Sorting

Assignment Project Exam Help

Dr Timothy Kimber

https://eduassistpro.github.i

Add WeChat edu_assist_pr

The Sorting Problem

Assignment Project Exam Help

- Sorting data is one of the most thoroughly explored computing problems.

Problem

In
Out

such that $a'_1 \leq a'_2 \leq \cdots \leq a'$

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- Sorting is an important problem. It is part of the solution to other problems.
- Understanding the complexity of sorting algorithms helps design good solutions to these other problems.

Incremental Sorting

The incremental approach of Simple Search can be applied to sorting.

Assignment Project Exam Help

https://eduassistpro.github.i

- Proceed from left
- Gradually grow a sorted region (note loop invari

Add WeChat edu_assist_pr

EXERCISE

Invent an incremental sorting algorithm.

Incremental Sorting

Assignment Project Exam Help

https://eduassistpro.github.i

There are two options:

1. Add the next element `a[i]` to the sorte
2. Add the lowest element outside the sorted regio

Add WeChat edu_assist_pr

Option 1 leads to the Insertion Sort algorithm

Insertion Sort

- Insertion Sort divides a into a sorted part, initially just $a[0]$, and the remaining unsorted part
- Elements from the unsorted part are then inserted into the sorted part

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Insertion Sort

Insertion Sort(Input: sequence $a = [a_0, \ldots, a_{N-1}]$)

```
For i = 1 to N                  // initialise invariant
    next = a[i]                 // save. a[i] overwritten later
    j    = i
    While
      j = j - 1
    EndWhile
    a[j] = next
EndFor
```

Assignment Project Exam Help
https://eduassistpro.github.i
Add WeChat edu_assist_pr

- The sorted region can be initialised to contain `a[0]`
- Do not need to compare `next` with all `a[0,..,i-1]` (sorted)

Time Complexity

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- What is the worst case input?
- What is the best case input?
- What is the time complexity in the best and worst cases?

Worst Case

- Running time of Insertion Sort has two dimensions:
  - Number of insertions
  - Size of insertion
- Informally: both dimensions are $\Theta(N)$, so $T(N) = \Theta(N^2)$

Assignment Project Exam Help
https://eduassistpro.github.i
Add WeChat edu_assist_pr

Worst Case

More formally, the total number of iterations of the inner loop is

Assignment Project Exam Help

$$t(N) = 1 + 2 + \cdots + (N-1) = \sum_{i=1}^{N-1} i$$

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Worst Case

So, the worst case time complexity is $aN^2 + bN + c \leq O(N^2)$

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Best Case

In the best case

- $t(N) = N - 1$
- So, $T(N) = aN + b = \Theta(N)$

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Other Properties

- Insertion Sort works in place (space complexity is $\Theta(1)$)
- Assuming randomised input, average case is $T(N) = \Theta(N^2)$
- For any input $T(N) = O(N^2)$

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Divide and Conquer

Will a divide and conquer approach work?

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- Divide into subproblems
- Solve the subproblems
- Combine into overall solution

EXERCISE

Design a combining algorithm.

Combining Sorted Sequences

Merge (Input: array a, indices l, m and r, where $r > m \geq l$)

```
left = a[l,...,m-1]
right = a[m,...,r-1]
i = j = 0, k = l
whi
  i

  else
    a[k] = left[j]
    j = j + 1
  end
  k = k + 1
end
```

Assignment Project Exam Help
https://eduassistpro.github.i
Add WeChat edu_assist_pr

- The procedure takes $\Theta(N)$ time for N total elements

Divide and Conquer

Assignment Project Exam Help

https://eduassistpro.github.i

- Time to combine subproblem solutions is Θ(

Add WeChat edu_assist_pr

EXERCISE

What is worst case time complexity of divide and conqu

- Write recurrence (assume $N = 2^a$ so no floors)
- Solve using master theorem

Time Complexity

Assignment Project Exam Help

The proposed algorithm divides the problem (in constant time) into 2 subproblems of size $N/2$, solves both, and combines the solutions in $\Theta(N)$ time, so the time complexity is:

https://eduassistpro.github.i

So, $N^{\log_b a} = N^{\log_2 2} = N^1 = N$, and therefor

- $f(N) = \Theta(N^{\log_b a}) = \Theta(N^{\log_b a} \times \log_2^0 N)$

Add WeChat edu_assist_pr

and Case 2, with $k = 0$, applies.

- $T(N) = \Theta(N^{\log_b a} \log_2^1 N) = \Theta(N \log_2 N)$

The divide and conquer algorithm is faster than Insertion Sort. Surprised?

Time Complexity

Alternative informal view of time complexity: recursion tree

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- Each level of the tree contributes cN
- There are $\log_2 N + 1$ levels

MergeSort

You have invented Mergesort

Assignment Project Exam Help

```
MergeSort(input array a, index l, index r)
    if r - l < 2
      r
    m = (l + r) / 2
    Mer
    MergeSort(a, m, r)
    Merge(a,l,m,r)
    return
```

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- The sorting appears to be happening in place, but the list is copied during `Merge`
- What is the best case?

Properties of Merge Sort

Assignment Project Exam Help

- Spa
- Tim
- Fas
- Slower than Insertion Sort if the list is already sorte
- Slower than Insertion Sort for small N

https://eduassistpro.github.i
Add WeChat edu_assist_pr

Alternative Divide and Conquer

- Merge Sort divides the data in half and sorts the halves
- Alternative approach: do a quick (rough) sort into two groups
- a_p is called the pivot and the division ensures that $a \quad A_{low}(a < a_p)$ and

Assignment Project Exam Help
https://eduassistpro.github.i
Add WeChat edu_assist_pr

- Left with subproblems of sorting A_{low} and A_{high}
- No combining needed

Quicksort

This procedure sorts the array $a[l, ..., r-1]$

Assignment Project Exam Help

Quicksort (Input: array a, index l, index r)

```
if r < l + 2                    // 0 or 1 elements to so
  r
p = Par
Qui
Quicksort(a, p + 1, r)
return
```

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- The Quicksort divide step is called partitioning
- The Partition procedure must return the final index of the pivot
- The base case must work for an empty array

Suggested Partition Design

Assignment Project Exam Help

- Use last element as the pivot
- Maintain two subarrays which grow
- Ele
- Ele
- Ele

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Tutorial Exercise

Write the Partition procedure.

Lomuto Partitioning

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Partition

This procedure partitions the array $a[l, ..., r-1]$

Assignment Project Exam Help

```
Partition (Input: array a, index l, index r)
    i = j = l                  // both partitions are empty
    p = r - 1
    whi
      i
        swap(a, i, j)
        j = j + 1
      i = i + 1
    swap(a, p, j)
    return j
```

https://eduassistpro.github.i

Add WeChat edu_assist_pr

- The time complexity is $\Theta(N)$ (where $N = r - l$)

Quicksort Performance

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Question

What is the worst case time complexity of Quicksort. A

Quicksort Worst Case

The given `partition` procedure:

- will only remove one element from sorted or reverse-sorted data
- will b

Assignment Project Exam Help
https://eduassistpro.github.i
Add WeChat edu_assist_pr

This leads to incremental execution resembling inse

- N levels of recursion
- $N - i$ elements to partition at level i
- So worst case time complexity is $\Theta(N^2)$

Quicksort Best Case

- Fewest levels of recursion when the partitioning is balanced
- Subproblems are no larger than $N/2$
- Same bound as Mergesort: $\Theta(N \log_2 N)$
- As re

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Quicksort Performance

With rather unbalanced partitioning performance is still $O(N \log_2 N)$

- Any dividing factor will introduce a $\log N$ term
- e.g. 9 to 1 (Cormen p. 176)

Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Randomised Quicksort

If the partition procedure is altered to choose the pivot at random, then the worst-case behaviour becomes a matter of chance for all inputs.

Assignment Project Exam Help

Partitio

```
x = ran
swa
... a
```

https://eduassistpro.github.i

Assuming N distinct values:

Add WeChat edu_assist_pr

- The probability of choosing the worst pivot in eve !
- This becomes vanishingly small as N increases
- Randomised Quicksort is algorithm of choice if N more than ~ 10

Expected Performance

Assignment Project Exam Help

Possible to give average case time complexity for Randomised Quicksort:

- Assume N distinct values again
- Ran
- Tim
- Pro r rank by value (see books for full explanation)
- Average number of comparisons is sum of proba i : j

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Average case complexity is $\Theta(N \log_2 N)$

- This is called expected running time for randomised algorithm

Partition Variations

Assignment Project Exam Help

There are many ways to implement partitioning. E.g.

- Hoare paritioning
  -
  -
- Thr
  - Includes a region for values equal to pivot
  - Handles duplicates better
- Median of 3
  - Choose pivot as median of three random eleme
  - Better balance between subproblems

https://eduassistpro.github.i
Add WeChat edu_assist_pr