

2. Searching and Sorting

⇒ Searching is the finding location of the data within the given data structure.

• Types of Searching

1] Linear Search

Linear Search is also called as Sequential Search. In Sequential Search all elements are examined sequentially starting from 1st element.

The process of searching terminates when list is ended or comparison results in success.

Also Search Algorithm starts comparison between data and key value.

• Advantages of Linear Search

- Simple
- Easy to understand
- not complex

• Disadvantages of Linear Search

- Should not be used for long list.

The Time for searching will increase if list is long.

searching and sorting

Ex: List = 10, 20, 30, 40, 50 & Search element 30

Step 1: 10 20 30 40 50

0 1 2 3 4

10 = 30 Index = Index + 1

Step 2: 10 20 30 40 50

20 = 30 Index = Index + 1

Step 3: 10 20 30 40 50 Search is successful

30 = 30, Name is found

2] Binary Search:

Binary Search is applicable only when the given array is sorted. This method makes a comparison between key value that is element to be search and the middle element or Centre element of the array.

Since elements are sorted, Comparison could be continue with either Left half of elements or right half of the elements.

Left half of the elements could be selected by simply making $J = \text{start}$

Right half of the elements could be selected by simply making $J = \text{end} + 1$

Process of selecting either the left half or right half continues until the element is found or element is not there.

Ex:

Array List: 0, 5, 11, 19, 27, 35

0 1 2 3 4 5

2 5 11 19 27 35

↑ ↑ ↑ ↑ ↑

i=0 c=2 j=5

i = index of first element

j = index of last element

$$C = \frac{\text{index of first element} + \text{index of last element}}{2}$$

$$C = \frac{i+j}{2}$$

$$C = \frac{0+5}{2} = 2$$

Case 1: if $\text{key} == a[C]$ then element is found

Case 2: if $\text{key} > a[C]$ then right half 19, 27, 35 is selected.

Case 3: if $\text{key} < a[C]$ then left half 0, 5 is selected.



○ Search the Element 29 in the given array
 $\{ 5, 9, 11, 15, 25, 29, 30, 35, 40 \}$

→	0	1	2	3	4	5	6	7	8	→ Index
	5	9	11	15	25	29	30	35	40	→ Element

i = Index of first element = 0

J = Index of last element = 8

$$\text{Centre} = \frac{i+J}{2}$$

$$= \frac{0+8}{2}$$

$$= 4$$

Centre 4 element 25

key Value 29 The element to be
 search that is key Value equal to 29

Step 1: $\text{key} > a[i]$ $29 > 25$ $\therefore$ $i = i+1$
 $29 > 25$ $\therefore$ $i = 5$

The right half part is selected

29	30	35	40	Element
5	6	7	8	Index
↑	↑		↑	
$i=5$	$C=6$		$J=8$	



$$c = \frac{i+j}{2} = \frac{5+8}{2} = \frac{13}{2} = 6$$

Step 2: key is less than $a[c]$. that is
 $29 < 30$.

Left half part will be selected.

$29 \rightarrow$ Element, $5 =$ Index

$\therefore$ left half part contain only one element
 that is key value

$\therefore$ search is successful.

$\therefore$ key value is get on location 5 that is 29

Q. Find the position of element 29 using Binary search in Array given below

$A = \{ 11, 5, 21, 3, 29, 17, 2, 43 \}$

$\rightarrow$ Sorted list of given array:

Data should be sorted for Binary search.

2	3	5	11	17	21	29	43	$\rightarrow$ Element
0	1	2	3	4	5	6	7	$\rightarrow$ Index

$i =$ Index of first element $= 0$

$j =$ Index of last element $= 7$

$$c = \frac{0+7}{2} = 3$$

Centre = 3

element = 11

key Value is 29 The element to be search in right part.

Step 1 : $\text{Key} > a[c]$
 $29 > 11$

The right half part is selected

#	4	5	6	7	→ Index
#	17	21	29	43	→ Element
	↓	↓		↓	
	i	c		j	

$$c = \frac{4+7}{2} = 5$$

Step 2 : $\text{Key} > 21$
 $\text{Key} > a[c]$
 $29 > 21$

Left part is selected.

Elements of left part

29 43
 6 7

$$\therefore \text{The } c = \frac{6+7}{2} = \frac{13}{2} = 6$$

$\therefore$ The (element) position of element 29 is found at position 6

⇒ Sorting - Arranging data in a logical order.

• Types of Sorting.

i) Bubble Sort

Bubble Sort is one of the simplest and most popular sorting method.

The basic idea behind bubble sort is as a bubble rises up in the water, the smaller element goes to the beginning.

This method is based on successive selecting element through exchange of adjacent element.

Let n be the number of elements in array the 1st pass begins with the comparison of $a[n-1]$ and $a[n-2]$. If $a[n-2]$ is larger than $a[n-1]$ the two elements are exchange.

Ex: P-1: 5 2 1

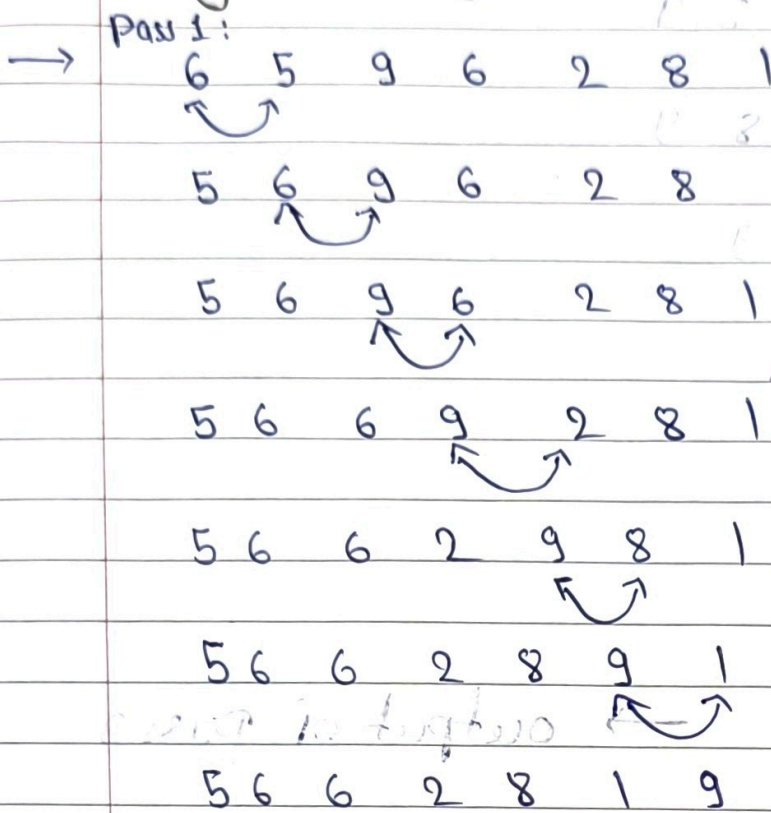
5 2 1
↗ ↘

P-2: 2 1 5
↗ ↘

1 2 5
↗ ↘

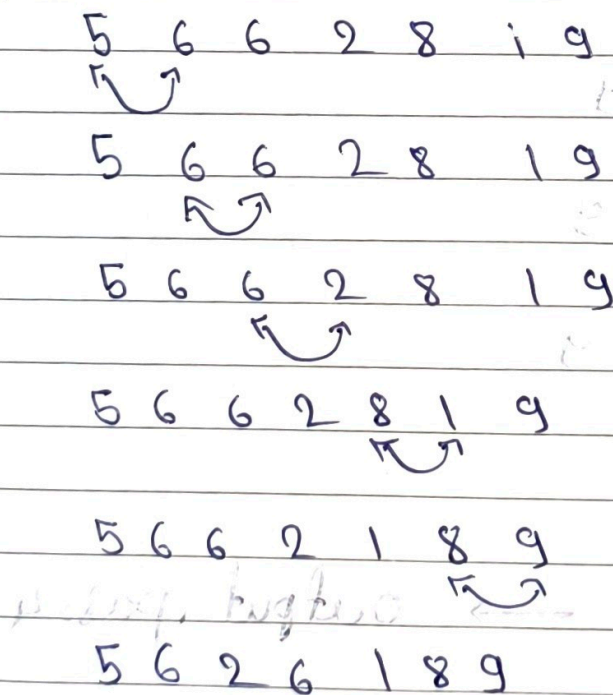
1 2 5

Q. original array with $n = 6, 5, 9, 6, 2, 8, 1$



Output of Pass 1

Pass 2 :



Output of Pass 2

Pass 3: 5, 2, 6, 1, 6, 8, 9

5 6 2 6 1 8 9

5 6 2 6 1 8 9

5 2 6 6 1 8 9

5 2 6 6 1 8 9

5 2 6 1 6 8 9

5 2 6 1 6 8 9

5 2 6 1 6 8 9

output of Pass 3

Pass 4:

5 2 6 1 6 8 9

2 5 6 1 6 8 9

2 5 6 1 6 8 9

2 5 1 6 6 8 9

2 5 1 6 6 8 9

2 5 1 6 6 8 9

output Pass 4



Pass 5: List is sorted with last 12
 List after pass 5: 2 5 1 6 6 8 9 2 5 1 2 1 2 1

2 5 1 6 6 8 9 2 5 1 2 1 2 1

Pass 6: List is sorted with last 12
 List after pass 6: 2 1 5 6 6 8 9 2 5 1 2 1 2 1

2 1 5 6 6 8 9 2 5 1 2 1 2 1

2 1 5 6 6 8 9

2 1 5 6 6 8 9 2 5 1 2 1 2 1

2 1 5 6 6 8 9

Pass 6:

2 1 5 6 6 8 9 2 5 1 2 1 2 1

1 2 5 6 6 8 9 Sorted List

Swapping logic:

temp = a[j]

a[j] = a[j+1]

a[j+1] = temp



2] Selection Sorting

Selection Sorting is the Very Simple Sorting Method.

In the i th pass we'll select the element with ~~lowest~~ lowest value among $a[i]$, $a[i+1]$, upto $a[i] n-1$ and be swap it with $a[i]$. As a result after i passes elements will be sorted order.

Ex: 5 9 1 11 2 4

original
Array

5 9 1 11 2 4

After pass 1 → 1 9 5 11 2 4

After pass 2 → 1 2 5 11 9 4

After pass 3 → 1 2 4 11 9 5

After pass 4 → 1 2 4 5 9 11

After pass 5 → 1 2 4 5 9 11



2. Sort the no in ascending order
 $A = \{42, 23, 74, 11, 65\}$

→ original array 42 23 74 11 65

The small no is 11
 Comparing with $a[0]$ and
 exchange it.

After Pass 1 11 23 74 42 65

The small no is 42
 Comparing with $a[2]$ &
 exchange it

After Pass 2 11 23 42 74 65

The small no is 65
 comparing with $a[3]$ &
 exchange it!

After Pass 3 11 23 42 65 74

11 23 42 65 74

→ Sorted array

3] Insertion Sort

An Element can always be placed at right place in the Sorted list

Ex: 5 9 10 15 20

Element to be place : 6
If the element 6 is to be inserted in sorted list of elements its right place is between 5 and 9

Elements with value greater than 6 should be moved right by 1 place.

Thus creating a space for incoming element.



Insertion Sort

Q1. Arrange the given no in Ascending order using Insertion Sort.

77 33 44 11 88 22 66 55

→

77

 |

33	44	11	88	22	66	55
----	----	----	----	----	----	----

 → Initial unsorted array

Sorted unsorted
Next element is 33 insert before 77 in Sorted array

33	77
----	----

 |

44	11	88	22	66	55
----	----	----	----	----	----

 → After Pass 1

Sorted unsorted
Next element is 44 insert between 33 & 77

33	44	77
----	----	----

 |

11	88	22	66	55
----	----	----	----	----

 → After Pass 2

Sorted unsorted
Next element is 11 insert before 33 in sorted array

11	33	44	77
----	----	----	----

 |

88	22	66	55
----	----	----	----

 → After Pass 3

Sorted unsorted
Next element is 88 insert before 77 in Sorted array

11	33	44	77	88
----	----	----	----	----

 |

22	66	55
----	----	----

 → After Pass 4

Sorted unsorted
Next element is 22 insert between 11 & 33 in sorted array



11 22 33 44 77 88 | 66 55 | → After Pass 5
 Sorted unsorted

Next element is 66 insert in between 44 & 77 in sorted array.

11 22 33 44 66 77 88 | 55 | → After Pass 6
 Sorted unsorted

Next element is 55 insert in between 44 & 66 in Sorted array

11 22 33 44 55 66 77 88 |
 OP: PP PP 2P 3P After Pass 7
Sorted list

41' Radix Sort :

Radix Sort is generalization of bucket sort to sort decimal numbers where the radix or base is 10. we need 10 bucket

In the first pass numbers are sorted on least significant digit. numbers with the same least significant digit are stored in same bucket in the.

In the 2nd pass numbers are sorted on 2nd least significant digit.

at the end of every pass numbers in buckets are merge to produce a Common List

Number of passes required to sort is equal to number of digit in the largest number in the list.

Sort 348, 14, 614, 6381, 47

3. 2011

2

Pass 2:

PPLE 2209.

009 001 200

000, 001, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837

47

14

14, 47, 348, 5381, 614

Pass 4:

614	147	348	5381	0	1	2	3	4	5	6	7	8	9
-----	-----	-----	------	---	---	---	---	---	---	---	---	---	---

merge list : 14, 47, 348, 614, 5381.

This is the sorted list

if $V < X$ then swap V and X

if $V > X$ then swap V and X

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

(2) 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

5] Quick Sort:

Quick Sort is the internal Sorting Algorithm with the time complexity equal to $O(n \log n)$

The basic algorithm to sort an array of n elements can be describe recursively as follow

Step 1: if $n \leq 1$ return

Step 2: Pick any element V in array $a[]$ this is called the pivot

Step 3: rearrange elements of the array by moving all elements $x, > V$

~~Step 4:~~ Right of V & all the elements $x, < V$ left of V

Q. 5, 1, 2, 9, 8, 13

→

5 | 2 9 8 13

← original list / array

Select Pivot element 5

1 2 0 5 9 8 13

Selecting 1 as pivot elem
 0 1 2 5 8 9 13

selecting 9 as pivot element

This is sorted list by 1, 2, 5, 8, 9, 13
 using quick sort algorithm

Time Complexities of different Algorithm

1) Binary Search.

Time Complexity: $\log_2 n$

2) Sequential Search

Time Complexity: $O(n)$

3) Bubble Sort

Time Complexity: $O(n^2)$

4) Insertion Sort

Time Complexity: $O(n^2)$

5) Selection Sort

Time Complexity: $O(n \log n)$

6) Quick Sort

Time Complexity: $O(n^2)$