eeksforGeeks
A computer science portal for geeks
es

Analysis of different sorting techniques

GeeksforGeeks
A computer science portal for geeks

In this article, we will discuss i techniques including their complexity, stability and memo this article, you should understand basics of different sorting techniques (See : Sorting Techniques).

Time complexity Analysis –
We have discussed the best, average and wor complexity of different sorting techniques with possible scenarios.

Comparison based sorting –
In comparison based sorting, elements of an array are compared with each other to find the sorted array.

arpansahu

My Profile

Logout

- **Bubble sort and Insertion sort –**
  Average and worst case time complexity: n^2
  Best case time complexity: n when array is already sorted.
  Worst case: when the array is reverse sorted.
- **Selection sort –**
  Best, average and worst case time complexity: n^2 which is independent of distribution of data.
- **Merge sort –**
  Best, average and worst case time complexity: nlogn which is independent of distribution of data.
- **Heap sort –**
  Best, average and worst case time complexity: nlogn which is independent of distribution of da
- **Quick sort –**
  It is a divide and conquer approach with recurrence relation:

```
T(n) = T(k) + T(n-k-1) + cn
```

Worst case: when the array is sorted or reverse sorted, the partition algorithm divides the array in two subarrays with 0 and n-1 elements. Therefore,

```
T(n) = T(0) + T(n-1) + cn
Solving this we get, T(n) = O(n^2)
```

Best case and Average case: On an average, the partition algorithm divides the array in two subarrays with equal size. Therefore,

```
T(n) = 2T(n/2) + cn
Solving this we get, T(n) = O(nlogn)
```

Non-comparison based sorting –
In non-comparison based sorting, elements of array are not compared with each other to find the sorted array.

- **Radix sort –**
  Best, average and worst case time complexity: nk where k is the maximum number of digits in elements of array.
- **Count sort –**
  Best, average and worst case time complexity: n+k where k is the size of count array.
- **Bucket sort –**
  Best and average time complexity: n+k where k is the number of buckets.
  Worst case time complexity: n^2 if all elements belong to same bucket.

In-place/Outplace technique –
A sorting technique is inplace if it does not use any extra memory to sort the array.
Among the comparison based techniques discussed, only merge sort is outplaced technique as it requires an extra array to merge the sorted subarrays.
Among the non-comparison based techniques discussed, all are outplaced techniques. Counting sort uses a counting array and bucket sort uses a hash table for sorting the array.

Online/Offline technique –
A sorting technique is considered Online if it can accept new data while the procedure is ongoing i.e. complete data is not required to start the sorting operation.
Among the comparison based techniques discussed, only Insertion Sort qualifies for this because of the underlying algorithm it uses i.e. it processes the array (not just elements) from left to right and if new elements are added to the right, it doesn't impact the ongoing operation.

Stable/Unstable technique –
A sorting technique is stable if it does not change the order of elements with the same value.
Out of comparison based techniques, bubble sort, insertion sort and merge sort are stable techniques.

Selection sort is unstable as it may change the order of elements with the same value. For example, consider the array 4, 4, 1, 3.

In the first iteration, the minimum element found is 1 and it is swapped with 4 at 0th position. Therefore, the order of 4 with respect to 4 at the 1st position will change. Similarly, quick sort and heap sort are also unstable.

Out of non-comparison based techniques, Counting sort and Bucket sort are stable sorting techniques whereas radix sort stability depends on the underlying algorithm used for sorting.

Analysis of sorting techniques :

Fast, easy, and secure WordPress Hosting in minutes.
Starts at ~~₹499.00~~ ₹199/mo
Get Started
Recommended by WordPress.org since 20
bluehost

- When the array is almost sorted, insertion sort can be preferred.
- When order of input is not known, merge sort is preferred as it has worst case time complexity of nlogn and it is stable as well.
- When the array is sorted, insertion and bubble sort gives complexity of n but quick sort gives complexity of n^2.

Que – 1. Which sorting algorithm will take the least time when all elements of input array are identical? Consider typical implementations of sorting algorithms.

(A) Insertion Sort

(B) Heap Sort

(C) Merge Sort

(D) Selection Sort

Solution: As discussed, insertion sort will have the complexity of n when the input array is already sorted.

Que – 2. Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then, (GATE-CS-2012)

(A) T(n) <= 2T(n/5) + n

(B) T(n) <= T(n/5) + T(4n/5) + n

(C) T(n) <= 2T(4n/5) + n

(D) T(n) <= 2T(n/2) + n

Solution: The complexity of quick sort can be written as:

```
T(n) = T(k) + T(n-k-1) + cn
```

As given in question, one list contains 1/5th of total elements. Therefore, another list will have 4/5 of total elements. Putting values, we get:

T(n) = T(n/5) + T(4n/5) + cn, which matches option (B).

Time and Space Complexity Comparison Table :

SORTING ALGORITHM	TIME COMPLEXITY			SPACE COMPLEXITY
	BEST CASE	AVERAGE CASE	WORST CASE	WORST CASE
Bubble Sort	**Ω(N)**	**Θ(N^2)**	**O(N^2)**	**O(1)**
Selection Sort	**Ω(N^2)**	**Θ(N^2)**	**O(N^2)**	**O(1)**
Insertion Sort	**Ω(N)**	**Θ(N^2)**	**O(N^2)**	**O(1)**
Merge Sort	**Ω(N log N)**	**Θ(N log N)**	**O(N log N)**	**O(N)**
Heap Sort	**Ω(N log N)**	**Θ(N log N)**	**O(N log N)**	**O(1)**
Quick Sort	**Ω(N log N)**	**Θ(N log N)**	**O(N^2)**	**O(N log N)**
Radix Sort	**Ω(N k)**	**Θ(N k)**	**O(N k)**	**O(N + k)**
Count Sort	**Ω(N + k)**	**Θ(N + k)**	**O(N + k)**	**O(k)**
Bucket Sort	**Ω(N + k)**	**Θ(N + k)**	**O(N^2)**	**O(N)**

Recommended Posts:

Asymptotic Analysis and comparison of sorting algorithms
Analysis of Algorithm | Set 5 (Amortized Analysis Introduction)
Analysis of Algorithms | Set 1 (Asymptotic Analysis)
Analysis of Algorithms | Set 4 (Analysis of Loops)
Analysis of Algorithms | Big-O analysis
Techniques to handle Thrashing
Concurrency Control Techniques
Resource Allocation Techniques for Processes
Database Recovery Techniques in DBMS
Congestion Control techniques in Computer Networks
Implementing Non-contiguous Memory Management Techniques
Implementation of Contiguous Memory Management Techniques
FIRST Set in Syntax Analysis
FOLLOW Set in Syntax Analysis
Introduction of Lexical Analysis

This article is contributed by **Sonal Tuteja**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Improved By : Akshay Gaursundar, mdamircoder

Article Tags : Analysis GATE CS

19

2

☐ To-do ☐ Done

Based on **7** vote(s)

Feedback/ Suggest Improvement Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks, Some rights reserved