Code 2 Learn

Learn, Code, Share

QuickSort Algorithm Tutorial

We have already done tutorial on Merge Sort and a tutorial on Heap Sort (Array Based) with both having a time complexity of O(n*log n). Here is another algorithm which has a time complexity of O(n*log n) and it's called QuickSort.

QuickSort as we all know has a similar approach to Merge Sort i.e. it uses Divide-and-Conquer recursive algorithm to sort the values. The difference being is it's an **in-place** sorting algorithm.

Basically an in-place algorithm is one which transforms the input using a data structure with a small, constant amount of extra storage space.

Basic Idea of QuickSort

1. Pick an element in the array as the pivot element.

2. Make a pass to the array, called the PARTITION step, which rearranges the

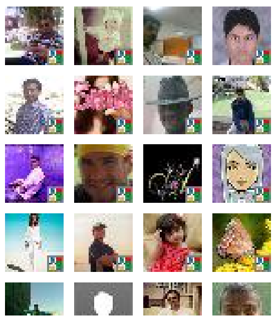
Translate

Select Language ▼

Followers

Join this site
with Google Friend Connect

Members (169) More »



Popular Posts

Java : Encryption and Decryption of Data using AES algorithm with example code

There are many problems when you try encrypting a string such password, credit card nos, phone no. etc ie 1. which algorithm to use. 2. ...

QuickSort Algorithm Tutorial

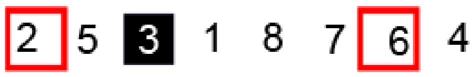
We have already done tutorial on Merge Sort and a tutorial on Heap Sort (Array Based) with both having a time complexity elements in the array:

a. The pivot element is in the proper place

b. The elements less than pivot element are on the left of it

c. The elements greater than pivot element are on the right of it

3. Recursively apply the above process to the left and right part of the pivot element.



Quick Sort Example

Algorithm

Step 1. Choosing the Pivot Element

Choosing the pivot element can determine the complexity of the algorithm i.e. whether it will be n*logn or quadratic time:

a. Normally we choose the first, last or the middle element as pivot. This can harm us badly as the pivot might end up to be the smallest or the largest element, thus leaving one of the partitions empty.

b. We should choose the Median of the first, last and middle elements. If there are N elements, then the ceiling of N/2 is taken as the pivot element.

of O(n*log n). ...

HeapSort (array Based) implementation in Java

There are two types of heaps. First one is Max heap and second one is Min heap. Heap (Max/Min) is a special type of binary tree.The roots o...

Counting Sort Algorithm with Example

After a long interval of time I am writing a post on ALGORITHMS . After writing posts on Heap Sort , Merge Sort and Insertion Sort , I de...

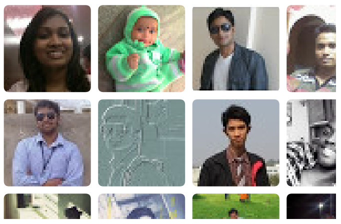
Binary Search Tree (BST) Algorithm Tutorial

Earlier we had a tutorial on Binary Seach Tree Basics , which you can check for refreshing the knowledge about it. Today we will be taking ...

Code2Learn on Google+

Code 2 Learn

Follow



Example:

8, 3, 25, 6, 10, 17, 1, 2, 18, 5

first element: 8
middle element: 10
last element: 5

Therefore the median on [8,10,5] is 8.

Step 2. Partitioning

a. First thing is to get the pivot out of the way and swapping it with the last number.

Example: (shown using the above array elements)

5, 3, 25, 6, 10, 17, 1, 2, 18, **8**

b. Now we want the elements greater than pivot to be on the right side of it and similarly the elements less than pivot to be on the left side of it.

For this we define 2 pointers, namely i and j. i being at the first index and j being and the last index of the array.

* While **i** is less than **j** we keep in incrementing **i** until we find an element greater than pivot.
* Similarly, while **j** is greater then **i** keep decrementing **j** until we find an element less than pivot.
* After both **i** and **j** stop we swap the elements at the indexes of **i** and **j** respectively.

c. Restoring the pivot

When the above steps are done correctly we

304 have us in circles

Tutorials

algorithms
CodeIgniter
data mining
Data Warehouse
excel tutorial
image processing tutorial
javascript tutorial
java tutorial
jdbc tutorial
php tutorial
problems
python tutorial
sql tutorial
Teradata
Unix program
visual basic tutorial

Blog Archive

► 2015 (5)
▼ 2013 (6)
  ► October (1)
  ► May (1)
  ► April (1)
  ▼ February (2)
    QuickSort Algorithm Tutorial
    Binary Search Tree (BST) Algorithm Tutorial
  ► January (1)
► 2012 (44)
► 2011 (67)
► 2010 (9)

When the above steps are done correctly we will get this as our output:

[5, 3, 2, 6, 1] [8] [10, 25, 18, 17]

Step 3. Recursively Sort the left and right part of the pivot.

Java Code

```
public class QuickSortAlgorithm {

    protected static int A[];

    public void sort(int[] A) {
        if (A == null || A.length == 0)
            return;
        quicksort(A, 0, A.length - 1);
    }

    public void quicksort(int[] A, int l
eft, int right) {
        int pivot = A[left + (right - le
ft) / 2];

        int i = left;
        int j = right;
        while (i <= j) {
            while (A[i] < pivot) {
                i++;
            }
            while (A[j] > pivot) {
                j--;
            }
            if (i <= j) {
                exchange(i, j);
                i++;
```

```
                    i++;
                    j--;
                }
            }

            if(left < j)
                quicksort(A,left,j);
            if(i < right)
                quicksort(A,i,right);
        }

        public void exchange(int i, int j){
            int temp=A[i];
            A[i]=A[j];
            A[j]=temp;
        }

        public String toString() {
            String s = "";
            s += "[" + A[0];
            for (int i = 1; i < A.length; i+
    +) {
                s += ", " + A[i];
            }
            s += "]";
            return s;
        }

    }
```

Complexity of QuickSort

Worst Case : O(N^2)

This happens when the pivot is the smallest or the largest element. Then one of the partition is empty and we repeat the recursion for N-1 elements

Best Case: O(NlogN)

This is when the pivot is the median of the array and the left and right part are the of the same size.

There are logN partitions and to compare we do N comparisions

Conclusion :

Advantages

* One of the fastest algorithm on average.

* Doesn't need additional memory i.e. it's an in-place sorting algorithm.

Disadvantages

Worst Case complexity is O(N^2).

Comparision to Merge Sort

* Merge Sort guarantee O(NlogN) time, however it requires additional memory with size N.
* Quick Sort doesn't require additional memory but the running time is not guaranteed.
* Usually Merge Sort is not used for Memory Sorting. Its used only for external memory sorting.

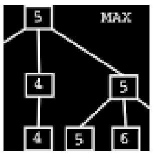
Do leave your comments and let me know if I am wrong somewhere. Thanks!

2/10/2013 FARHAN KHWAJA ALGORITHMS 3 COMMENTS

SHARE THIS POST: Tweet 6 G+1 19 Like 19 Share

0 Share

Related Posts:

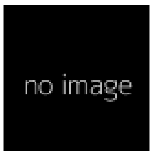


Minimax Algorithm Tutorial

There are plenty of application in AI but games are the most important thing in today's world and nowadays every OS comes with two player games like chess, checkers etc. So there are algorithm that help to design these ga… Read More

Algorithm Tutorials - Part 1 : Ad-Hoc Problems

I will start with the very basic of Algorithms. Before moving on to the different algorithms, let's first see what does an algorithm actually mean? Definition : A process or set of rules to be followed in cal… Read More

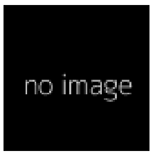


Counting Sort Algorithm with Example

After a long interval of time I am writing a post on ALGORITHMS. After writing posts on Heap Sort, Merge Sort and Insertion Sort, I decided to write on Counting Sort. Counting sort is an algorithm for sorting a collection … Read More

QuickSort Algorithm Tutorial

We have already done tutorial on Merge Sort and a tutorial on Heap Sort (Array Based) with both having a time complexity of O(n*log n). Here is another algorithm which has a time complexity of O(n*log n) and it's called Qui… Read More



Binary Search Tree (BST) Algorithm Tutorial

Earlier we had a tutorial on Binary Seach Tree Basics, which you can check for refreshing the knowledge about it. Today we will be taking a look on BST algorithm and implementing it using Java. Binary Search Tree is node ba… Read More

← Newer Post Home Older Post →

3 comments:

Anonymous January 6, 2015 at 11:08 PM

Small correction:

```
public void sort(int[] A) {
  if (A == null || A.length == 0)
   return;
  quicksort(A, 0, A.length - 1);
}
```

Reply

Replies

farhan khwaja January 7, 2015 at 10:47 AM

Well the same is mentioned on the post!!

Reply

Daniel Moisset July 27, 2015 at 2:24 PM

I made a visualization on the walnut that makes it a bit more clear: https://thewalnut.io/visualizer/visualize/3474/334/

You can also see explain graphically how the choice of pivot affects running time: Compare the call tree at https://thewalnut.io/visualizer/visualize/3474/426/#time=41 vs https://thewalnut.io/visualizer/visualize/3474/426/#time=41

Reply

Enter your comment...

Comment as: VISHWARUP ▼ Sign out

Publish Preview ☐ Notify me