INTERPOLATION SEARCH

ALGORITHMIC PROBLEM SOLVING

17ECSE309

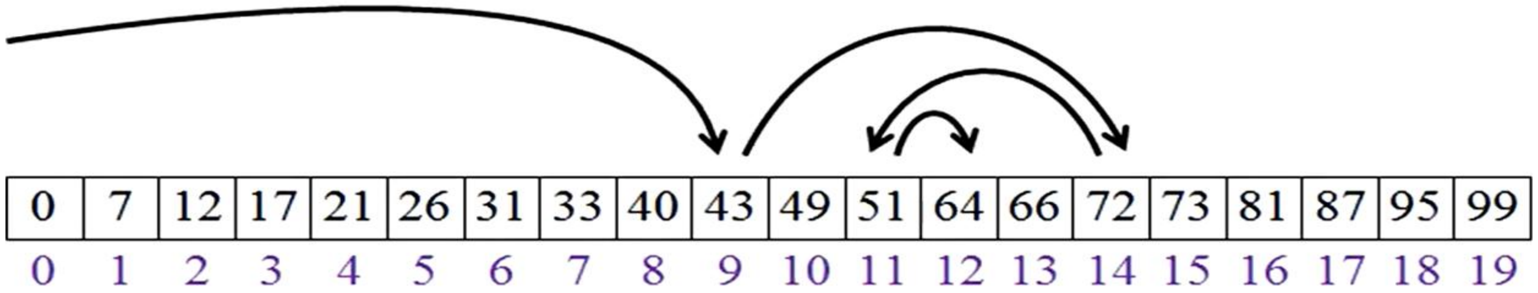
Prajakta Desai
01FE15BCS133

What is Interpolation Search?

- It is a variation of binary search that uses additional information of the key to be searched.
- Here, the position of the key is determined based on the minimum and maximum number in the array.
- For efficient working of this method, the array should be sorted.
- **Performance :**
    - Average case = O(log (log n))
    - Worst case = O(n)
- **Applications:**

  Used in book-based searching, such as telephone metadata.

Binary Search vs Interpolation Search



0	7	12	17	21	26	31	33	40	43	49	51	64	66	72	73	81	87	95	99
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

- Suppose the key to be searched is 64.
- In binary search, the mid value is first calculated.
- If the key is lesser than mid value, then left sub-array is considered else right sub-array is considered.
- The above two steps are repeated until the key is found.
- This gives the time complexity of O(log n)

Binary Search vs Interpolation Search

- Interpolation Search makes use of the probe position formula to search a key.
- **Probe Position Formula:**

distance = target – array[min]

value_range = array[max]– array[min]

fraction = distance / value_range

index_range = max – min

pos = min + fraction * index_range

Binary Search vs Interpolation Search

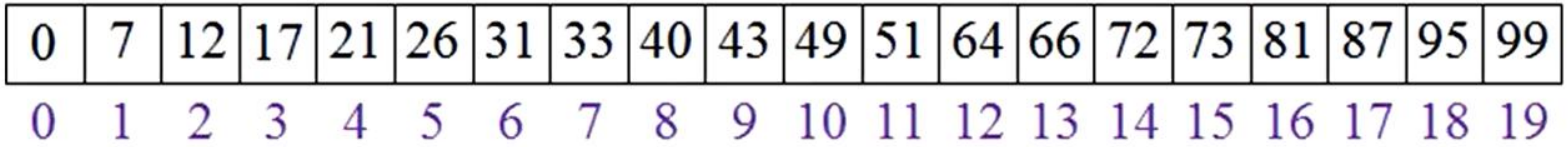
- **Steps:**

1. In a loop, calculate the value of ‘pos’ using probe position formula.
2. If it is a match, return the index of the item and exit.
3. If the item is less than array[pos], calucate probe position of left sub-array. Otherwise, calculate position for right sub-array.
4. Repeat until match is found or sub-array reduces to 0.

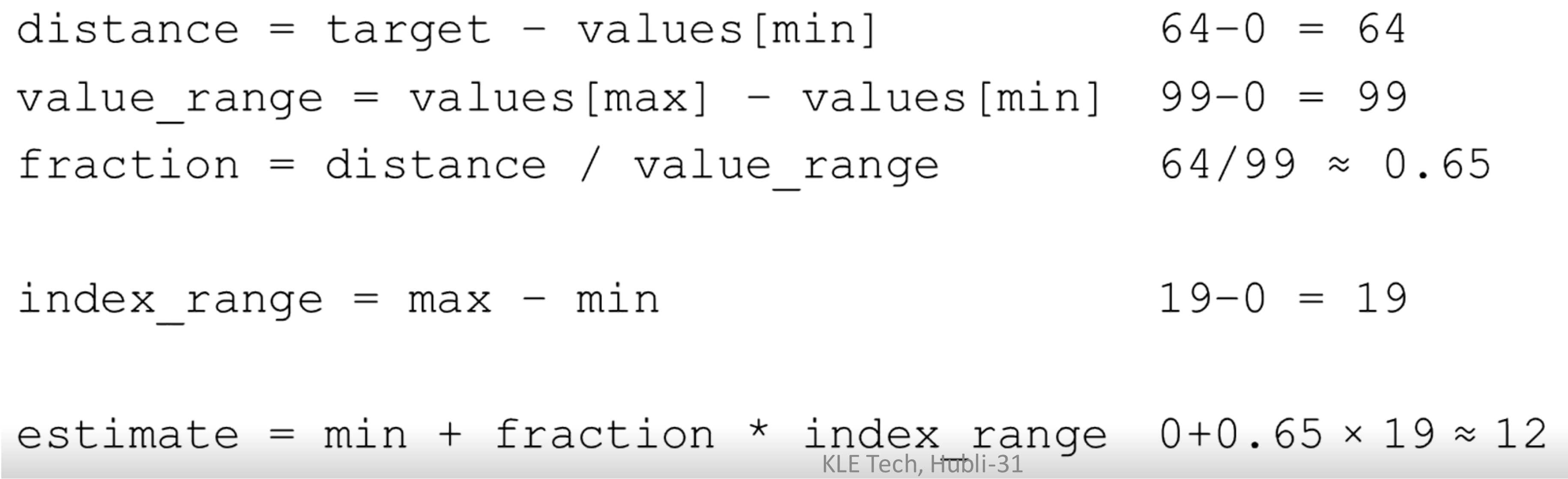
- Here, if the key is close to the end of the array element, search begins near the end. Hence, it is a better searching method compared to binary search.

Binary Search vs Interpolation Search

- **Example:** To search value 64



0	7	12	17	21	26	31	33	40	43	49	51	64	66	72	73	81	87	95	99
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

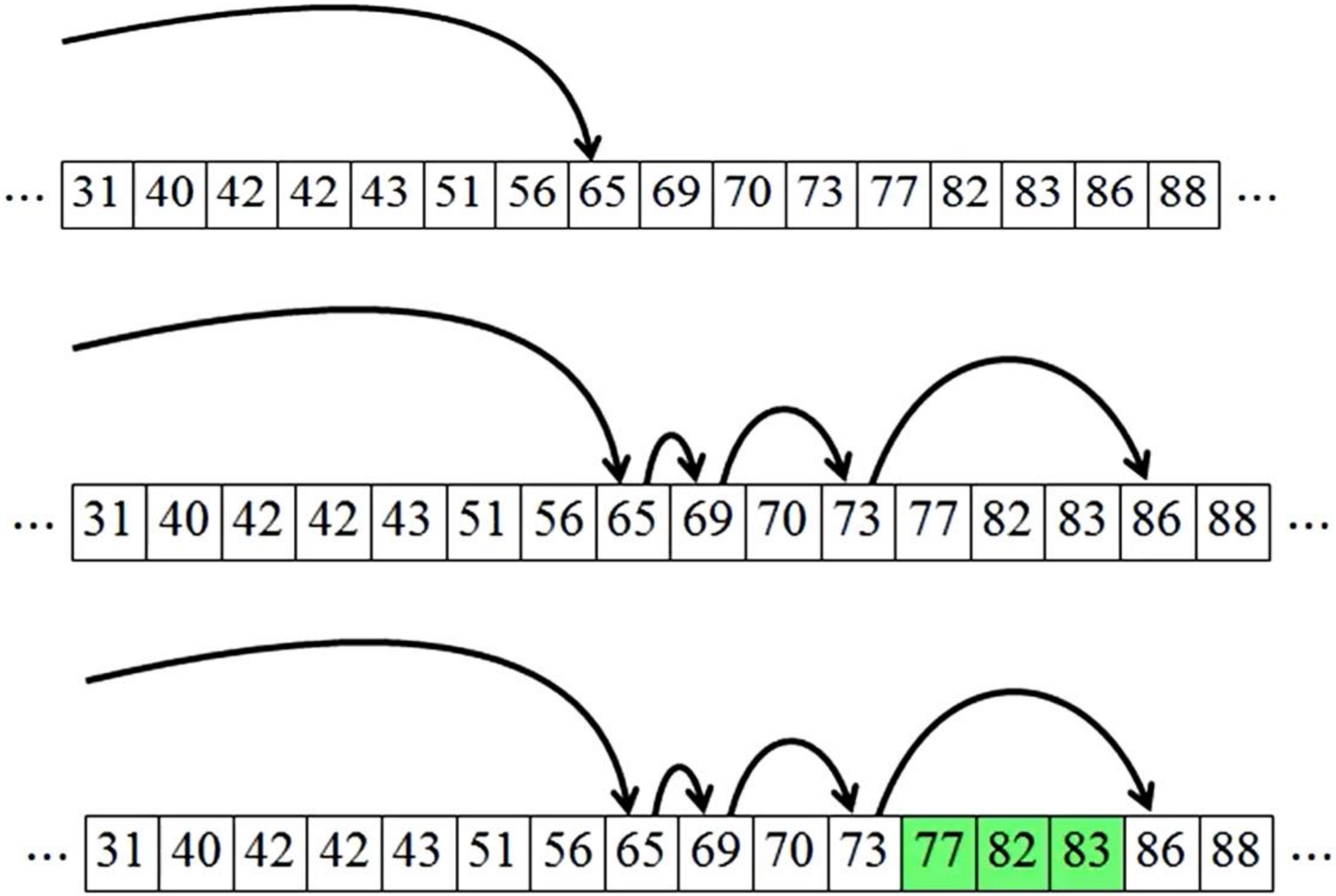


```
distance = target - values[min]                64-0 = 64
value_range = values[max] - values[min]        99-0 = 99
fraction = distance / value_range              64/99 ≈ 0.65

index_range = max - min                        19-0 = 19

estimate = min + fraction * index_range        0+0.65 × 19 ≈ 12
```

Varied Interpolation Search



… 31 40 42 42 43 51 56 65 69 70 73 77 82 83 86 88 …

… 31 40 42 42 43 51 56 65 69 70 73 77 82 83 86 88 …

… 31 40 42 42 43 51 56 65 69 70 73 77 82 83 86 88 …

- Suppose, we want to search for the value 83.
- First, the position is calculated using probe position formula, giving us arr[pos] = 65.
- Since, 65 is less than 83, we move in the right sub-array in small increments (1,2,4,..)
- Incrementing by 1, we get 69 which is still less than 83. Hence increment by 2.
- We get 73, less than 83. Increment by 4.
- We get 86, greater than 83.
- Consider the array ranging between 73 and 86, and perform binary search

References

1. https://en.wikipedia.org/wiki/Interpolation_search
2. https://www.geeksforgeeks.org/interpolation-search/
3. https://www.quora.com/What-is-interpolation-search-in-data-structures
4. https://www.quora.com/What-is-interpolation-search-and-what-is-it-used-for
5. https://www.youtube.com/watch?v=aODWxapipRk