Sorting

- Arranging data in some given order

Sorting Algorithms/Techniques

- Insertion Sort
- Selection Sort
- Merge Sort
- Bubble Sort
- Quick Sort
- Heap Sort
- Radix Sort

Numeric Data 8, 4, 2, 6

Increasing Order 2, 4, 6, 8

Decreasing Order 8, 6, 4, 2

Character Data f, d, a, c

Alphabetically a, c, d, f

Selection Sort

Sort the following numbers using Selection Sort

77, 33, 44, 11, 88, 22, 66, 55

Pass	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
K=1, LOC=4	77	33	44	11	88	22	66	55

Selection Sort

Sort the following numbers using Selection Sort

77, 33, 44, 11, 88, 22, 66, 55

Pass	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
K=1, Loc=4	(77)	33	44	(11)	88	22	66	55
K=2, Loc=	11	(33)	44	77	88	(22)	66	55

Selection Sort

Sort the following numbers using Selection Sort

77, 33, 44, 11, 88, 22, 66, 55

Pass	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
K=1, Loc=4	(77)	33	44	(11)	88	22	66	55
K=2, Loc=6	11	(33)	44	77	88	(22)	66	55
K=3, Loc=6	11	22	(44)	77	88	(33)	66	55

Selection Sort

Sort the following numbers using Selection Sort

77, 33, 44, 11, 88, 22, 66, 55 ✓

Pass	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
K=1, LOC=4	(77)	33	44	(11)	88	22	66	55
K=2, LOC=6	11	(33)	44	77	88	(22)	66	55
K=3, LOC=6	11	22	(44)	77	88	(33)	66	55
K=4, LOC=6	11	22	33	(77)	88	(44)	66	55
K=5, LOC=8	11	22	33	44	(88)	77	66	(55)
K=6, LOC=7	11	22	33	44	55	(77)	(66)	88
K=7, LOC=7	11	22	33	44	55	66	(77)	88
Sorted	11	22	33	44	55	66	77	88

✓

Bubble Sort

Sort the following numbers using Bubble Sort

32, 51, 27, 85, 66, 23, 13, 57

Pass 1

32	51	27	85	66	23	13	57
32	27	51	85	66	23	13	57
32	27	51	66	85	23	13	57
32	27	51	66	23	85	13	57
32	27	51	66	23	13	85	57
32	27	51	66	23	13	57	85

Largest Number

Pass 2

32	27	51	66	23	13	57	85
27	32	51	66	23	13	57	85
27	32	51	23	66	13	57	85
27	32	51	23	13	66	57	85
27	32	51	23	13	57	66	85

Second Largest Number

Bubble Sort

Sort the following numbers using Bubble Sort

32, 51, 27, 85, 66, 23, 13, 57

Pass 3

27	32	51	23	13	57	66	85
27	32	23	51	13	57	66	85
27	32	23	13	51	57	66	85 ✓

Third Largest Number

Pass 4

27	32	23	13	51	57	66	85
27	23	32	13	51	57	66	85
27	23	13	32	51	57	66	85 ✓

Fourth Largest Number

Pass 5

27	23	13	32	51	57	66	85
23	27	13	32	51	57	66	85
23	13	27	32	51	57	66	85

Bubble Sort

Sort the following numbers using Bubble Sort

32, 51, 27, 85, 66, 23, 13, 57 ✓

Pass 6

✓

Pass 7

✓ Sorted

Insertion Sort

Sort the following numbers using Insertion Sort

77, 33, 44, 11, 88, 22, 66, 55 ✓

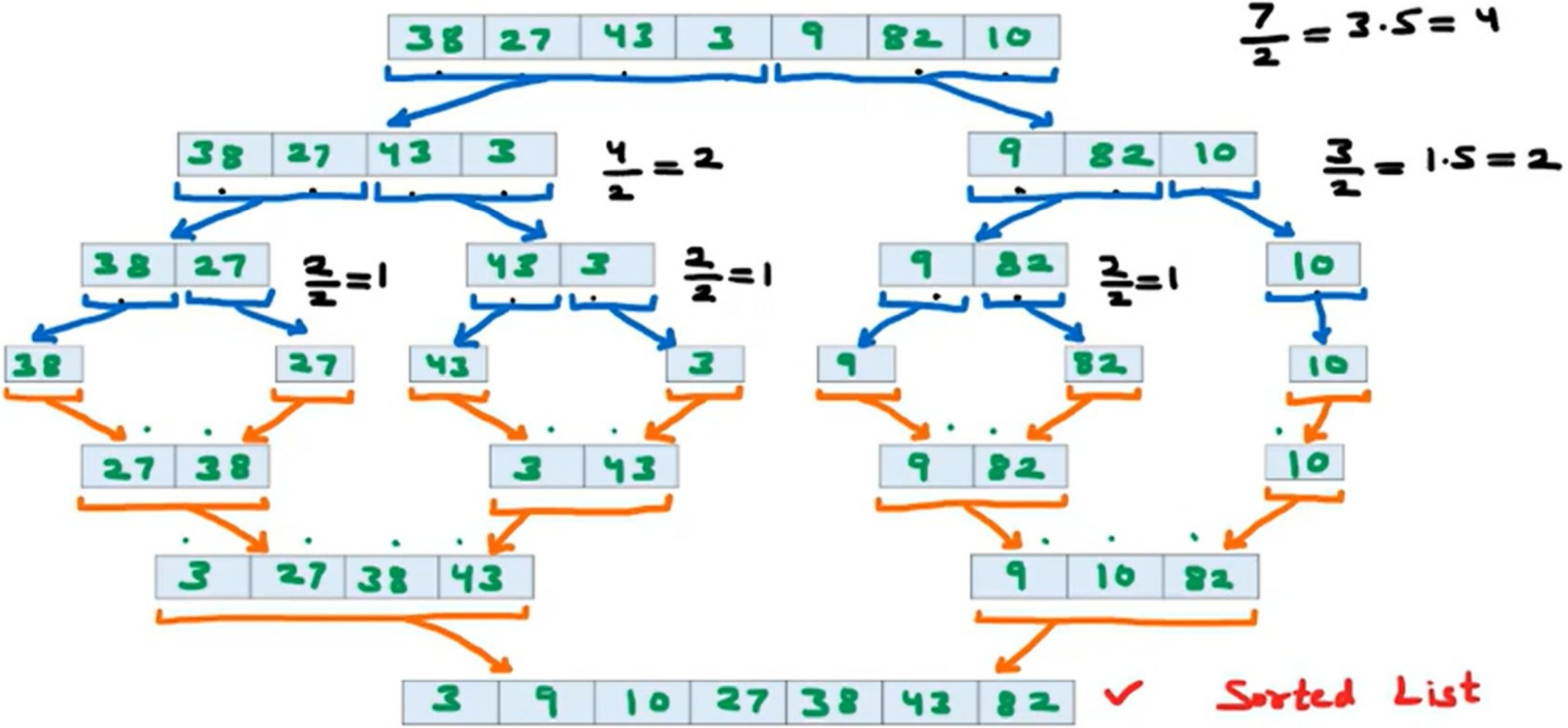
Pass	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
K=1	-∞	77	33	44	11	88	22	66	55
K=2	-∞	77	33	44	11	88	22	66	55
K=3	-∞	33	77	44	11	88	22	66	55
K=4	-∞	33	44	77	11	88	22	66	55
K=5	-∞	11	33	44	77	88	22	66	55
K=6	-∞	11	33	44	77	88	22	66	55
K=7	-∞	11	22	33	44	77	88	66	55
K=8	-∞	11	22	33	44	66	77	88	55
Sorted	-∞	11	22	33	44	55	66	77	88

✓

Merge Sort

Sort the following numbers using Merge Sort

38, 27, 43, 3, 9, 82, 10 ✓

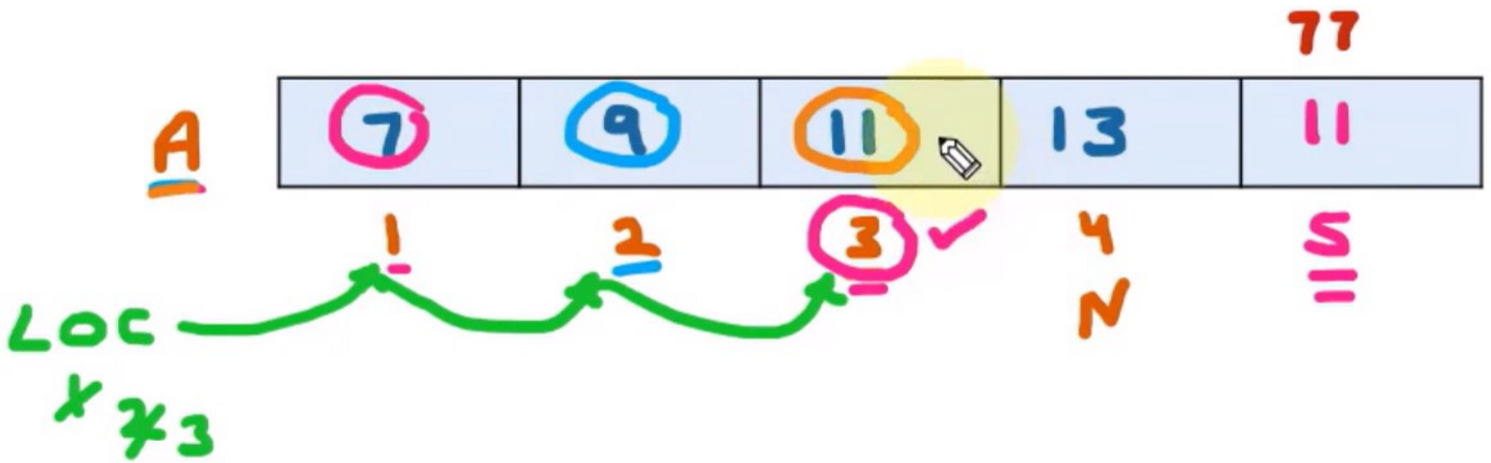


ALGORITHM: LINEAR SEARCH (A, N, ITEM, LOC)

A is linear array with N elements. Algorithm finds location LOC of ITEM in A, or sets LOC := 0 if search is unsuccessful.

1. Set A[N + 1] := ITEM.
2. Set LOC := 1.
3. Repeat while A[LOC] != ITEM:
   Set LOC := LOC + 1.
4. If LOC = N + 1, then Set LOC := 0.
5. Exit

Item



Binary Search

Search Item 23 from following Sorted Data using Binary Search

2, 5, 8, 12, 16, 23, 38, 56, 72, 91

$$MID = Int\left(\frac{LOW + HIGH}{2}\right)$$

If Item == A[MID] ✓

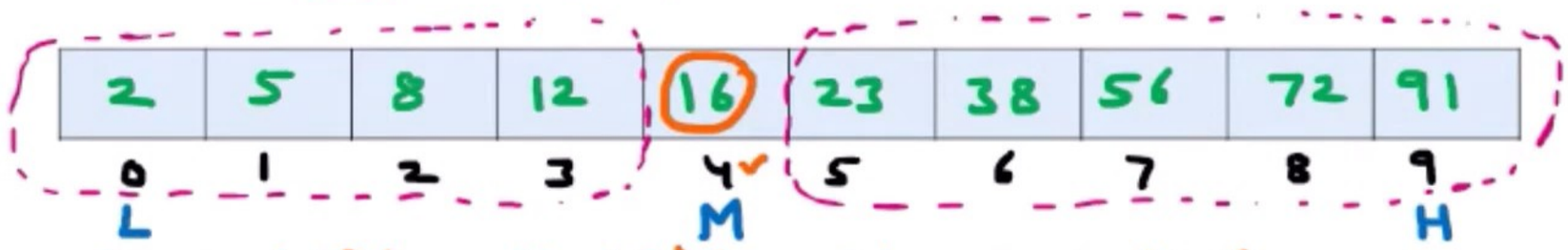
Element Found ✗

Else If Item < A[MID] ✓

HIGH = MID - 1 ✗

Else

LOW = MID + 1 ✓

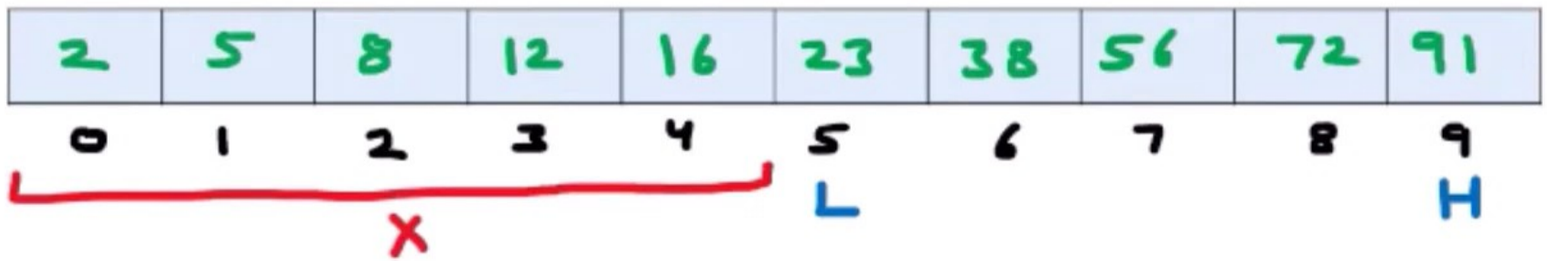


M = (L + H)/2 = (0 + 9)/2 = 9/2 = 4.5 = 4 ✓

23 == 16 ? X

23 < 16 ? X

L = M + 1 = 4 + 1 = 5 ✓



Binary Search

Search Item 23 from following Sorted Data using Binary Search

2, 5, 8, 12, 16, 23, 38, 56, 72, 91

$$MID = Int\left(\frac{LOW + HIGH}{2}\right)$$ ✓

If Item == A[MID] ?

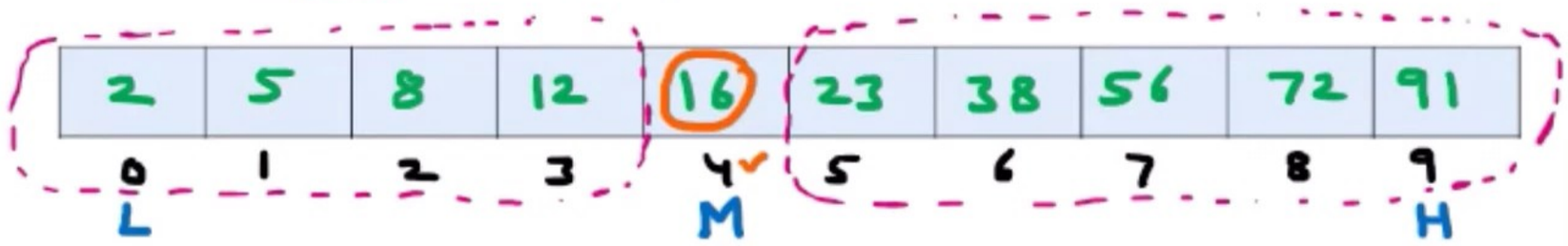
    Element Found ✗

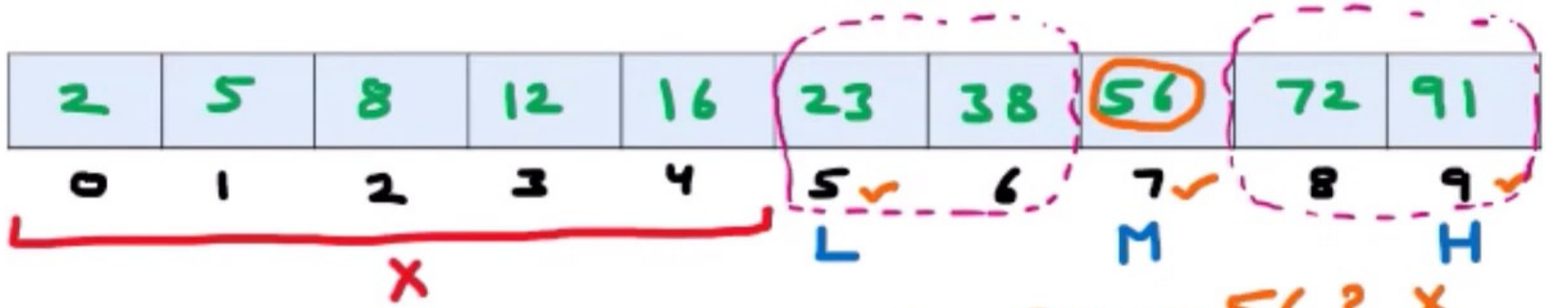
Else If Item < A[MID] ?

    HIGH = MID - 1 ✓

Else

    LOW = MID + 1



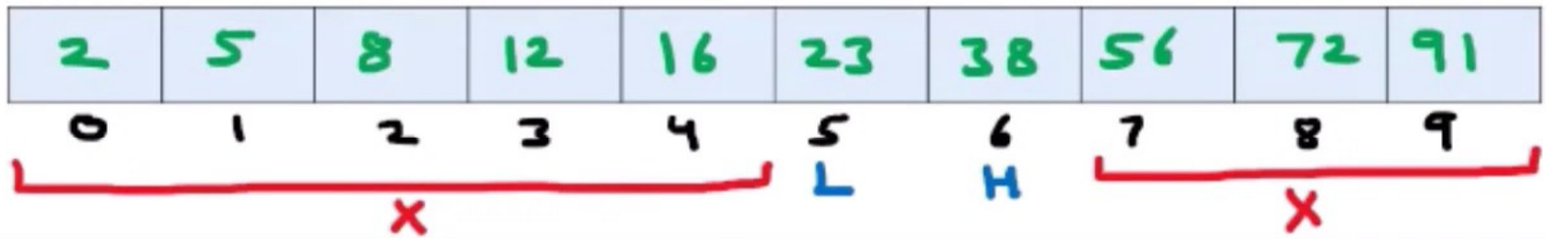


$$M = \frac{L+H}{2} = \frac{5+9}{2} = \frac{14}{2} = 7$$ ✓

23 == 56 ? ✗

23 < 56 ? ✓

H = M - 1 = 7 - 1 = 6 ✓



Binary Search

Search Item 23 from following Sorted Data using Binary Search

2, 5, 8, 12, 16, 23, 38, 56, 72, 91

$$MID = Int\left(\frac{LOW + HIGH}{2}\right)$$

If Item == A[MID]

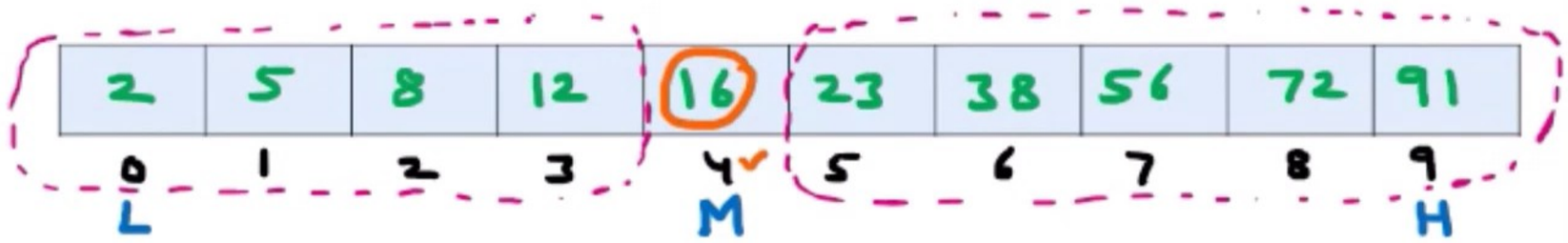
    Element Found

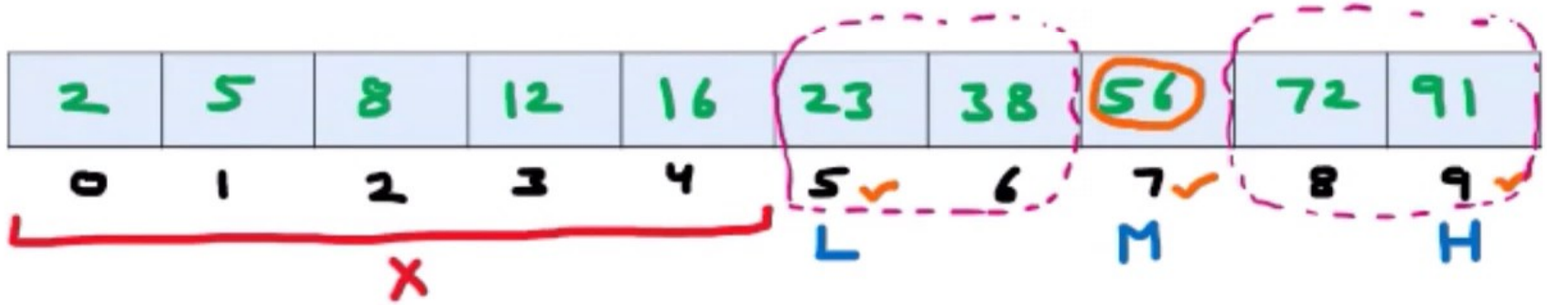
Else If Item < A[MID]

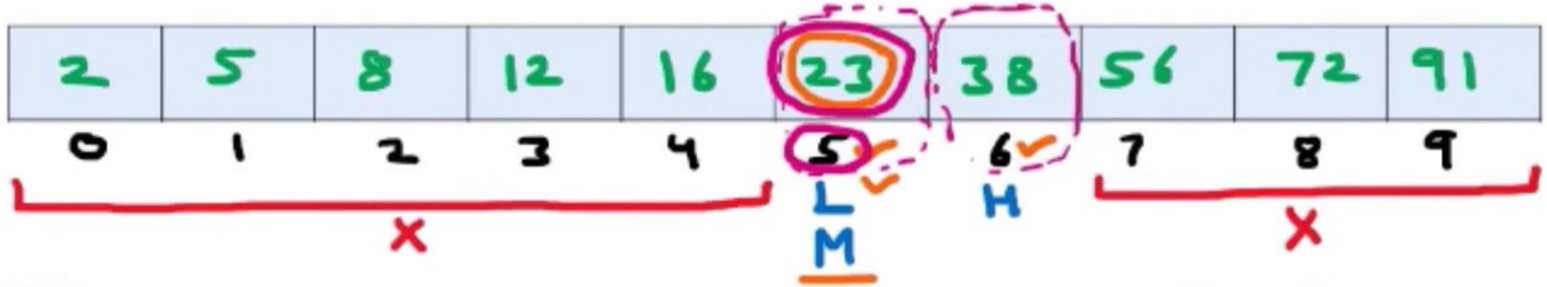
    HIGH = MID - 1

Else

    LOW = MID + 1







$M = \frac{L+H}{2} = \frac{5+6}{2} = \frac{11}{2}$

$= 5.5 = 5$

23 == 23 ?

Element Found at Position = M = 5

Time Complexity of Searching & Sorting Algorithms

Time Complexity of Searching Algorithms			
Algorithm	**Best Case**	**Average Case**	**Worst Case**
Linear Search	O(1)	O(n)	O(n)
Binary Search	O(1)	O(log n)	O(log n)

7, 6, 5, 2, 9, 22

Best — Average — Worst

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$

Time Complexity of Sorting Algorithms			
Algorithm	**Best Case**	**Average Case**	**Worst Case**
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$
Heap Sort	O(n log n)	O(n log n)	O(n log n)
Merge Sort	O(n log n)	O(n log n)	O(n log n)

Hashing
(Hash Addressing)

- Hashing is searching technique which is independent of number of elements
- Used in File Management where Key is used to Access Record
- Term hashing comes from technique of chopping a key into pieces

Hashing is divided into two parts:
- Hash Function ✓
- Collision Resolution ✓

Hash Table

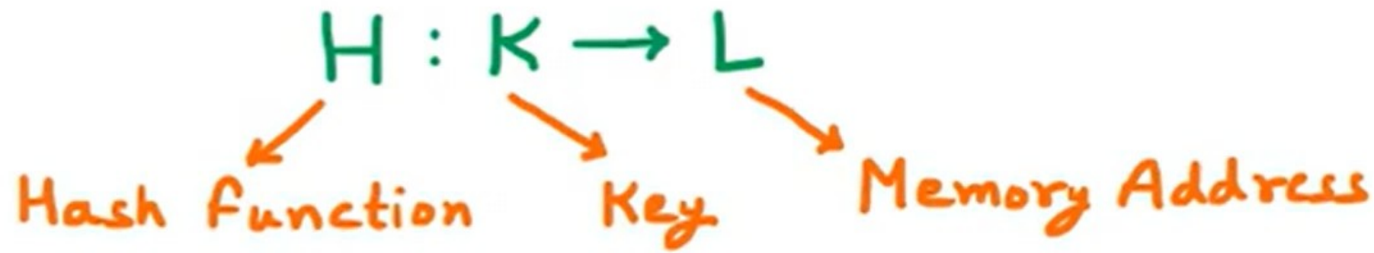
21	29	37	45	11
0	1	2	3	4

29, 45, 37, 11, 21

K = N mod 7 ✓

29 mod 7 = 1
45 mod 7 = 3
37 mod 7 = 2
11 mod 7 = 4
21 mod 7 = 0

Hash Function



Criteria of Selecting Hash function

- Function should be easy & quick to compare
- Minimum number of collision

Hash Function

Types of Hash Functions

Division Method

$$H(K) = K \pmod m$$

Remainder when K is divided by m

- Choose m larger than numbers
- To minimize collision, m should be Prime or Number with small divisors

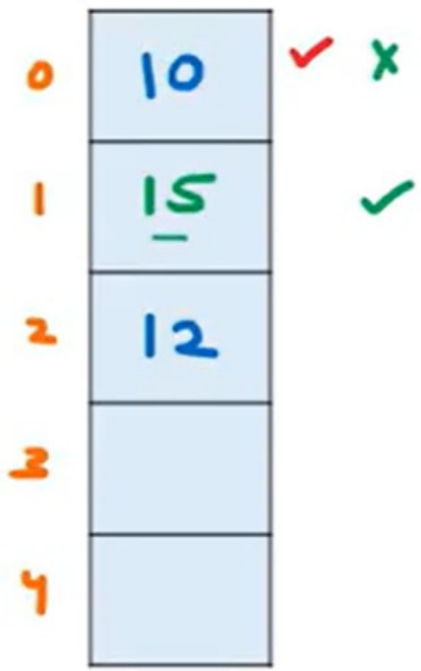
10, 15, 12

$H(10) = 10 \pmod 5 = 0$

$H(15) = 15 \pmod 5 = 0$ (Collision) (1)

$H(12) = 12 \pmod 5 = 2$

Hash Table



Hash Function

Types of Hash Functions

Mid Square Method

$H(K) = l$

- Key is squared (K^2)
- l is obtained by deleting the digits from both ends of K^2

K 10, 15, 12

K^2 $(10)^2$ $(15)^2$ $(12)^2$

100 225 144

x 0 x x 2 x x 4 x

0 2 4

Hash Table

0	10
1	
2	15
3	
4	12

Hash Function

Types of Hash Functions

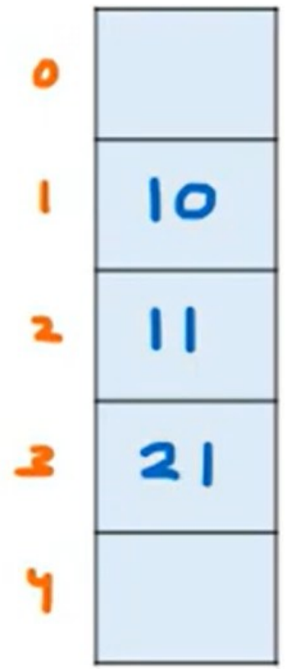
Folding Method

$$H(K) = K_1 + K_2 + K_3 + \_\_ + K_n$$

- Keys are partitioned into parts
- Parts are added together

K	10	21	11
	K_1 K_2	K_1 K_2	K_1 K_2
	1+0	2+1	1+1
	1	3	2

Hash Table



Collision

If we want to add new Record with Keys to file but Memory Address H(k) is already occupied, this is called Collision

23, 21, 20, 5, 10

$H(k) = k \pmod m$

$H(23) = 23 \pmod 5 = 3$

$H(21) = 21 \pmod 5 = 1$

$H(20) = 20 \pmod 5 = 0$

$H(5) = 5 \pmod 5 = 0$ (Collision)

Hash Table

0	20 ✓
1	21
2	
3	23
4	

Collision Resolution

- Linear Probing
- Chaining

Collision Resolution

Linear Probing

- Empty space is searched using Linear Search

23, 21, 20, 5, 10

$H(k) = k \pmod m$

$H(23) = 23 \pmod 5 = 3$

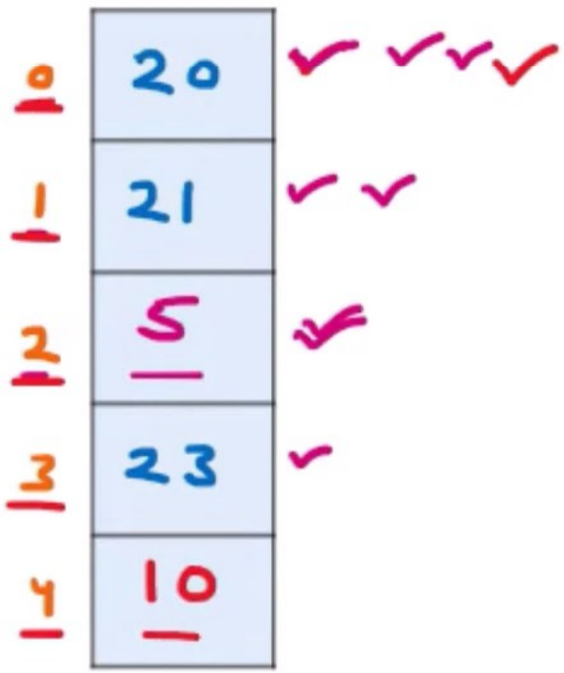
$H(21) = 21 \pmod 5 = 1$

$H(20) = 20 \pmod 5 = 0$

$H(5) = 5 \pmod 5 = 0$ (Collision) (5)

$H(10) = 10 \pmod 5 = 0$ (Collision)

Hash Table



Disadvantages

- Record tend to cluster & appear next to other
- Increase Average Time to Search Record

Collision Resolution

Chaining

- Records with same hash address are linked together in form of Linked List

23, 21, 20, 5, 10

$H(k) = k \pmod m$

$H(23) = 23 \pmod 5 = 3$ ✓

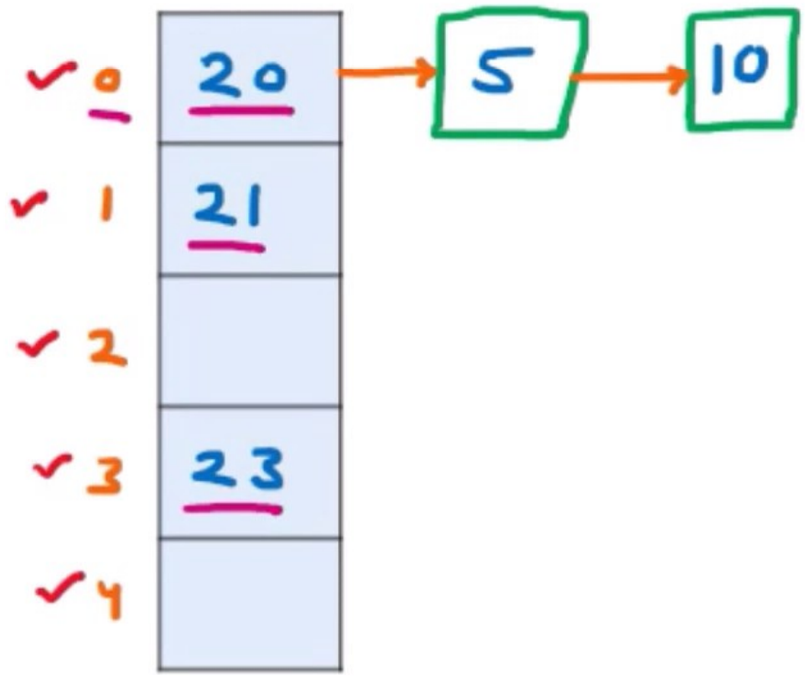
$H(21) = 21 \pmod 5 = 1$ ✓

$H(20) = 20 \pmod 5 = 0$ ✓

$H(5) = 5 \pmod 5 = 0$ (Collision)

$H(10) = 10 \pmod 5 = 0$ (Collision)

Hash Table



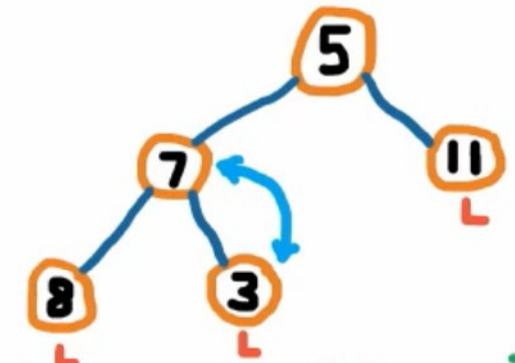
Heap Sort

Ankit Verma

Sort the following elements using Heap Sort

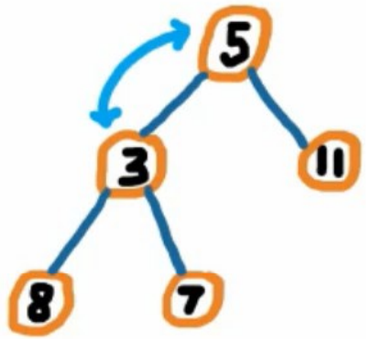
5, 7, 11, 8, 3

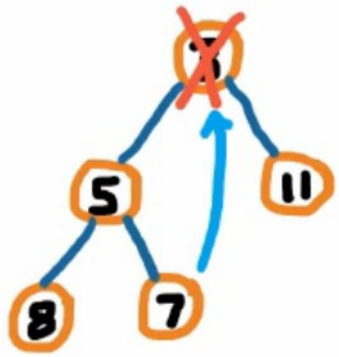
3, 5, 7, 8, 11 Sorted



Complete Binary Tree

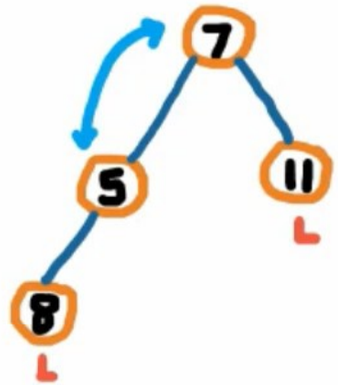
Heap

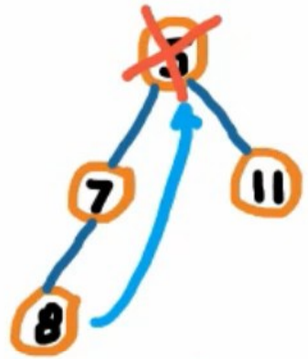




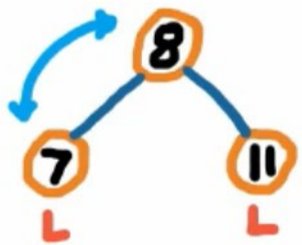
Min Heap

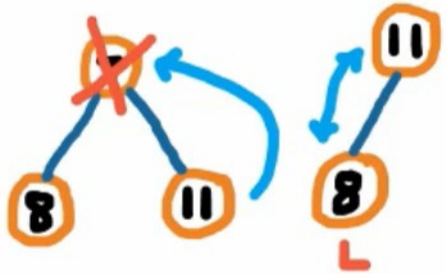
Heapify





Min Heap





Min Heap

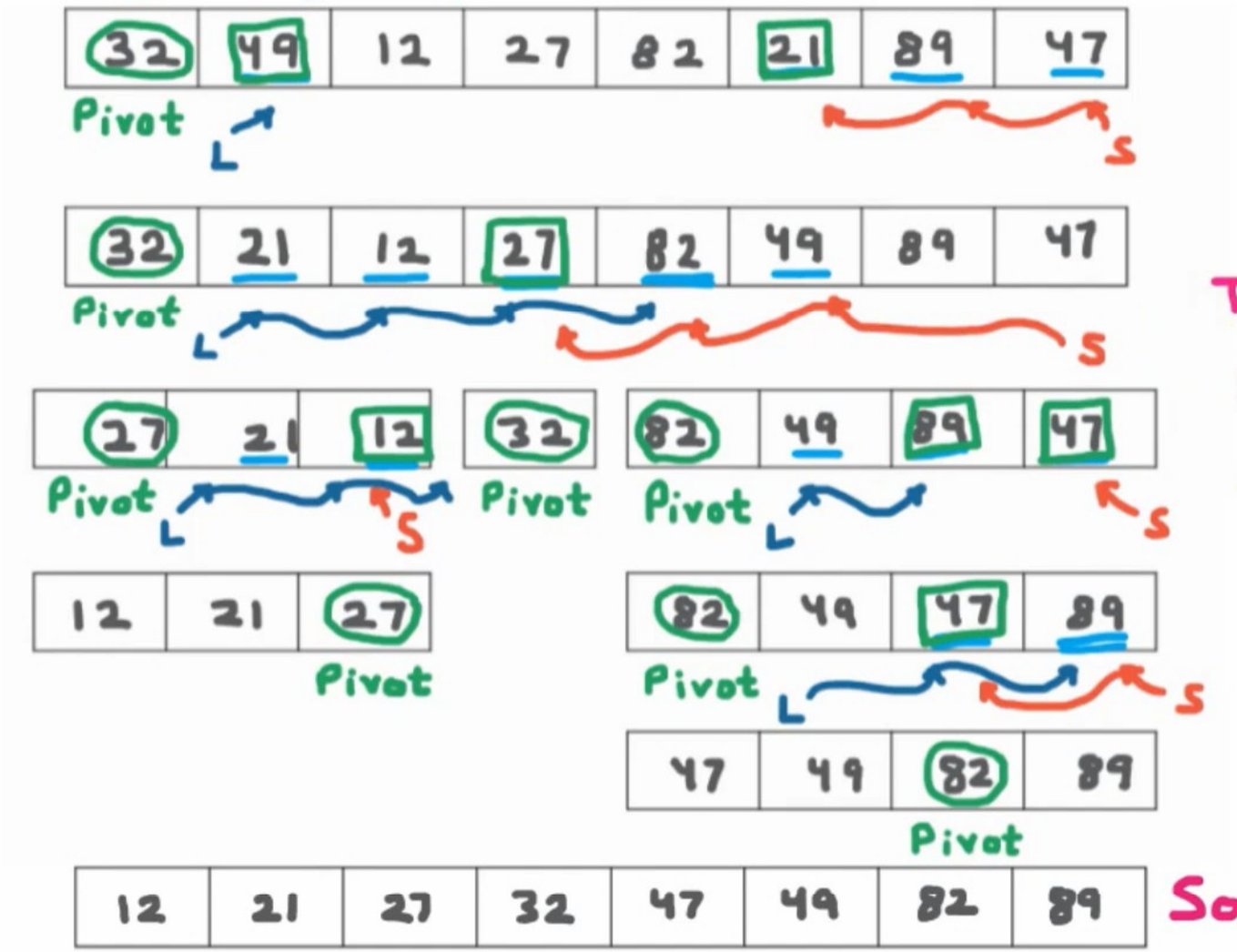
MinHeap

Quick Sort

DrAnkitVerma

Sort the following elements using Quick Sort

32, 49, 12, 27, 82, 21, 89, 47



Time Complexity

Worst Case $O(n^2)$

Average Case $O(n \log n)$

Best Case $O(n \log n)$

Radix Sort

DrAnkitVerma

Sort elements using Radix Sort

803, 57, 48, 93, 1, 72

H	T	O
8	0	3
0	5	7
0	4	8
0	9	3
0	0	1
0	7	2

H	T	O
0	0	1
0	7	2
8	0	3
0	9	3
0	5	7
0	4	8

H	T	O
0	0	1
8	0	3
0	4	8
0	5	7
0	7	2
0	9	3

H	T	O
0	0	1
0	4	8
0	5	7
0	7	2
0	9	3
8	0	3

1, 48, 57, 72, 93, 803 Sorted