Question

Sorting Algorithms.

1. Explain how main sorting algorithms can be performed using an appropriate array example.

2. Compare and contrast bubble sort and selection sort algorithms.

3. What are the real-world examples of sorting.

4. Write function using pseudo or source codes to sort an integer array using bubble sort and selection sort.

Answers



a. Bubble Sort:

Bubble Sort compares adjacent elements and swaps them if they are in the wrong order, repeatedly iterating through the array until it is sorted.

Example:

Let's consider an array: [5, 2, 8, 1, 6]

Pass 1: [2, 5, 1, 6, 8]

Pass 2: [2, 1, 5, 6, 8]

Pass 3: [1, 2, 5, 6, 8]

The array is now sorted in ascending order.

b. Selection Sort:

Selection Sort divides the array into two parts: the sorted part and the unsorted part. It repeatedly selects the minimum (or maximum) element from the unsorted part and swaps it with the first element of the unsorted part, expanding the sorted part.

Example:

Let's consider an array: [5, 2, 8, 1, 6]

Pass 1: [1, 2, 8, 5, 6]

Pass 2: [1, 2, 8, 5, 6]

Pass 3: [1, 2, 5, 8, 6]

Pass 4: [1, 2, 5, 6, 8]

The array is now sorted in ascending order.

c. Insertion Sort:

Insertion Sort builds the final sorted array one element at a time by comparing each element with the elements before it and inserting it at the correct position.

Example:

Let's consider an array: [5, 2, 8, 1, 6]

Pass 1: [2, 5, 8, 1, 6]

Pass 2: [2, 5, 8, 1, 6]

Pass 3: [1, 2, 5, 8, 6]

Pass 4: [1, 2, 5, 6, 8]

The array is now sorted in ascending order.



|  |  |
| --- | --- |
| Bubble Sort | Selection Sort |
| Compare adjacent elements and swap if necessary | Find the minimum element and swap it with the first element of the unsorted part |
| Time Complexity  Worst Case: O(n^2)  Average Case: O(n^2)  Best Case: O(n) | Time Complexity  Worst Case: O(n^2)  Average Case: O(n^2)  Best Case: O(n^2) |
| Space Complexity  In-place sorting, uses constant extra space | Space Complexity  In-place sorting, uses constant extra space |
| Stable | Not stable |
| Requires more swaps in general | Requires fewer swaps in general |
| Less efficient for large datasets | More efficient for large datasets |
| Small-sized arrays or nearly sorted arrays | Small-sized arrays or nearly sorted arrays |



Sorting is a very common operation in computer science, and it has many real-world applications. Some examples of sorting in the real world include:

* Sorting data in a database
* Sorting files on a computer
* Sorting search results
* Sorting playlists
* Sorting cards in a deck



Bubble Sort:

BubbleSort(arr):

n = length of arr

for i from 0 to n-1:

for j from 0 to n-i-1:

if arr[j] > arr[j+1]:

swap arr[j] and arr[j+1]

Selection Sort:

SelectionSort(arr):

n = length of arr

for i from 0 to n-1:

minIndex = i

for j from i+1 to n:

if arr[j] < arr[minIndex]:

minIndex = j

swap arr[i] and arr[minIndex]