Quick Sort

Quick Sort is an efficient divide-and-conquer algorithm. It divides a large list into two smaller sub-lists based on a pivot chosen, into smaller and larger elements. Quick Sort then recursively does this to the sub-lists finally producing a sorted list.

**Steps**

1. Pick an element, called the pivot.
2. Partitioning: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it. After this partitioning, the pivot is in its final position. This is called the partition operation.
3. Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values.

Chart

Description automatically generated

1. #include <stdio.h>
2. **void** swap(**int** a, **int** b)
3. {
4. **int** t = a;
5. a = b;
6. b = t;
7. }
8. **int** partition (**int** arr[], **int** low, **int** high)
9. {
10. **int** pivot = arr[high]; // pivot
11. **int** i = (low - 1); // index of smaller element
12. **for** (**int** j = low; j <= high- 1; j++)
13. {
14. // if current element is smaller than the pivot
15. **if** (arr[j] < pivot)
16. {
17. i++; // increment index of smaller element
18. swap(&arr[i], &arr[j]);
19. }
20. }
21. swap(&arr[i + 1], &arr[high]);
22. **return** (i + 1);
23. }
25. **void** quick\_sort(**int** arr[], **int** low, **int** high)
26. {
27. **if** (low < high)
28. {
29. **int** pi = partition(arr, low, high);
30. quick\_sort(arr, low, pi - 1);
31. quick\_sort(arr, pi + 1, high);
32. }
33. }
34. **int** main()
35. {
36. **int** a[100], n, i;
37. printf("No. of elements to sort");
38. scanf("%d", &n);
39. printf("\nEnter the elements:\n");
40. **for**(i = 0; i < n; i++)
41. scanf("%d", &a[i]);
42. quick\_sort(a, 0, n - 1);
43. printf("\nArray after sorting:");
44. **for**(i = 0; i < n; i++)
45. printf("%d ",a[i]);
46. **return** 0;
47. }</stdio.h>