SEARCHING ALGORITHMS

1.JUMP SEARCH ALGORITHM

How does Jump Search work?

- In jump search, we traverse the **sorted array** and jump ahead with the specific **block size**. We will determine the block size to jump by getting the **square root** of the **size of an array**.
- i.e $\sqrt{n}$ where **n** is the **length** of the sorted array.
- When we find an element greater than the search element, a **linear search** is performed in the **current block**. The element, if present in the array, will be found in the block.
- As jump search uses **blocks** in the search operations, it is also known as **block search algorithm.**

Algorithm Steps

Let’s understand the sequence in steps:

- **Sort** the array if not sorted already.
- Calculate **block size** to jump. Generally, it is the **square root** of **array length**.
- **Traverse** the sorted array and jump elements based on calculated block size.
- Perform **linear search** when the **current value** is **greater** than the given value.
- Return the index of the element once a match is found.

Pictorial Representation of Jump Search with an Example

Let us trace the above algorithm using an example:

Consider the following inputs:

A[] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 55, 77, 89, 101, 201, 256, 780}

item = 77

Pictorial Representation of Jump Search with an Example

Step 1: **m** = $\sqrt{n}$ = **4** (**Block Size**)

Step 2: Compare **A[0]** with **item**. Since **A[0]** != **item** and **A[0]**< **item**, skip to the **next block.**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	2	3	5	8	13	21	55	77	89	101	201	256	780

Figure 2: Comparing A[0] and item

Pictorial Representation of Jump Search with an Example

Step 3: Compare **A[3]** with item. Since **A[3]** != **item** and **A[3]** < **item**, skip to the **next block.**

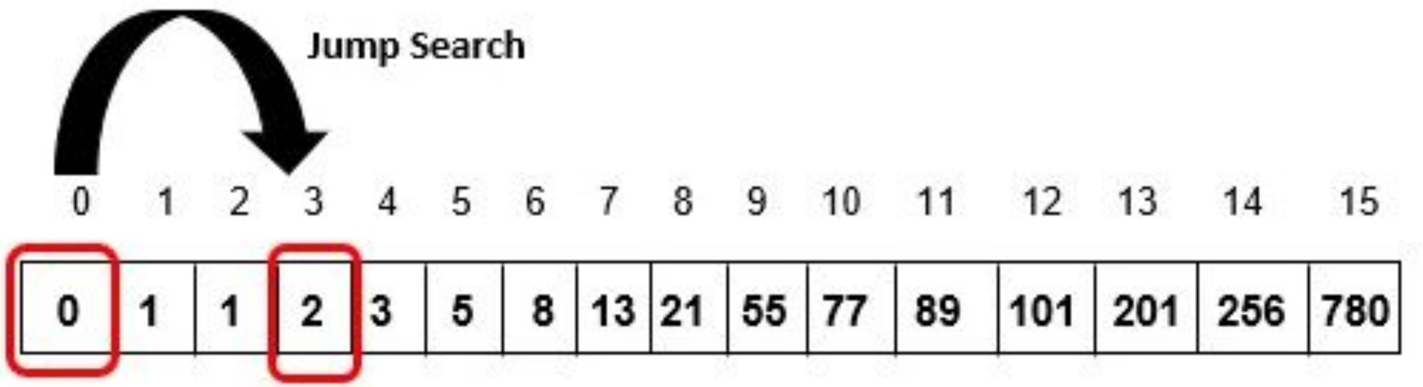

Figure 3: Comparing A[3] and item

Pictorial Representation of Jump Search with an Example

Step 4: Compare **A[6]** with item. Since **A[6]** != **item** and **A[6]** < **item**, skip to the **next block.**

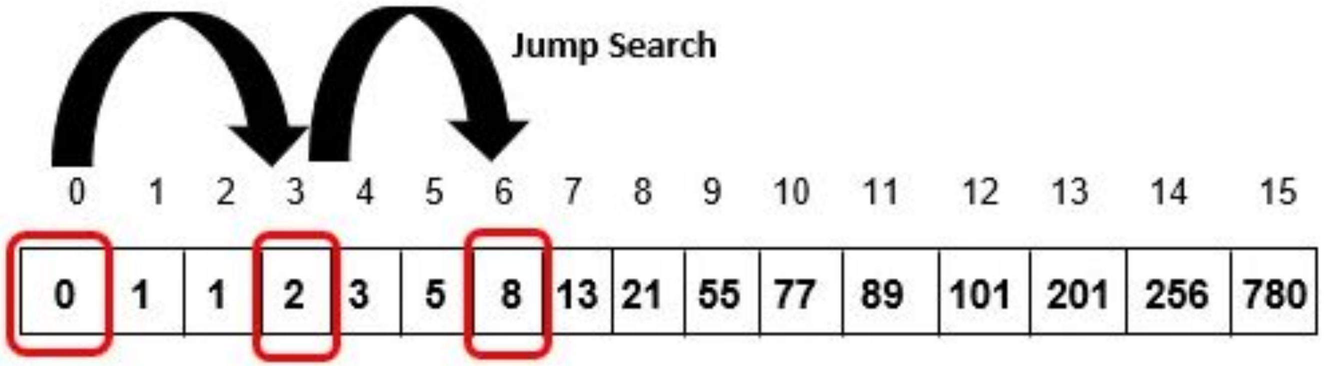

Figure 4: Comparing A[6] and item

Pictorial Representation of Jump Search with an Example

Step 5: Compare **A[9]** with item. Since **A[9]** != **item** and **A[9]** < **item**, skip to the **next block.**

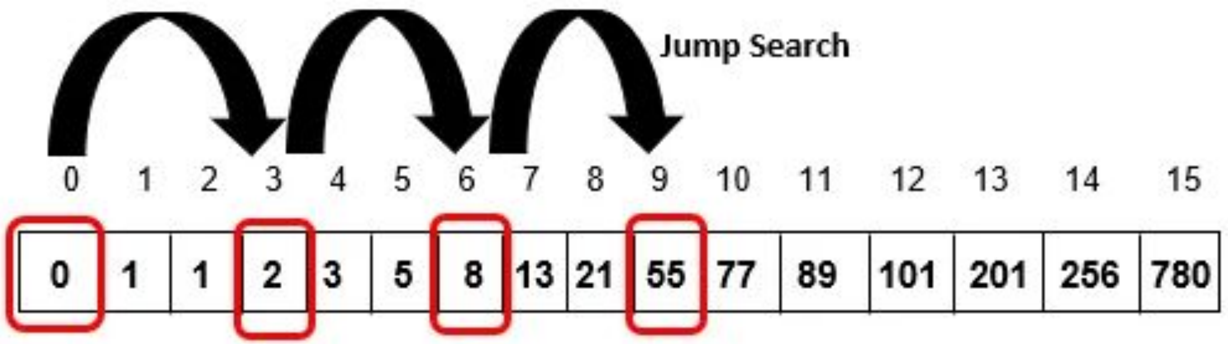

Figure 5: Comparing A[9] and item

Pictorial Representation of Jump Search with an Example

Step 6: Compare **A[12]** with **item**. Since **A[12]** **!= item** and **A[12]** **> item**, skip to **A[9]** (beginning of the current block) and perform a **linear search**.

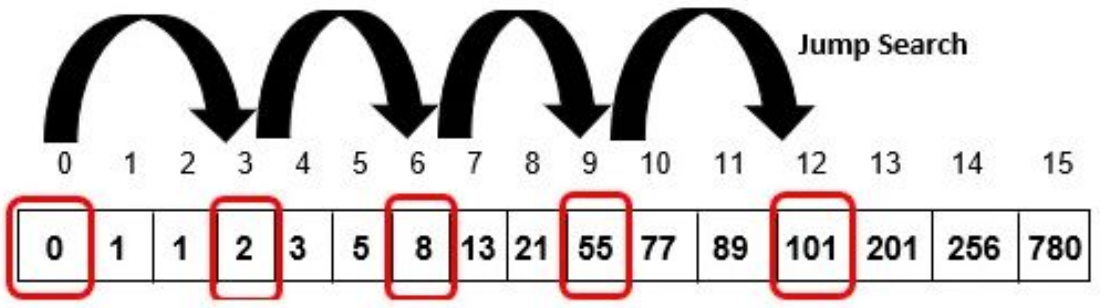

Figure 6: Comparing A[12] and item

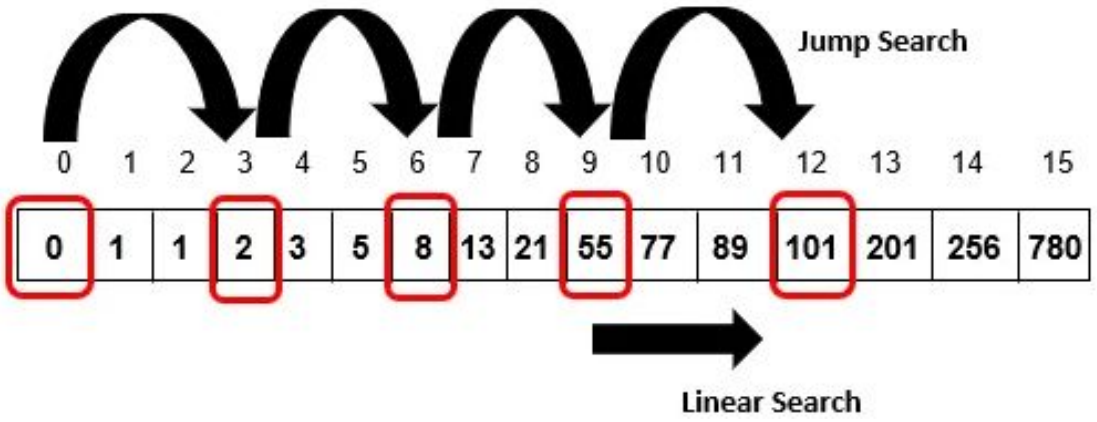

Figure 7: Comparing A[9] and item

Pictorial Representation of Jump Search with an Example

- Compare **A[9]** with **item**. Since **A[9]** != **item**, scan the next element
- Compare **A[10]** with item. Since **A[10]** == **item**, index **10** is printed as the **valid location** and the algorithm will terminate.

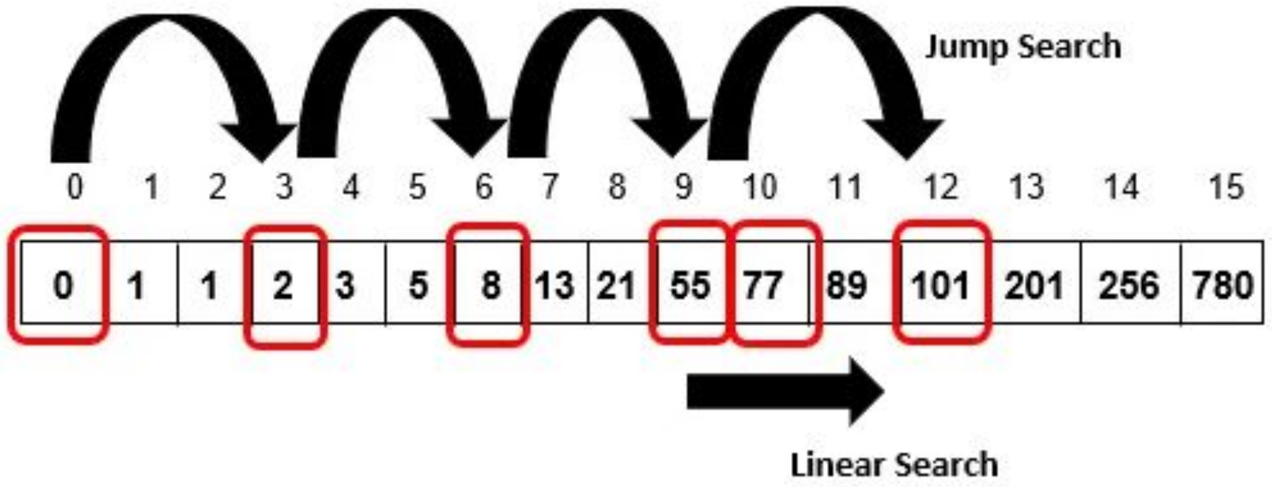

Figure 8: Comparing A[10] and item (Linear Search)

Algorithm

jumpSearch(**array**, **size**, **key**)

Input: An **sorted array**, **size of the array** and the **search key**

Output – **location** of the key (if found), otherwise wrong location.

```
Begin
  blockSize := √size
  start := 0
  end := blockSize
  while array[end] <= key AND end < size do
    start := end
    end := end + blockSize
    if end > size – 1 then
      end := size
  done
  for i := start to end -1 do
    if array[i] = key then
      return i
  done
  return invalid location
End
```

QUESTION 01

Input:

A sorted list of data:

10 13 15 26 28 50 56 88 94 127 159 356 480 567 689 699 780 850 956 995

The search key **356**

Output:

Item found at location: **11**

2.INTEPOLATION SEARCH ALGORITHM

NOTE

"In a typical English dictionary, data is quite uniformly distributed, **For example**, all the words **starting** with **a** or **b** will be in the **beginning** of the dictionary, and all the words starting with **y** and **z** will be in the very **end** of the dictionary. On the other hand, words starting with **m** and **n** will probably lie somewhere in the middle.
angle, binary, ……………………, yesterday, zoo"

https://www.scaler.com/topics/data-structures/interpolation-search-algorithm/

How does Interpolation Search work?

- Interpolation search is an improvement over **binary search**.
- Binary Search always checks the value at **middle index**. But, interpolation search may check at **different locations** based on the value of **element** being searched.
- For interpolation search to work efficiently the array elements/data should be **sorted** and **uniformly distributed**.

$$\boxed{pos = l + \frac{(x - arr[l])}{(arr[h] - arr[l])} \times (h - l)}$$

pos - index of the searched item
l - lowest point
h - highest point
x - search element

QUESTION 02

Input:

A sorted list of data:

10, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 33, 35, 42, 47

The search key **18**

Output:

Item found at location: **4**