

Sorting

Assignment Project Exam Help

D Timothy K. Miller

January 2018
<https://powcoder.com>

Add WeChat powcoder



The Sorting Problem

Assignment Project Exam Help

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

Problem (*Sort*)

Input: a sequence A of values $\langle a_1, a_2, \dots, a_N \rangle$

Output: a permutation (reordering) $\langle a'_1, a'_2, \dots, a'_N \rangle$ of A
such that $a'_1 \leq a'_2 \leq \dots \leq a'_N$

- Sorting is an important problem. It is part of the solution to many 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.

a

0	1	2	3	4	5	6	
21	29	7	6	23	50	5	

<https://powcoder.com>

- Proceed from left
- Gradually grow a sorted region (hole loop invariant)

EXERCISE

Invent an incremental sorting algorithm.

Incremental Sorting

Assignment Project Exam Help

a

0	1	2	3	4	5	6	
21	29	7	6	23	50	5	

<https://powcoder.com>

There are two options:

- 1 Add the **next** element $a[i]$ to the sorted region
- 2 Add the **lowest** element outside the sorted region

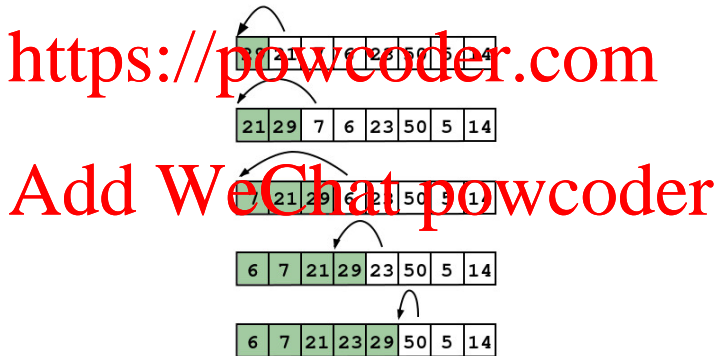
Add WeChat powcoder

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



Insertion Sort

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

```
For  $i = 1$  to  $N$  // initialise invariant
    next =  $a[i]$  // save.  $a[i]$  overwritten later
     $j = i$ 
    While  $j > 0$  and  $next < a[j-1]$  // use invariant
         $a[j] = a[j-1]$ 
         $j = j - 1$ 
    EndWhile
     $a[j] = next$ 
EndFor
```

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

Time Complexity

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



- 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 $T(N) = \Theta(N^2)$

<https://powcoder.com>

Add WeChat powcoder



Worst Case

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

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

$$= \frac{(N-1+1) \times (N-1)}{2} = \frac{N^2 - N}{2}$$



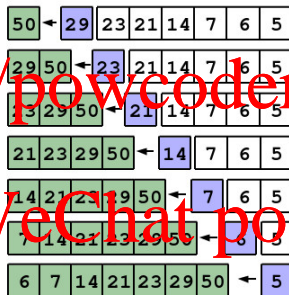
Worst Case

For the worst case time complexity is $aN^2 + bN + c = \Theta(N^2)$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Best Case

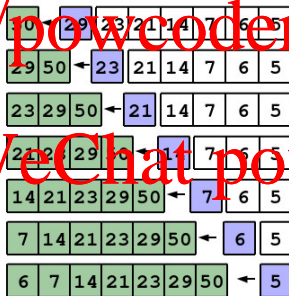
In the best case

- $T(N) = N - 1$

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

<https://powcoder.com>

Add WeChat powcoder

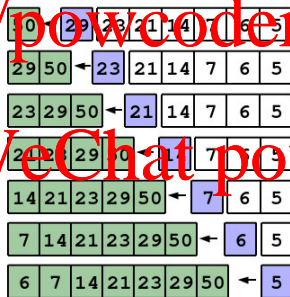


Other Properties

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

<https://powcoder.com>

Add WeChat powcoder



Divide and Conquer

Will a divide and conquer approach work?

Assignment Project Exam Help

29	21	7	6	13	50	5	14
----	----	---	---	----	----	---	----

6	7	21	29
---	---	----	----

5	14	23	50
---	----	----	----

<https://powcoder.com>

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

Add WeChat powcoder

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
while k < r
  if (i >= (m-l)) or (j < (r-m) and right[j] < left[i])
    a[k] = right[j]
    j = j + 1
  else
    a[k] = left[i]
    i = i + 1
  end
  k = k + 1
end

```

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

Divide and Conquer

Assignment Project Exam Help

5	6	7	14	21	23	29	50
---	---	---	----	----	----	----	----

6	7	21	29
---	---	----	----

5	14	23	50
---	----	----	----

<https://powcoder.com>

- Time to combine subproblem solutions is $\Theta(N)$

EXERCISE

Add WeChat powcoder

What is worst case time complexity of divide and conquer algorithm?

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

Time Complexity

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:

$$T(N) = \begin{cases} \Theta(1) & \text{if } N = 1 \\ 2T(N/2) + \Theta(N) & \text{if } N > 1 \end{cases}$$

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

- $f(N) = \Theta(N^{\log_b a}) = \Theta(N^{\log_2 2}) = \Theta(N)$

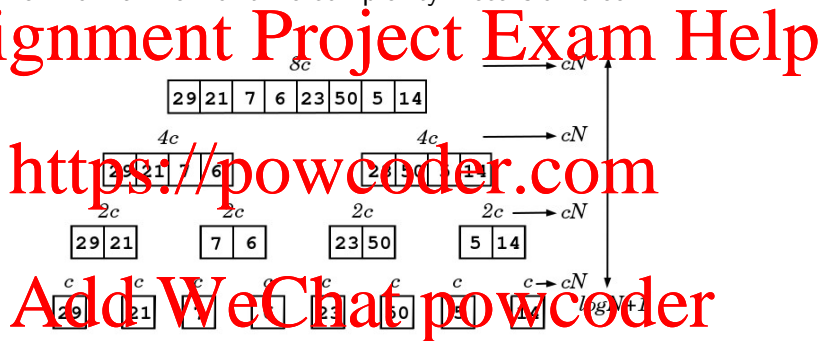
and Case 2, with $k = 0$, applies.

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

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

Time Complexity

Alternative informal view of time complexity: recursion tree



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

MergeSort

You have invented Mergesort

```
MergeSort (input array a, index l, index r)
```

```
    if  $r - l < 2$ 
```

```
        return
```

```
     $m = (l + r) / 2$ 
```

```
    MergeSort(a, l, m)
```

```
    MergeSort(a, m, r)
```

```
    Merge(a, l, m, r)
```

```
    return
```

- 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

- Space complexity is $\Theta(N)$
- Time complexity is $\Theta(N \log N)$
- Faster than Insertion Sort for large, unsorted lists
- Slower than Insertion Sort if the list is already sorted
- Slower than Insertion Sort for small N

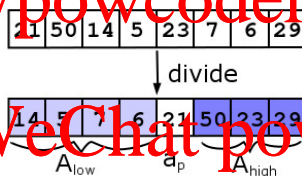
<https://powcoder.com>

Add WeChat powcoder

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 $\forall a \in A_{low}(a < a_p)$ and $\forall a \in A_{high}(a \geq a_p)$

<https://powcoder.com>



Add WeChat powcoder

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

Quicksort

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

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

```
if  $r < l + 2$                 // 0 or 1 elements to sort
    return
 $p = \text{Partition}(a, l, r)$ 
Quicksort( $a, l, p$ )
Quicksort( $a, p + 1, r$ )
return
```

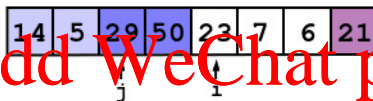
<https://powcoder.com>

Add WeChat powcoder

- 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

- Use last element as the pivot
- Maintain two subarrays which grow
 - Elements before j must be less than pivot
 - Elements $j, \dots, i-1$ must be equal or greater than the pivot
 - Elements i, \dots are unseen so far

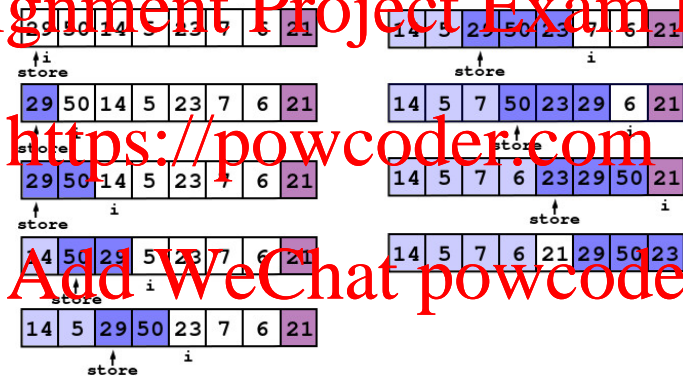


Tutorial Exercise

Write the Partition procedure.

Lomuto Partitioning

Assignment Project Exam Help



<https://powcoder.com>

Add WeChat powcoder

Partition

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

```

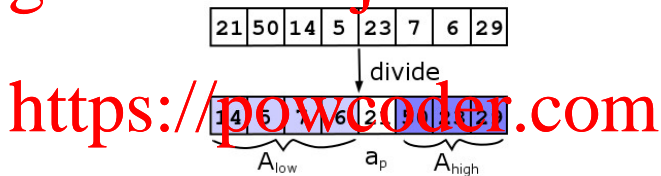
Partition (input: array a, index l, index r)
    i = j = l           // both partitions are empty
    p = r - 1
    while i < p
        if a[i] < a[p]
            swap(a, i, j)
            j = j + 1
        i = i + 1
    swap(a, p, j)
    return j

```

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

Quicksort Performance

Assignment Project Exam Help



Question

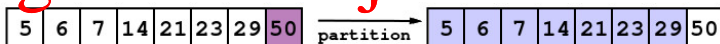
Add WeChat powcoder

What is the **worst case** time complexity of Quicksort. And why?

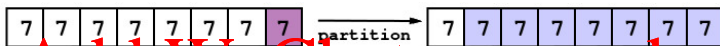
Quicksort Worst Case

The given partition procedure:

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



- will only remove one element from data with many duplicates



This leads to **incremental** execution resembling insertion sort

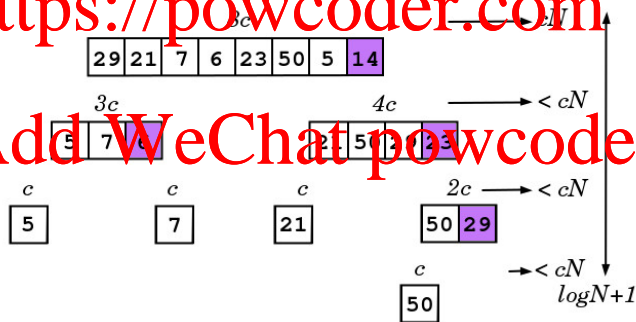
- N levels of recursion
- $N - i$ elements to partition at level i
- So worst case time complexity is $\Theta(N^2)$

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 recursion tree suggests, constants smaller than Mergesort

<https://powcoder.com>

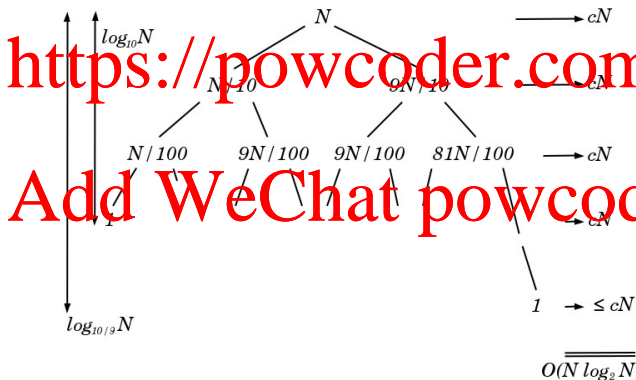
Add WeChat powcoder



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)



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.

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

```
x = random(l, r) // random integer in [l,r-1]
swap(a, x, r-1)
... as before
```

Assuming N distinct values:

- The probability of choosing the worst pivot in every call is $1/N!$
- This becomes vanishingly small as N increases
- Randomised Quicksort is algorithm of choice if N more than ~ 10

Expected Performance

Possible to give **average case** time complexity for Randomised Quicksort:

- Assume N distinct values again
- Randomisation means all inputs equally likely
- Time depends on number of comparisons performed
- Probability of comparing $a[i]$ with $a[j]$ determined by their rank by value (see books for full explanation)
- Average number of comparisons is sum of probabilities for all i, j

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 partitioning
 - Partitions grow inwards from end
 - Handles duplicates better
- Three-way partitioning
 - Includes a region for values equal to pivot
 - Handles duplicates better
- Median of 3
 - Choose pivot as median of three random elements
 - Better balance between subproblems

<https://powcoder.com>

Add WeChat powcoder