Welcome to the Class

Department of Computing and Information System

Data Structure

Md. Selim Hossain
Senior Lecturer
Department of Computing and Information System
Daffodil International University (DIU),Dhaka, Bangladesh

Algorithm Analysis

- Data structures
  - Methods of organizing data in order to facilitate access and modification.
- What is Algorithm?
  - a clearly specified set of simple instructions on the data to be followed to solve a problem
    - Takes a set of values, as input and
    - produces a value, or set of values, as output
  - May be specified
    - In English
    - As a computer program
    - As a pseudo-code
- Program = data structures + algorithms

Algorithm Analysis

- Why need algorithm analysis ?
  - writing a working program is not good enough
  - The program may be inefficient!
  - If the program is run on a large data set, then the running time becomes an issue

Example: Selection Problem

- Given a list of N numbers, determine the *k*th largest, where $k \leq N$.
- Algorithm 1:

  (1) Read N numbers into an array

  (2) Sort the array in decreasing order by some simple algorithm

  (3) Return the element in position k

Algorithm Analysis

- Algorithm 2:

  (1) Read the first k elements into an array and sort them in decreasing order

  (2) Each remaining element is read one by one
  - If smaller than the kth element, then it is ignored
  - Otherwise, it is placed in its correct spot in the array, bumping one element out of the array.

  (3) The element in the kth position is returned as the answer.

Algorithm Analysis

- Which algorithm is better when
  - N =100 and k = 100?
  - N =100 and k = 1?
- What happens when
  - N = 1,000,000 and k = 500,000?
- We come back after sorting analysis, and there exist better algorithms

Algorithm Analysis

- We only analyze *correct* algorithms
- An algorithm is correct
  - If, for every input instance, it halts with the correct output
- Incorrect algorithms
  - Might not halt at all on some input instances
  - Might halt with other than the desired answer

Complexity..

- Predict the amount of resources required:
  - memory: how much space is needed?
  - computational time: how fast the algorithm runs?
- FACT: running time grows with the size of the input
- Input size (number of elements in the input)
  - Size of an array, polynomial degree, # of elements in a matrix, # of bits in the binary representation of the input, vertices and edges in a graph

Def: Running time = the number of primitive operations (steps) executed before termination

- Arithmetic operations (+, -, *), data movement, control, decision making (*if*, *while*), comparison

Complexity..

- We define complexity as a function f(n)
- Worst Case Complexity:
  - the function defined by the *maximum* number of steps taken on any instance of size *n*
- Best Case Complexity:
  - the function defined by the *minimum* number of steps taken on any instance of size *n*
- Average Case Complexity:
  - the function defined by the *average* number of steps taken on any instance of size *n*

Searching is an operation or a technique that helps finds the place of a given element or value in the list. Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not.

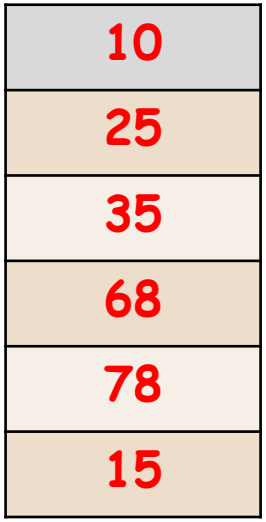
Based on the type of search operation, these algorithms are generally classified into two categories:

1.Sequential Search: In this, the list or array is traversed sequentially and every element is checked. For example: Linear Search.

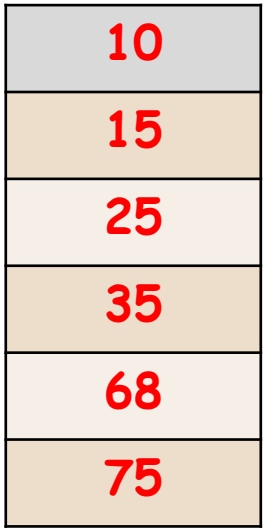
2.Interval Search: These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: Binary Search.

Searching Algorithm

- Depends on the way the information in DATA is organized.
- Linear Search: when the data is unsorted or sorted
- Binary Search: when the data is sorted only!!!



Linear Search



Binary Search

Linear Search

- We have an array DATA with N elements.
- We have to find the LOC of ITEM (Say ITEM=15)

10
25
35
68
78
15

Working Process:

-check element one by one.

-DATA[1]=ITEM ???

-If YES then OK.

-Otherwise DATA[2]=ITEM ???

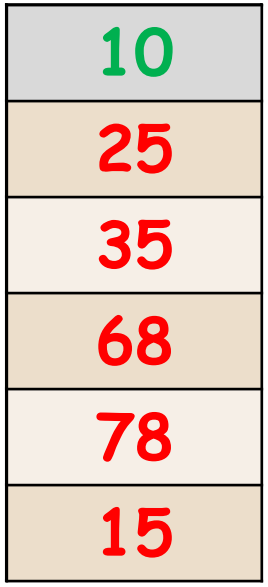
……So on.

Linear search

Element to search: 8

Linear Search

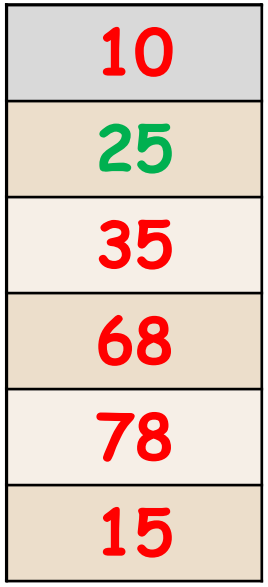
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



1st: K=1 DATA[1]=10.
15!=10 and LOC=0 so K=2

Linear Search

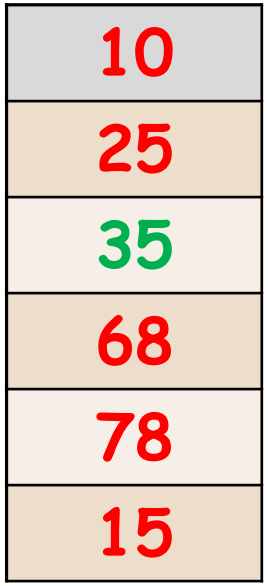
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



2nd: K=2 DATA[2]=25.
15!=25 and LOC=0 so K=3

Linear Search

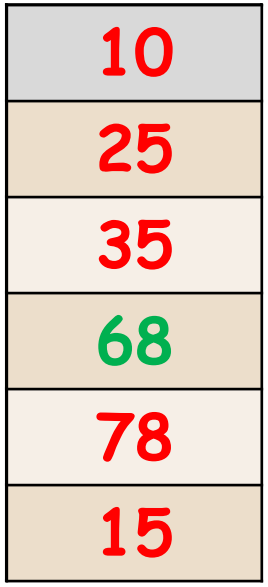
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



3rd: K=3 DATA[3]=35.
15!=35 and LOC=0 so K=4

Linear Search

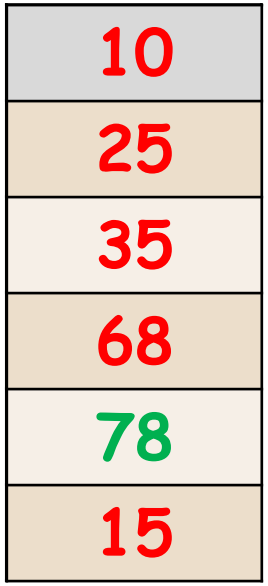
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



4th: K=4 DATA[4]=68.
15!=68 and LOC=0 so K=5

Linear Search

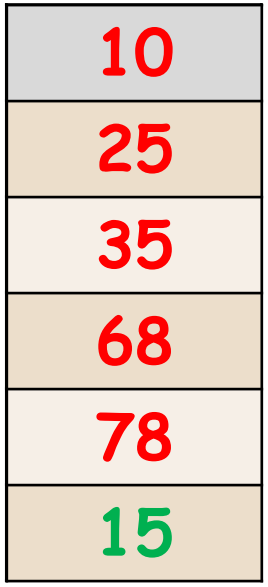
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



4th: K=5 DATA[5]=78.
15!=78 and LOC=0 so K=6

Linear Search

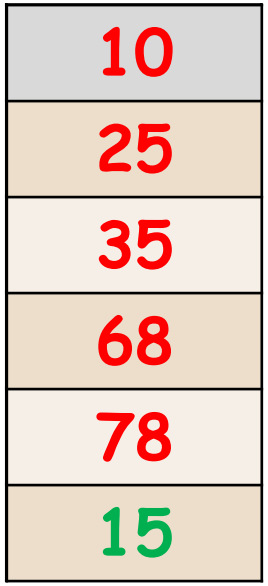
1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



4th: K=6 DATA[6]=15.
15==15 and LOC=6

Linear Search

1. K:=1 and LOC=0.
2. Repeat Steps 3 and 4 while K<=N and LOC=0.
3. If ITEM=DATA[K],then LOC:=K
4. Set K:=K+1
5. If LOC=0 then
   -Write :ITEM is not Found
6. Else
   -Write : ITEM Found at LOC
7. Exit.



4th: K=6 DATA[6]=15.
15==15 and LOC=6

Complexity of linear Search

- Worst case:
  - Occurs when item is in last location of the array.
    - C(n)=n
- Average case:
  - We assume ITEM is in the array.
  - So it can be found at any location.
  - Number of comparison be any of 1,2,3,…n.

- Thanks to All