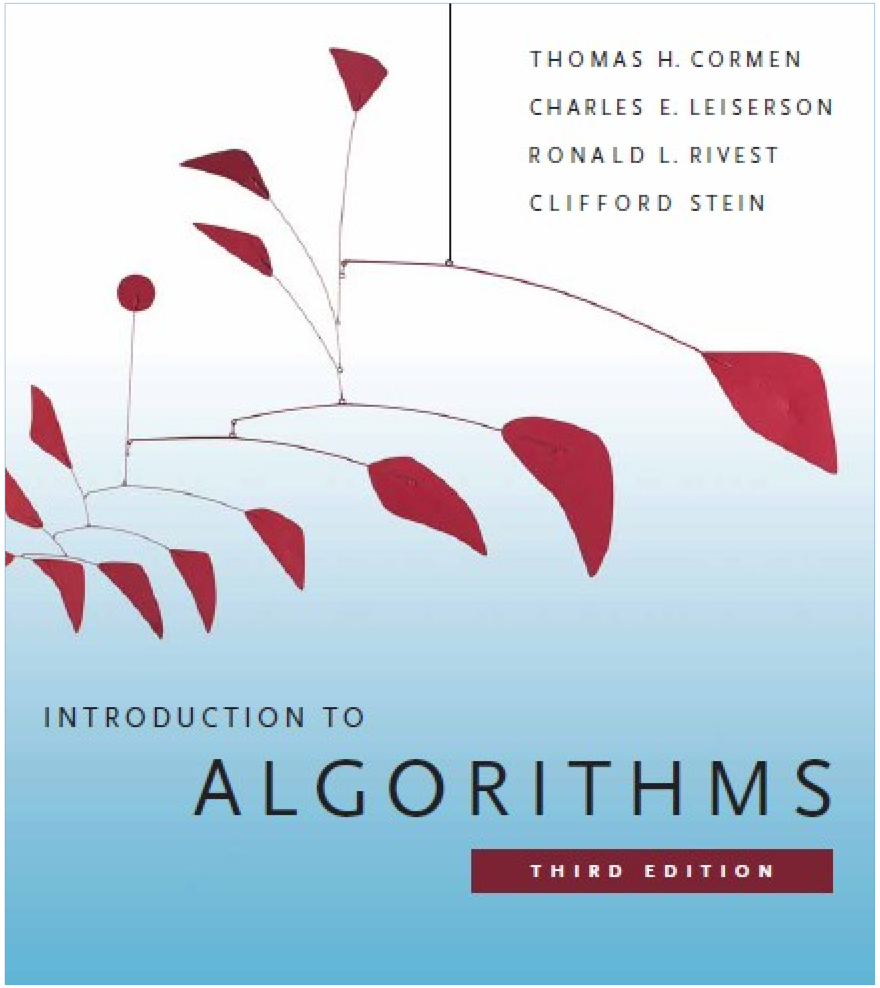


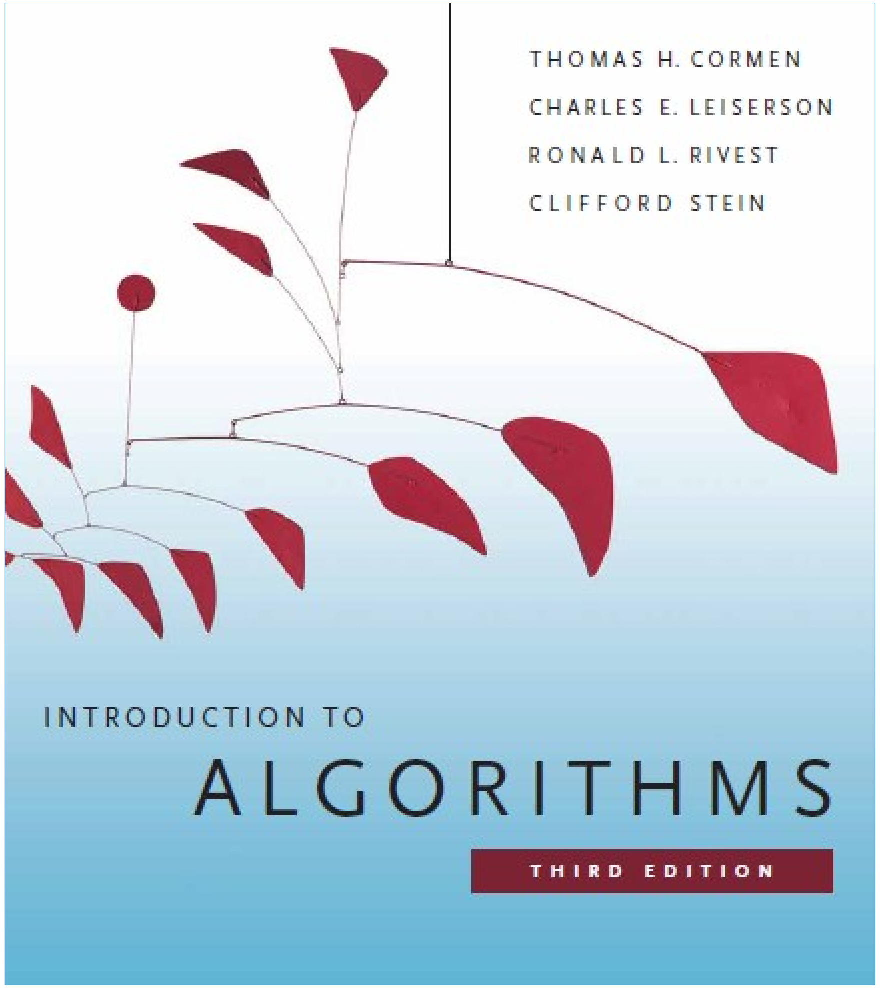
Searching

- Searching Technique
- Sequential Search
- Binary Search

Searching

Why searching is important?

- Most frequent operation in a computer.
- Knuth says over 25% of computer time in 60’s was used in searching.
- Generally we search for solutions or answers in databases or in computation intensive problems.
- Our shops are arranged to minimize searching time, as is the chess board or household goods, admission test results or even dictionaries.



Searching

- Searching Technique
- Sequential Search
- Binary Search

Searching Algorithm

Problem: Given a sorted array and a key, find index of the key in the array.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	14	25	33	35	40	49	55	58	60	69	77	88	91	93

Solve: Sequential Search

Complexity: ?

Sequential_search(A, n, z, index)

```
i = 1, index = -1
While i <= n  and  A(i) ≠ z  do
    i++
enddo
If i <= n then
    index = i
endif
```

Sequential Search

Problem: Can you do better? (hint: every instruction takes time)

```
Sequential_search_1(A, n, z,
    index)
i = 1, index = -1
While i <= n  and  A(i) ≠ z  do
    i++
enddo
If i <= n then
    index = i
endif
```

```
Sequential_search_2(A, n, z,
    index)
A(n+1) = z, i = 1, index = -1
While A(i) ≠ z  do
    i++
enddo
If i <= n then
    index = i
endif
```

Sequential Search

Why algorithm 2 is better?

- In *sequential_search_2* algorithm *i* <= *n* index comparison is not required
- Even if the element were not in the list it has been kept at (n+1)st location we are not going to run out of the array.

```
Sequential_search_2(A, n, z,
    index)
A(n+1) = z, i = 1, index = -1
While A(i) ≠ z  do
    i++
enddo
If i <= n then
    index = i
endif
```

Sequential Search

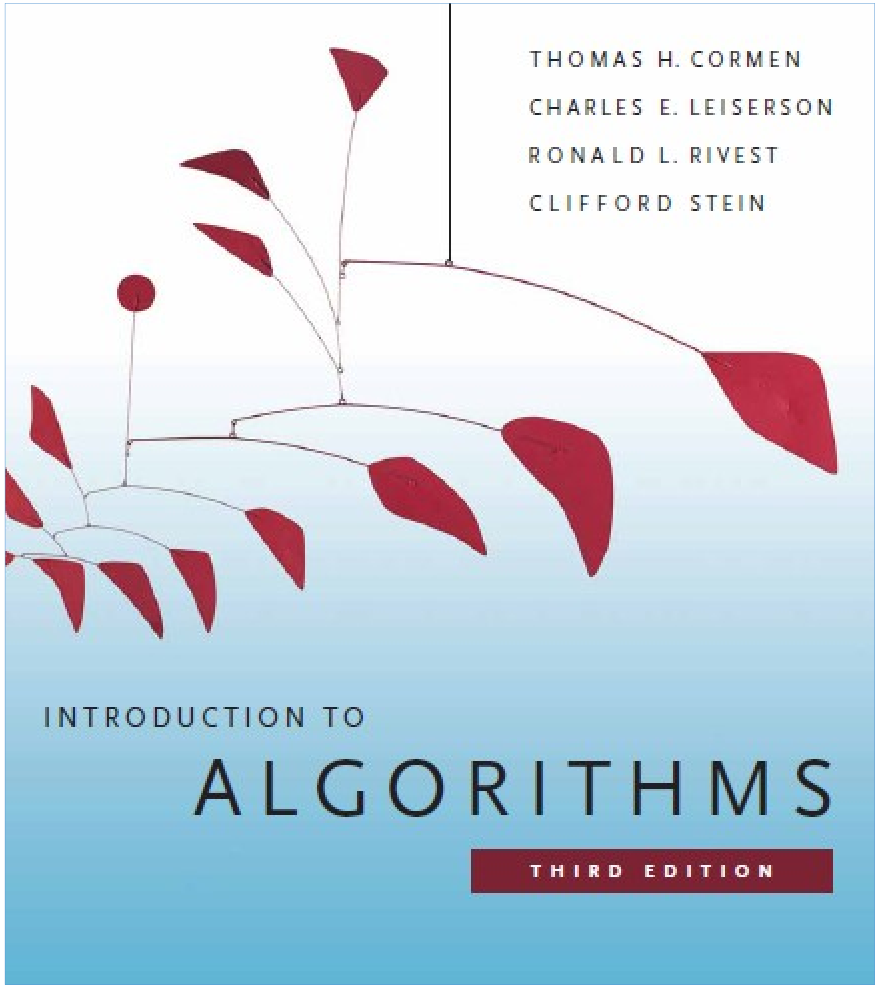
Sequential Search complexity

Operation	Best Case	Worst Case	Average Case
Successful Search	1	n	(n+1) / 2
Unsuccessful Search	n	n	n

Searching Algorithm

Sequential Search complexity

- However, when n is very large and we need to search out too many elements sequential search would be too costly to pursue.
- Imagine had your roll numbers in the successful lists of DU admission test or that at BUET appeared in no order how difficult would it have been for you to go through the whole list, on the average half of it, to find your name or after the list is exhausted you come to learn that you are unsuccessful.



Searching

- Searching Technique
- Sequential Search
- Binary Search

Guessing Game?

Binary Search

Problem: Given a sorted array and a key, find index of the key in the array.

Binary search. Compare key against middle entry.

- Too small, go left.
- Too big, go right.
- Equal, found.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	14	25	33	35	40	49	55	58	60	69	77	88	91	93

Binary Search

- Loop invariant?

If key appears in the array a[], then a[lo] ≤ key ≤ a[hi].

```
int binarySearch(int[] a, int key)
{
    int lo = 0, hi = a.length-1;
    while (lo <= hi)
    {
        int mid = lo + (hi - lo) / 2;
        if (key < a[mid])
            hi = mid - 1;
        else if (key > a[mid])
            lo = mid + 1;
        else return mid;
    }
    return -1;
}
```

Binary Search : mathematical analysis

- Proposition. Binary search uses at most 1 + lg N key compares to search in a sorted array of size N.
  - Step 0: N
  - Step 1: N/2 ~ $N/2^1$
  - Step 2: N/4 ~ $N/2^2$
  - …….
  - Step k: $N/2^k$ ~ 1 [checking last element]

Binary Search : mathematical analysis

- Find the value of k.
- Step 0: N
- Step 1: N/2 ~ $N/2^1$
- Step 2: N/4 ~ $N/2^2$
- …….
- Step k: $N/2^k$
- [checking last element]

Binary Search : mathematical analysis

- Find the value of k.
- Step 0: N
- Step 1: N/2 ~ $N/2^1$
- Step 2: N/4 ~ $N/2^2$
- …….
- Step k: $N/2^k$
- [checking last element]

$$\frac{N}{2^k} = 1$$

or, $N = 2^k$

or, $log_2 N = log_2 2^k$

or, $k log_2 2 = log_2 N$

or, $k = log_2 N$

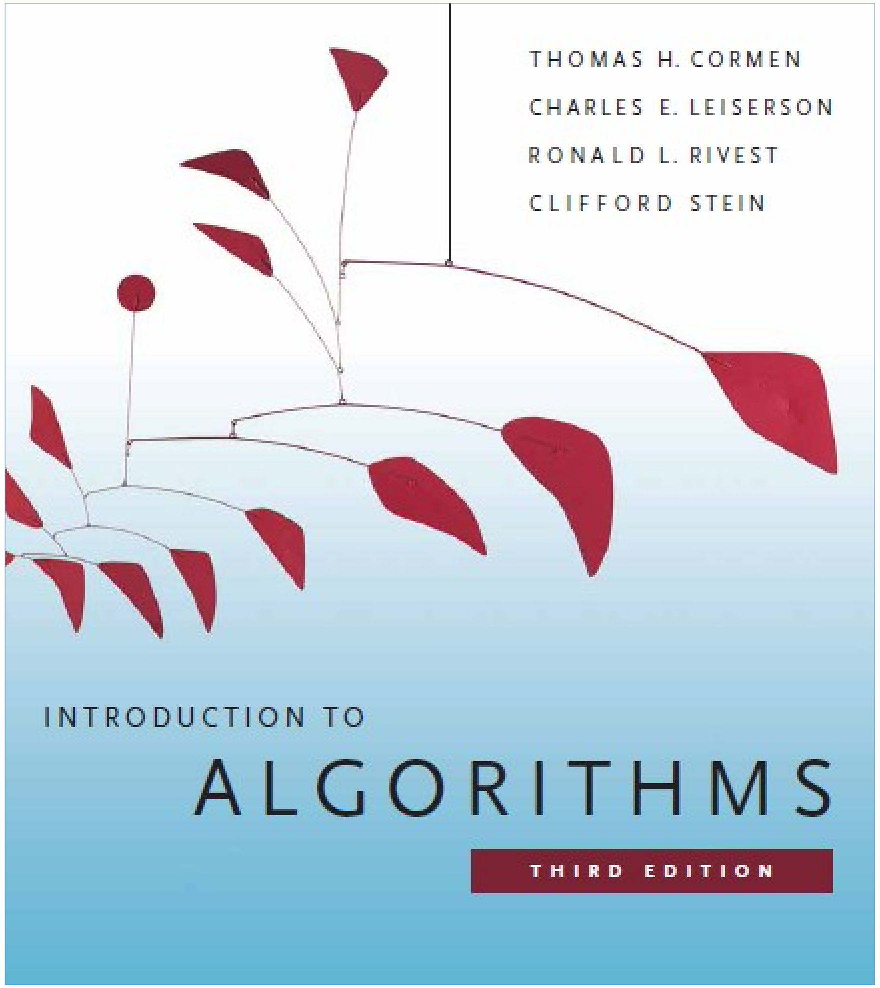
Binary Search : Which algorithm is better?

```
int binarySearch_1(int a[], int key)
{
 int lo = 0; hi = a.length-1;
 while (lo <= hi)
 {
  int mid = lo + (hi - lo) / 2;
  if(key == a[mid])
   return mid;
  else if(key < a[mid])
   hi = mid - 1;
  else if(key > a[mid])
   lo = mid + 1;
 }
 return -1;
}
```

```
int binarySearch_2(int a[], int key)
{
 int lo = 0; hi = a.length-1;
 while (lo <= hi)
 {
  int mid = lo + (hi - lo) / 2;
  if(key > a[mid])
   lo = mid + 1;
  else if(key < a[mid])
   hi = mid - 1;
  else if(key == a[mid])
   return mid;
 }
 return -1;
}
```

Other searches

- Dictionary searching is interpolation search.
- Hashing is another way of searching efficiently. We can store elements at a location that is function of the element. Say we want to store “dog” we compute location as 4+15+7=26. If number of location cannot exceed m then we take 26 mod m and store “dog” there if it is empty. If not follow some hash collision resolution procedure.



Searching

- Searching Technique
- Sequential Search
- Binary Search