Introduction to Computing

Sorting, Searching

Malay Bhattacharyya

Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

November, 2025

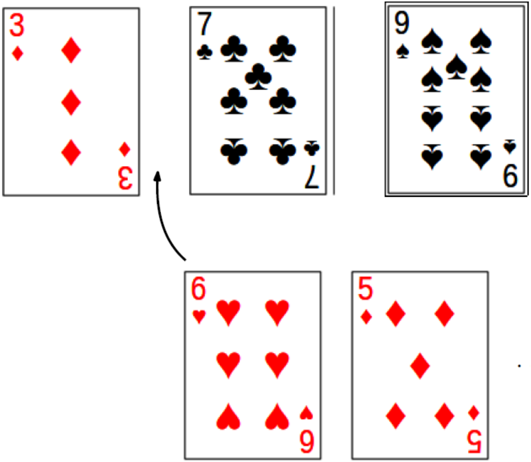
1 Sorting

2 Searching

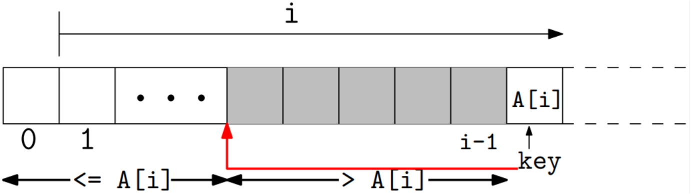
Bubble sort

```
LIST = [2, 6, 4, 8, 1]
n = len(LIST)
for i in range(n-1):
    for j in range(n-i-1):
        if LIST[j+1] < LIST[j]:
            LIST[j], LIST[j+1] = LIST[j+1], LIST[j]
```

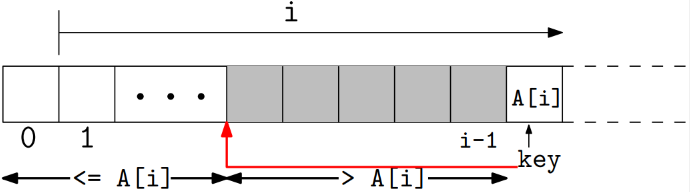
Insertion sort



Insertion sort

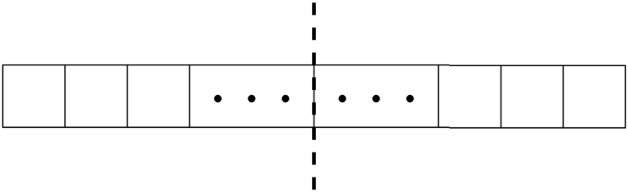


Insertion sort

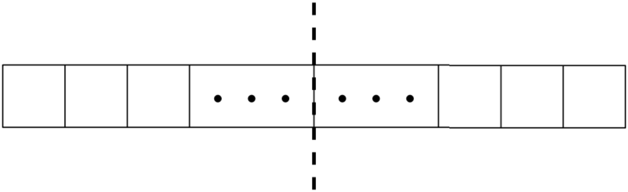



```
LIST = [2, 6, 4, 8, 1]
n = len(LIST)
for i in range(1, n):
    key = LIST[i]
    # Find the right place for LIST[i] in LIST[0 ... i-1]
    j = i - 1
    while j >= 0 and LIST[j] > key:
        LIST[j+1] = LIST[j]
        j = j - 1
    LIST[j+1] = key
```

Merge sort



Merge sort

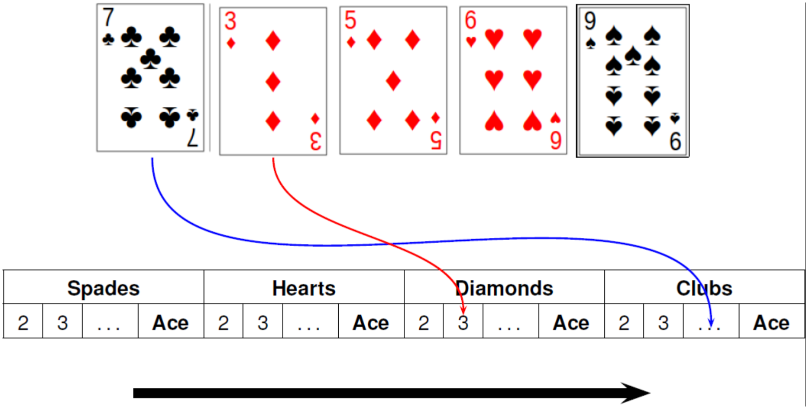


```
def msort(LIST, beginning, end):
     if beginning < end:
        middle = (beginning + end) / 2
        msort(LIST, beginning, middle)
        msort(LIST, middle+1, end)
        merge(LIST, beginning, middle, end)
```

Merge sort

```
def merge(LIST, b, m, e):
    i, j, k = b, m + 1, 0
    while i <= m and j <= e:
        if LIST[i] < LIST[j]:
            AUXLIST[k] = LIST[i]
            k, i = k+1, i+1
        else if A[i] > A[j]:
            AUXLIST[k] = LIST[j]
            k, j = k+1, j+1
        else:
            AUXLIST[k] = LIST[i]
            k, i = k+1, i+1
            AUXLIST[k] = LIST[j]
            k, j = k+1, j+1
     while i <= m:
        AUXLIST[k] = LIST[i]
        k, i = k+1, i+1
     while j <= e:
        AUXLIST[k] = LIST[j]
```

Bucket sort



Bucket sort

```
# Allocate space for array A
for i in range(n):
    A[i] = i
```

Timsort

Timsort is a highly optimized and stable version of merge sort that effectively combines insertion sort.

Timsort was implemented by Tim Peters in 2002 for use in the Python programming language.

- `https://en.wikipedia.org/wiki/Timsort`
- `https://realpython.com/sorting-algorithms-python/#pythons-built-in-sorting-algorithm`

Timsort

Input: Unsorted collection
Output: Sorted collection

```
// Calculate run length
runLength := calculateRunLength(array);
// Perform insertion sort on each run
for start ← 0 to array.length by runLength do
    end := min(array.length − 1, start + runLength − 1);
    insertionSort(array, start, end);
end
// Recursively merge adjacent runs
mergeSize := runLength;
while mergeSize < array.length do
    for left ← 0 to array.length by size ∗ 2 do
        mid := left + size − 1;
        right := min(array.length − 1, left + (2 ∗ size) − 1);
        if mid < right then
            merge(array, left, mid, right);
        end
    end
    mergeSize := mergeSize ∗ 2;
end
```

The searching problem

Given a data structure S over a domain of data items D, verify (search) whether a given data item s belongs to D or not. If it belongs, then return the position (index) of s in D with respect to its organization in S.

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	$\downarrow$					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	$\downarrow$					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		$\downarrow$				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	$\downarrow$					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		$\downarrow$				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$			$\downarrow$			
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	$\downarrow$					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		$\downarrow$				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$			$\downarrow$			
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$				$\downarrow$		
Whether $LIST[i] = s$	30	10	20	40	50	TRUE

Return $i = 3$.

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

			$\downarrow$			
$pre\_i = 0, i = b$						
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

			$\downarrow$			
$pre\_i = 0, i = b$						
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

					$\downarrow$	
$pre\_i = i, i = i + b$						
Whether $LIST[i] < s$	10	20	30	40	50	FALSE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre\_i = 0, i = b$			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$pre\_i = i, i = i + b$					$\downarrow$	
Whether $LIST[i] < s$	10	20	30	40	50	FALSE
$pre\_i = pre\_i + 1$				$\downarrow$		
Whether $LIST[pre\_i] = s$	10	20	30	40	50	TRUE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre\_i = 0, i = b$			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$pre\_i = i, i = i + b$					$\downarrow$	
Whether $LIST[i] < s$	10	20	30	40	50	FALSE
$pre\_i = pre\_i + 1$				$\downarrow$		
Whether $LIST[pre\_i] = s$	10	20	30	40	50	TRUE

Note: The block size in jump search is taken as $b = \sqrt{n}$, where n denotes the size of the list.

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$l = i+1, i = l+(r-l)/2 = 3$				$\downarrow$		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Comparing linear/jump/binary search

	Linear search	**Jump search**	**Binary search**
Requirements	Nothing	Ordered list	Ordered list
Backward scan required	No	Once	Multiple times
Time complexity (best case)	$O(1)$	$O(b)$	$O(1)$
Time complexity (average case)	$O(n)$	$O(\sqrt{n})$	$O(\log n)$
Time complexity (worst case)	$O(n)$	$O(\sqrt{n})$	$O(\log n)$

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$i = 0 + (40 - 10) * (4 - 0)/(50 - 10) = 3$				$\downarrow$		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

				↓		
$i = 0 + (40 - 10) * (4 - 0)/(50 - 10) = 3$						
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Note: The interpolation search is efficient over binary search for the ordered lists having uniformly distributed data items.

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = Fibonacci(f-2) = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = Fibonacci(f-2) = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = Fibonacci(f-2) = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$f = 4 - 2 + 1 = 3, i = i + Fibonacci(f-2) = 3$				$\downarrow$		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = Fibonacci(f-2) = 2$			$\downarrow$			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			$\downarrow$			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$f = 4 - 2 + 1 = 3, i = i + Fibonacci(f-2) = 3$				$\downarrow$		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Note: Fibonacci search is a suitable replacement of binary search where the list is unbounded. Unlike binary search, it does not use the costly division operation (though avoidable with bitwise shift).