Searching Techniques:

a) Sequential Search
b) Binary Search

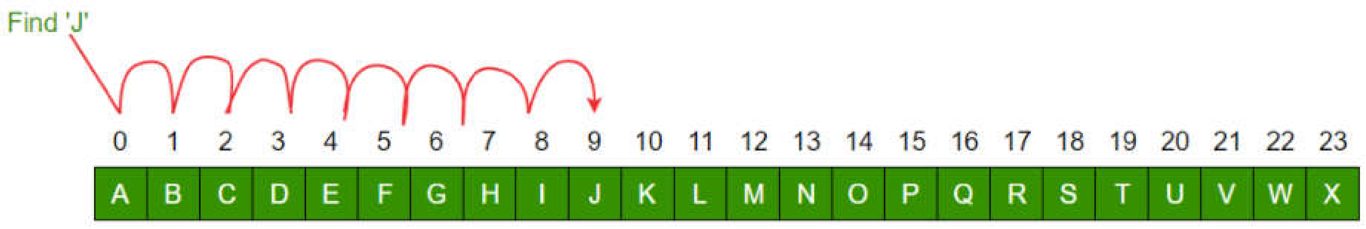
Applications of Searching:

- Searching for a keyword, a value, or a specific piece of data (information) is the basis of many computing applications,
    - whether it's looking up a bank account balance,
    - using an internet search engine,
    - or searching for a file on your laptop.
- Computers deal with a lot of information so we need efficient algorithms for searching.

Sequential / Linear Search

- A linear search scans one item at a time, without jumping to any item.
- Time taken to search elements keep increasing as the number of elements are increased.

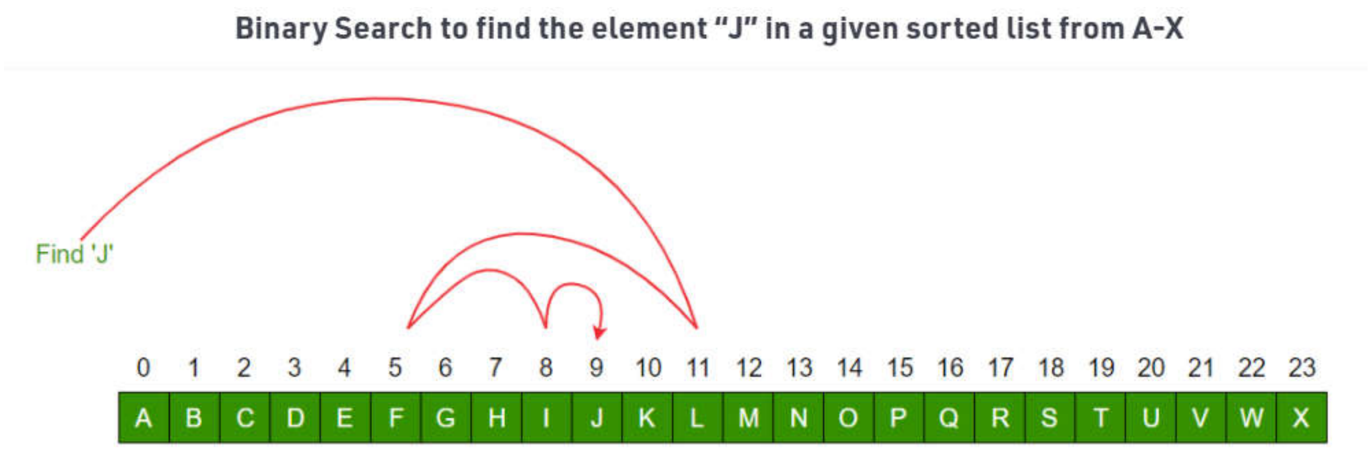
Linear Search to find the element "J" in a given sorted list from A-X



Binary search:

- It is called 'binary' search because each time you look at a value in the list you divide the list into 2 parts, one is discarded and the other is kept.
- The word "binary" here just means something that has two parts, such as a binary star system.
- Binary search shouldn't be confused with binary numbers.
- A binary search cut down our search to half as soon as you find middle of a sorted list.

1. The middle element is looked to check if it is greater than or less than the value to be searched.
2. Accordingly, search is done to either half of the given list


Binary Search to find the element "J" in a given sorted list from A-X
Find 'J'
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
A B C D E F G H I J K L M N O P Q R S T U V W X

Important Differences

- Input data needs to be sorted in Binary Search and not in Linear Search
- Linear search does the sequential access whereas Binary search access data randomly.

Activities:

1) **Write a python program to solve the following requirements.**

   a. Read **unsorted** 'n' integers using input () and store them in a list (say **list_numbers**). Minimum size of the **list_numbers** should be 10.

   b. Print the **list_numbers** on the screen.

   c. Read a number (say **number_to_be_searched**) using input().

   d. Perform 'linear search' to find whether the **number_to_be_searched** is present or not in **list_numbers.**

   e. Print the suitable result

      i. **The searched element (print the actual value) is PRESENT in list_number.**

         (or)

      ii. **The searched element (print the actual value) is NOT PRESENT in list_number.**

   f. Count the number of comparisons and print it.

   g. Run the same program for different **number_to_be_searched** and record the outputs.

2) **Write a python program to solve the following requirements.**

   a. Read **sorted** 'n' integers using input () and store them in a list (say **list_numbers**). Minimum size of the **list_numbers** should be 10.

   b. Print the **list_numbers** on the screen.

   c. Read a number (say **number_to_be_searched**) using input().

   d. Perform 'binary search' to find whether the **number_to_be_searched** is present or not in **list_numbers.**

   e. Print the suitable result

   i. **The searched element (print the actual value) is PRESENT in list_number.**

      (or)

   ii. **The searched element (print the actual value) is NOT PRESENT in list_number.**

f. Count the number of comparisons and print it.

g. Run the same program for different **number_to_be_searched** and record the outputs.

Attention:

- Prepare a single PDF file for the above activities with following contents.
    - Activity (i.e. the Question - just copy + paste)
    - Code (**complete screen shot including your face captured by camera, otherwise, your programs will not be evaluated**)
    - Code (only the program, which I can directly copy and execute)
    - Output screen shots **(complete screen including your face captured by camera; don't crop only the output, take a screenshot, which shows everything; otherwise, your programs will not be evaluated)**
    - Arrange the above contents in order under each question.
    - Use Landscape orientation, when you create the file.