

← → NS 11 Apr 2023

J Main.java ×

J Main.java > Main > main(String[])

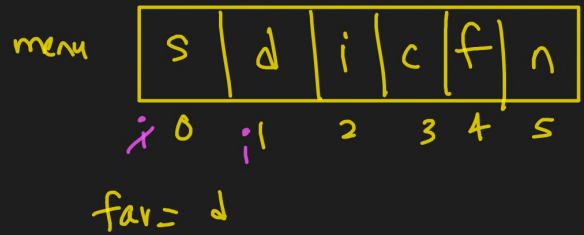
```
44 // Yogesh
45 class Main {
46
47     public static int linearSearch(String[] menu, String favourite) {
48         for (int i = 0; i < menu.length; i++) {
49
50             if (menu[i] == favourite) {
51
52                 return i;
53
54             }
55
56         }
57         return -1;
58     }
59
60     public static void main(String args[]) {
61
62         String[] menu = { "samosa", "dosa", "idli", "coke", "fruity", "noodles" };
63
64         String favourite = "dosa";
65
66         int index = linearSearch(menu, favourite);
67
68         if (index == -1) {
69
70
71         }
72     }
73 }
```

Run | Debug

Ln 70, Col 1 Tab Size: 4 UTF-8 LF {} Java

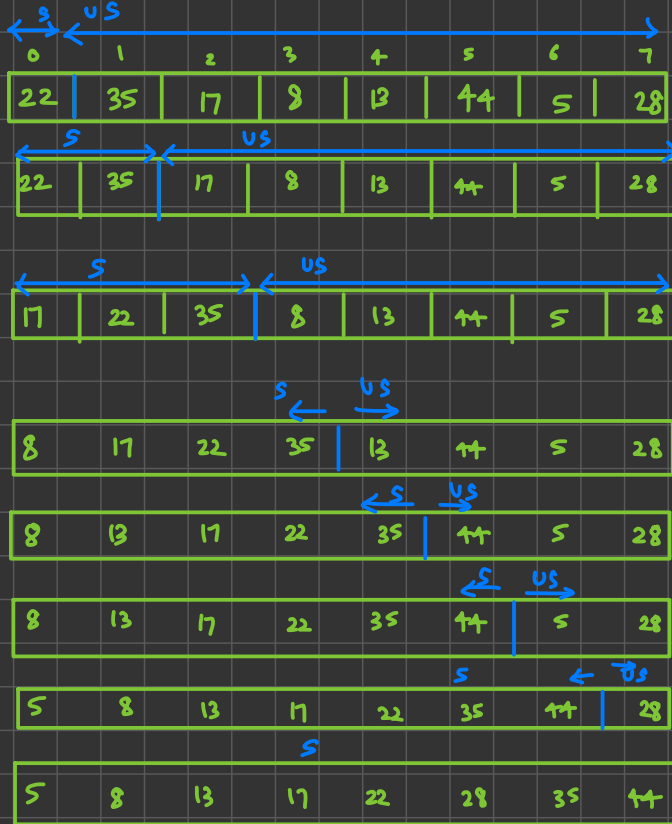
Live Share

menu.length = 6



Insertion Sort : efficient for sorting small collection of data.

↳ Online Sorting : becz it sorts elements as & when it receives it .



11 Apr 2023

key = arr[i] = 13

i = 4

j = i - 1 = 3

while (j >= 0 && arr[j] > key)

arr[j+1] = arr[j]

j = j - 1

arr[j+1] = key

Initial array: [22, 35, 17, 8, 13, 44, 5, 28]

Step 1: Pivot = 8 (index 3). Elements less than 8 are moved to the left, and elements greater than 8 are moved to the right.

Step 2: Array becomes [17, 22, 35, 8, 13, 44, 5, 28].

Step 3: Pivot = 13 (index 4). Elements less than 13 are moved to the left, and elements greater than 13 are moved to the right.

Step 4: Array becomes [8, 17, 22, 35, 13, 44, 5, 28].

Step 5: Pivot = 17 (index 2). Elements less than 17 are moved to the left, and elements greater than 17 are moved to the right.

Step 6: Array becomes [8, 13, 17, 22, 35, 44, 5, 28].

Step 7: Pivot = 22 (index 3). Elements less than 22 are moved to the left, and elements greater than 22 are moved to the right.

Step 8: Array becomes [8, 13, 17, 22, 35, 44, 5, 28].

Step 9: Pivot = 35 (index 5). Elements less than 35 are moved to the left, and elements greater than 35 are moved to the right.

Step 10: Array becomes [5, 8, 13, 17, 22, 35, 44, 28].

Step 11: Pivot = 28 (index 7). Elements less than 28 are moved to the left, and elements greater than 28 are moved to the right.

Step 12: Array becomes [5, 8, 13, 17, 22, 28, 35, 44].

NS 11 Apr 2023

Main.java 1 x

Main.java > Main > insertionsort(int[])

```

85 {
86     static void insertionsort(int arr[])
87     {
88         for(int i=1; i<arr.length; i++)
89         {
90             int key = arr[i];
91             int j = i-1;
92             while(j>=0 && arr[j]>key)
93             {
94                 arr[j+1] = arr[j];
95                 j = j-1;
96             }
97             arr[j+1] = key;
98         }
99     }
100 }
101
Run | Debug
102 public static void main(String[] args)
103 {
104     int i, n;
105     Scanner sc = new Scanner(System.in);
106     System.out.print("Enter the number of elements: ");
107     n = sc.nextInt();
108     int arr[] = new int[n];
109     System.out.print("Enter the elements one by one: ");
110     for(i=0; i<n; i++)

```

Handwritten annotations:

- Diagram showing the insertion sort process with indices 0, 1, 2, 3, 4 and values 9, 2, 7, 4, 9.
- Diagram showing the array state: $\begin{matrix} 9 & 2 & 7 & 4 & 9 \end{matrix}$ with arrows indicating shifts.
- Handwritten text: $key = 2$.

Ln 94, Col 14 Tab Size: 4 UTF-8 LF {} Java

Quick Sort

pivot = 7

0	1	2	3	4	5	6	7	8
7	6	10	5	9	2	1	15	7
7	6	5						e
7	6	7	5	9	2	1	15	10
		7	5	5		e	e	e
7	6	7	5	1	2	9	15	10
=				1	2	9		
				e	e	s		
2	6	7	5	1	7	9	15	10

Revise Recursion < Fibonacci Factorial

if u really want to understand Quick Sort.

X