GeeksforGeeks

A computer science portal for geeks

Practice IDE Q&A GeeksQuiz

Nuts & Bolts Problem (Lock & Key problem)

Given a set of n nuts of different sizes and n bolts of different sizes. There is a one-one mapping between nuts and bolts. Match nuts and bolts efficiently.
Constraint: Comparison of a nut to another nut or a bolt to another bolt is not allowed. It means nut can only be compared with bolt and bolt can only be compared with nut to see which one is bigger/smaller.

Other way of asking this problem is, given a box with locks and keys where one lock can be opened by one key in the box. We need to match the pair.

Brute force Way: Start with the first bolt and compare it with each nut until we find a match. In the worst case we require n comparisons. Doing this for all bolts gives us O(n^2) complexity.

Quick Sort Way: We can use quick sort technique to solve this. We represent nuts and bolts in character array for understanding the logic.

Nuts represented as array of character
char nuts[] = {‘@’, ‘#’, ‘$’, ‘%’, ‘^’, ‘&’}

Bolts represented as array of character
char bolts[] = {‘$’, ‘%’, ‘&’, ‘^’, ‘@’, ‘#’}

This algorithm first performs a partition by picking last element of bolts array as pivot, rearrange the array of nuts and returns the partition index ‘i’ such that all nuts smaller than nuts[i] are on the left side and all nuts greater than nuts[i] are on the right side. Next using the nuts[i] we can partition the array of bolts. Partitioning operations can easily be implemented in O(n). This operation also makes nuts and bolts array nicely partitioned. Now we apply this partitioning recursively on the left and right sub-array of nuts and bolts.

As we apply partitioning on nuts and bolts both so the total time complexity will be Θ(2*nlogn) = Θ(nlogn) on average.

Here for the sake of simplicity we have chosen last element always as pivot. We can do randomized quick sort too.

A Java based implementation of idea is below:

```
// Java program to solve nut and bolt problem using Quick Sort
public class NutsAndBoltsMatch
```

```
{
    //Driver method
    public static void main(String[] args)
    {
        // Nuts and bolts are represented as array of characters
        char nuts[] = {'@', '#', '$', '%', '^', '&'};
        char bolts[] = {'$', '%', '&', '^', '@', '#'};

        // Method based on quick sort which matches nuts and bolts
        matchPairs(nuts, bolts, 0, 5);

        System.out.println("Matched nuts and bolts are : ");
        printArray(nuts);
        printArray(bolts);
    }

    // Method to print the array
    private static void printArray(char[] arr) {
        for (char ch : arr){
            System.out.print(ch + " ");
        }
        System.out.print("\n");
    }

    // Method which works just like quick sort
    private static void matchPairs(char[] nuts, char[] bolts, int low,
                                                              int high)
    {
        if (low < high)
        {
            // Choose last character of bolts array for nuts partition.
            int pivot = partition(nuts, low, high, bolts[high]);

            // Now using the partition of nuts choose that for bolts
            // partition.
            partition(bolts, low, high, nuts[pivot]);

            // Recur for [low...pivot-1] & [pivot+1...high] for nuts and
            // bolts array.
            matchPairs(nuts, bolts, low, pivot-1);
            matchPairs(nuts, bolts, pivot+1, high);
        }
    }

    // Similar to standard partition method. Here we pass the pivot element
    // too instead of choosing it inside the method.
    private static int partition(char[] arr, int low, int high, char pivot)
    {
        int i = low;
        char temp1, temp2;
        for (int j = low; j < high; j++)
        {
            if (arr[j] < pivot){
                temp1 = arr[i];
                arr[i] = arr[j];
                arr[j] = temp1;
                i++;
            } else if(arr[j] == pivot){
                temp1 = arr[j];
                arr[j] = arr[high];
                arr[high] = temp1;
                j--;
            }
        }
        temp2 = arr[i];
        arr[i] = arr[high];
        arr[high] = temp2;

        // Return the partition index of an array based on the pivot
        // element of other array.
        return i;
    }
}
```

Run on IDE

Output:

```
Matched nuts and bolts are :
# $ % & @ ^
# $ % & @ ^
```

This article is contributed by **Kumar Gautam**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

37 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1’s is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

3.3 Average Difficulty : **3.3/5.0**
Based on **3** vote(s)

□ Add to TODO List
□ Mark as DONE

Like Share 24 people like this. Be the first of your friends.

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