Problem Solving and Programming CSE1001

Prof. Tulasi Prasad Sariki

October 14, 2019

Searching
Sequential & Binary Search

Introduction to Searching

Problem

When the city planners developed your neighborhood, they accidentally numbered the houses wrong. As such, the addresses of the houses on your street are in a random order. How does the postman find your house using a linear search method?

Introduction to Searching

Pseudocode

```
READ street_door_numbers and door_number_searched
FOR i = 0 to length(street_door_numbers )
    IF street_door_numbers [i] == door_number_searched
    THEN
        give the post in the house
        break
END FOR
```

SEARCHING

- Searching is the algorithmic process of finding a particular item in a collection of items.
- **A search typically answers either True or False as to whether the item is present.**

Introduction to Searching

TYPES OF SEARCHING

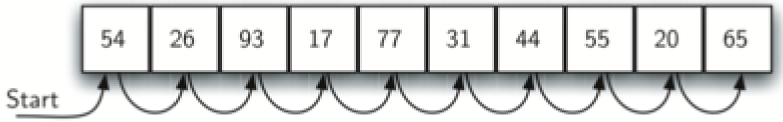
- SEQUENTIAL / LINEAR SEARCH
- BINARY SEARCH

SEQUENTIAL SEARCH

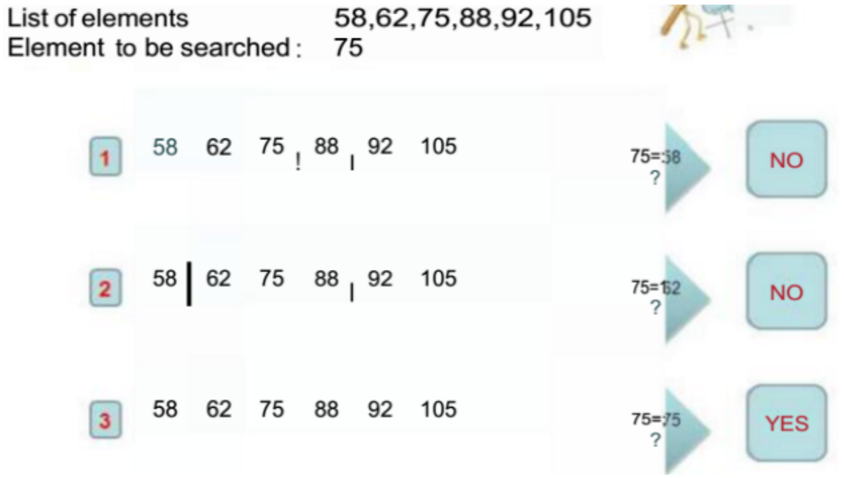
- When data items are stored in a collection such as a list, we say that they have a linear or sequential relationship.
- Each data item is stored in a position relative to the others.
- In Python lists, these relative positions are the index values of the individual items. Since these index values are ordered, it is possible for us to visit them in sequence.
- This process gives rise to our first searching technique, the sequential search

SEQUENTIAL SEARCH

- Starting at the first item in the list, we simply move from item to item, following the underlying sequential ordering until we either find what we are looking for or run out of items. If we run out of items, we have discovered that the item we were searching for was not present.



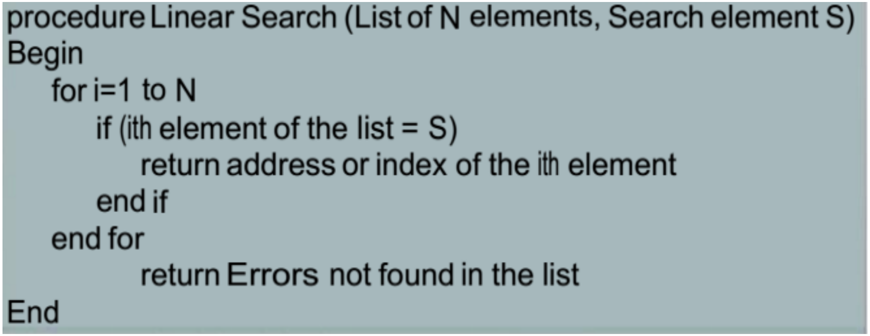
Example



List of elements 58,62,75,88,92,105

Element to be searched : 75

Linear Search Algorithm



```
procedure Linear Search (List of N elements, Search element S)
Begin
    for i=1 to N
        if (ith element of the list = S)
            return address or index of the ith element
        end if
    end for
            return Errors not found in the list
End
```

Python Implementation

- The function needs the list and the item we are looking for and returns a Boolean value as to whether it is present.
- The Boolean variable found is initialized to False and is assigned the value True if we discover the item in the list.

Python code - Sequential Search

```
def SeqSearch(alist,item):
    found = False
    for i in range(0,len(alist)):
        if item==alist[i]:
            found = True
    return found
testlist = [0,1,2,8,13,17,19,32,42]
print(SeqSearch(testlist,3))
print(SeqSearch(testlist,13))
```

Exercise-Sequential Search

Exercise - Sequential Search

You need a picture frame, so you walk down to the local photo store to examine their collection. They have all of their frames lined up against the wall. Apply the linear search algorithm to this problem, and describe how you would find the frame you wanted. Starting at the first frame, examine each frame along the wall (without skipping any) until you find the frame you want.

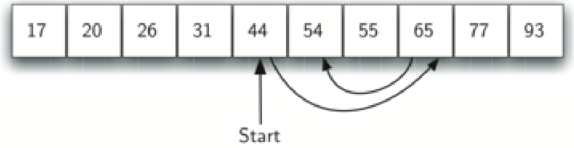
BINARY SEARCH

- In the sequential search, when we compare against the first item, there are at most n−1 more items to look through if the first item is not what we are looking for.
- Instead of searching the list in sequence; binary search will start by examining the middle item. If that item is the one we are searching for, we are done.
- If it is not the correct item, we can use the ordered nature of the list to eliminate half of the remaining items.

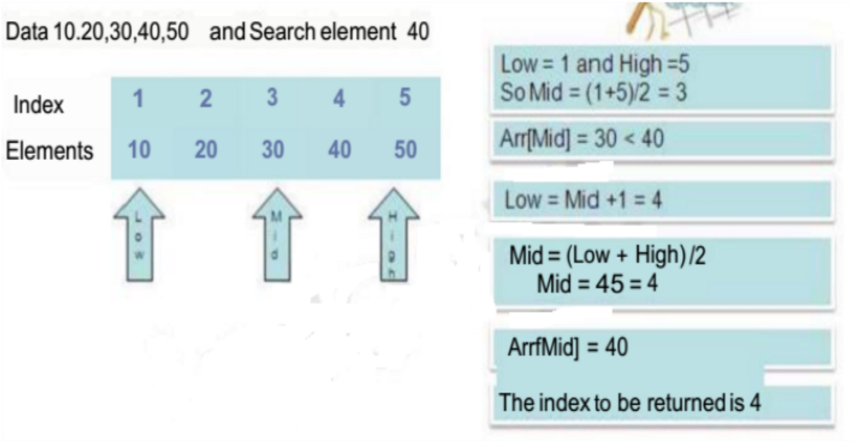
BINARY SEARCH

Algorithm can quickly find the value 54

- If the item we are searching for is greater than the middle item, we know that the entire lower half of the list as well as the middle item can be eliminated from further consideration.
- The item, if it is in the list, must be in the upper half. We can then repeat the process with the upper half. Start at the middle item and compare it against what we are looking for. Again, we either find it or split the list in half, therefore eliminating another large part of our possible search space



Binary Search Example



Data 10.20,30,40,50 and Search element 40

Index	1	2	3	4	5
Elements	10	20	30	40	50

Low = 1 and High =5
So Mid = (1+5)/2 = 3

Arr[Mid] = 30 < 40

Low = Mid +1 = 4

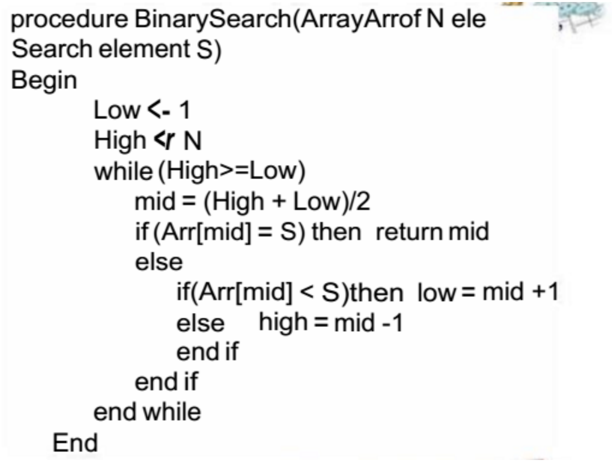
Mid = (Low + High)/2
Mid = 45 = 4

ArrfMid] = 40

The index to be returned is 4

Introduction to Searching

Binary Search Algorithm



```
procedure BinarySearch(ArrayArrof N ele
Search element S)
Begin
        Low <- 1
        High <r N
        while (High>=Low)
            mid = (High + Low)/2
            if (Arr[mid] = S) then  return mid
            else
                if(Arr[mid] < S)then  low = mid +1
                else      high = mid -1
                end if
            end if
        end while
    End
```

Introduction to Searching

Python code - Binary Search

```
def BinarySearch(alist,item):
    first=0
    last=len(alist)-1
    found=False
    while (first<=last and not found):
        midpoint = (first+last)//2
        if (alist[midpoint]==item):
            found = True
        else:
            if (item < alist[midpoint]):
                last=midpoint-1
            else:
                first=midpoint+1
    return found
testlist=[0,1,2,8,13,17,19,32,42]
print(BinarySearch(testlist,3))
print(BinarySearch(testlist,13))
```

Binary Search - Exercise

Binary Search

- Given a ordered list of student rank with the name of student. Your program will read the rank from the user and display the name of the student.
- An employee number is generated in ascending order whenever a new employee joins. Your program will read the employee number and display the dept of the employee

Binary Search - More Exercise

Binary Search

- The element being searched for is not in an array of 100 elements. What is the maximum number of comparisons needed in a sequential search to determine that the element is not there if the elements are: (a) completely unsorted? (b) sorted in ascending order? (c) sorted in descending order?
- Consider the following array of sorted integers: 10, 15, 25, 30, 33, 34, 46, 55, 78, 84, 96, 99 Using binary search algorithm, search for 23. Show the sequence of array elements that are compared, and for each comparison, indicate the values of low and high

Thank you