Quick sort:

Algorithm:

1. Take an unsorted array as an input.
2. Choose the first element as a pivot element of an array.
3. Take two variables to point left and right of the list excluding pivot.
4. Left variable points to the low index and right variable points to the high index.
5. If value at left index is less than the pivot, move the pointer towards right.
6. If value at right index is greater than the pivot, move the pointer towards left.
7. If condition in the both step 5 and step 6 does not satisfied swap left and right.
8. Repeat step 5 and 6 until left pointer crosses right.
9. When the left>right swap the right pointer with pivot element.
10. Now the pivot element is at its right position.
11. Again, take the first element as a pivot and repeat the same procedure until the array gets sorted.

Program code:

```
#include <bits/stdc++.h>

using namespace std;

void swap(int* a, int* b)

{

    int t = *a;

    *a = *b;

    *b = t;

}


int part (int arr[], int low, int high)

{

    int pivot = arr[high];

    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quick(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = part(arr, low, high);

        quick(arr, low, pi - 1);
        quick(arr, pi + 1, high);
    }
}

void print(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;

}



int main()
{
    int arr[10000];
    int n = sizeof(arr) / sizeof(arr[0]);
    for (int i = 0; i<n; i++){
    arr[i] = i;
}

    quick(arr, 0, n - 1);
    cout << "Sorted array: \n";
    print(arr, n);
    return 0;

}
```