Search Operation in an Array

Searching refers to the process / operation of finding the location of the ITEM/Value specified by the user in a data structure. The location is basically the index where that item/value is stored.

- Three outcomes are possible:
  - The item does not exist
  - The item exists once in a data structure
  - The item exist more than once in a data structure.

Search (linear) Operation in an Array

Variables:

DATA: Liner Array

N: Number of elements in DATA

ITEM: The value which is to be searched

LOC: Location of ITEM in DATA

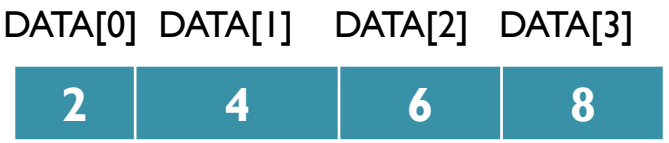
Algorithm:

1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC

Search (linear) Operation in an Array

Algorithm:

1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC



Iteration-1 (N=4,LOC=0,ITEM=2)
Set DATA[4] = 2, Set LOC = 0
Repeat while DATA[0] ≠ ITEM
If LOC = 4
Return 0

Search (linear) Operation in an Array

Algorithm:

1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC

Iteration-1(N=4,LOC=0,ITEM=4)
Set DATA[4] = 4, Set LOC = 0
Repeat while DATA[0] ≠ ITEM
Set LOC =1

[0]	[1]	[2]	[3]	[4]
2	4	6	8	4

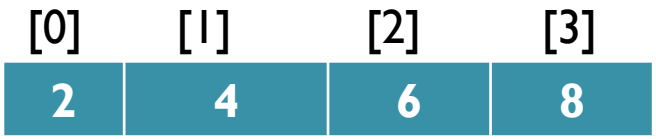
LA[0]	LA[1]	LA[2]	LA[3]
2	4	6	8

Iteration-2(N=4,LOC=1,ITEM=4)
Repeat while DATA[1] ≠ ITEM
If LOC = N
Return 1

Search (linear) Operation in an Array

Algorithm:

1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC



Iteration-1(N=4,LOC=0,ITEM=8)
Set DATA[4] = 8, Set LOC = 0
Repeat while DATA[0] ≠ ITEM
Set LOC =1

Iteration-2(N=4,LOC=1,ITEM=8)
Repeat while DATA[1] ≠ ITEM
Set LOC =2

Iteration-3(N=4,LOC=2,ITEM=8)
Repeat while DATA[2] ≠ ITEM
Set LOC =3

Iteration-4(N=4,LOC=3,ITEM=8)
Repeat while DATA[3] ≠ ITEM
If LOC = N
Return 3

Search (linear) Operation in an Array

Algorithm:

1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC

LA[0]	LA[1]	LA[2]	LA[3]
2	4	6	8

Iteration-1(N=4,LOC=0,ITEM=9)

Set DATA[4] = 9, Set LOC = 0

Repeat while DATA[0] ≠ ITEM

Set LOC =1

Iteration-2(N=4,LOC=1,ITEM=9)

Repeat while DATA[1] ≠ ITEM

Set LOC =2

Iteration-3(N=4,LOC=2,ITEM=9)

Repeat while DATA[2] ≠ ITEM

Set LOC =3

Iteration-4(N=4,LOC=3,ITEM=9)

Repeat while DATA[3] ≠ ITEM

Set LOC =4

Iteration-5(N=4,LOC=4,ITEM=9)

Repeat while DATA[4] ≠ ITEM

If LOC = N, Set LOC = -1

Return -1

Comparison of Search Operation in an Array

Algorithm:

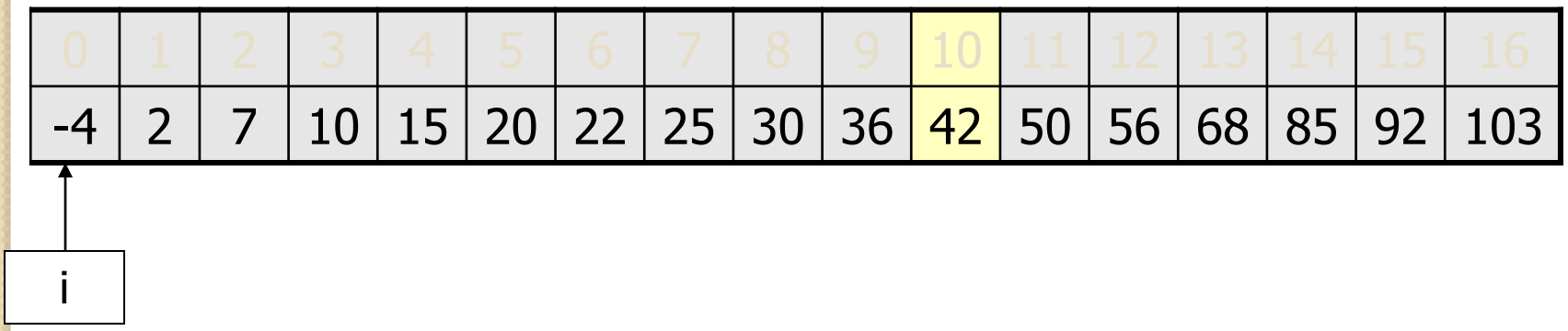
1. Set DATA[N] = ITEM
2. Set LOC = 0
3. Repeat while DATA[LOC] ≠ ITEM
4. Set LOC = LOC+1
5. If LOC = N
6. Set LOC= -1
7. Return LOC

LA[0]	LA[1]	LA[2]	LA[3]
2	4	6	8

Location of Item	At the start of array	At the end of array	In the middle of array	Item does not exist
Number of Iterations	0	3	1	4
Big O Notation		O(N-1)		O(N)

Sequential search

- **sequential search**: Locates a target value in an array / list by examining each element from start to finish.
  - How many elements will it need to examine?
  - Example: Searching the array below for the value **42**:



  - Notice that the array is sorted. Could we take advantage of this?

Binary Search Operation in an Array

- Binary search is an efficient algorithm for finding a value from a sorted list of values.
- It works by repeatedly dividing in half , the portion of the list that could contain the value, until the possible location is narrowed down to just one.
- One of the most common ways to use binary search is to find an item in an array.

(Binary Search) BINARY(DATA, LB, UB, ITEM, LOC)
Here DATA is a sorted array with lower bound LB and upper bound UB, and ITEM is a given item of information. The variables BEG, END and MID denote, respectively, the beginning, end and middle locations of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or sets LOC = NULL.

1. [Initialize segment variables.]
   Set BEG := LB, END := UB and MID = INT((BEG + END)/2).
2. Repeat Steps 3 and 4 while BEG ≤ END and DATA[MID] ≠ ITEM.
3. If ITEM < DATA[MID], then:
   Set END := MID − 1.
   Else:
   Set BEG := MID + 1.
   [End of If structure.]
4. Set MID := INT((BEG + END)/2).
   [End of Step 2 loop.]
5. If DATA[MID] = ITEM, then:
   Set LOC := MID.
   Else:
   Set LOC := NULL.
   [End of If structure.]
6. Exit.

Binary search

- Search 42

(Binary Search) BINARY(DATA, LB, UB, ITEM, LOC)
Here DATA is a sorted array with lower bound LB and upper bound UB, and ITEM is a given item of information. The variables BEG, END and MID denote, respectively, the beginning, end and middle locations of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or sets LOC = NULL.

1. [Initialize segment variables.]
   Set BEG := LB, END := UB and MID = INT((BEG + END)/2).
2. Repeat Steps 3 and 4 while BEG ≤ END and DATA[MID] ≠ ITEM.
3. If ITEM < DATA[MID], then:
   Set END := MID − 1.
   Else:
   Set BEG := MID + 1.
   [End of If structure.]
4. Set MID := INT((BEG + END)/2).
   [End of Step 2 loop.]
5. If DATA[MID] = ITEM, then:
   Set LOC := MID.
   Else:
   Set LOC := NULL.
   [End of If structure.]
6. Exit.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

Beg mid End

Binary search

- Search 42

(Binary Search) BINARY(DATA, LB, UB, ITEM, LOC)
Here DATA is a sorted array with lower bound LB and upper bound UB, and ITEM is a given item of information. The variables BEG, END and MID denote, respectively, the beginning, end and middle locations of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or sets LOC = NULL.

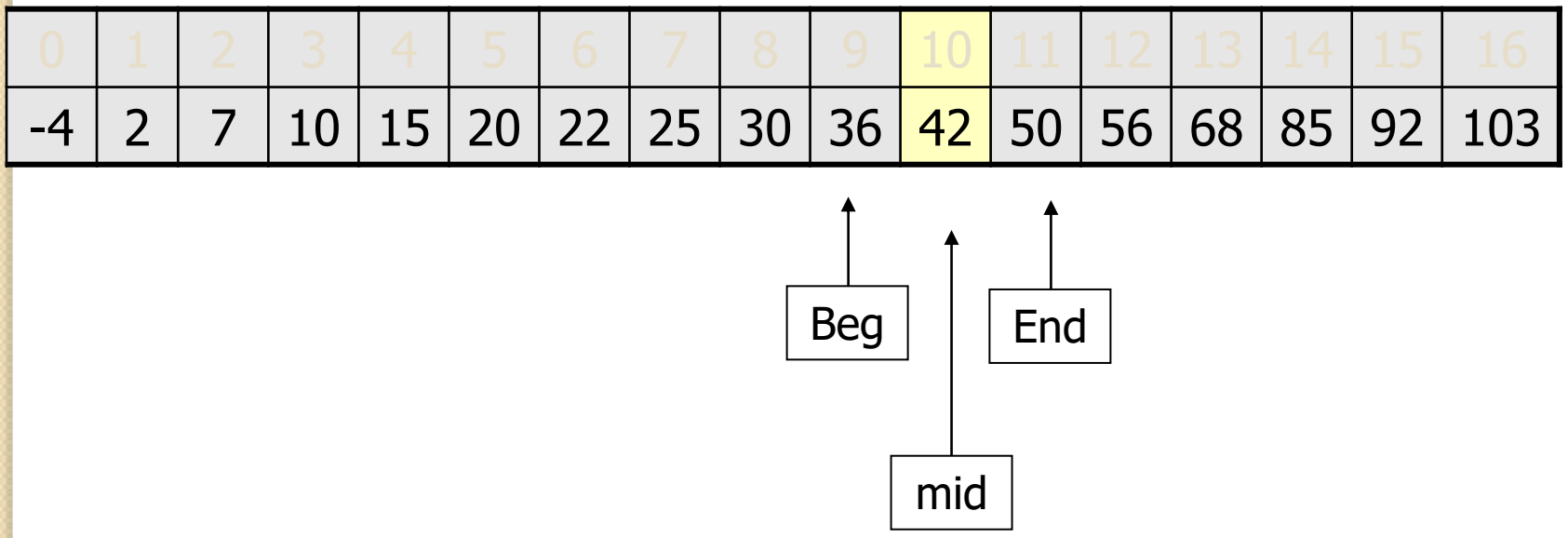
1. [Initialize segment variables.]
   Set BEG := LB, END := UB and MID = INT((BEG + END)/2).
2. Repeat Steps 3 and 4 while BEG ≤ END and DATA[MID] ≠ ITEM.
3. If ITEM < DATA[MID], then:
   Set END := MID − 1.
   Else:
   Set BEG := MID + 1.
   [End of If structure.]
4. Set MID := INT((BEG + END)/2).
   [End of Step 2 loop.]
5. If DATA[MID] = ITEM, then:
   Set LOC := MID.
   Else:
   Set LOC := NULL.
   [End of If structure.]
6. Exit.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

Beg mid End

Binary search

- Search 42



Summary

- Linear search is costly because the size of algorithm is proportional to the size of data
- Binary search divides the problem size by half in each iteration so it searches efficiently as compared to linear search.
- The Big O notation is Log N with base 2 because 2 is the dividing factor.

Assignment

- Apply Binary search for values:
  - 15
  - 55
  - 21

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

Assignment

Drive Big O notation for Binary search algo.