**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**RV COLLEGE OF ENGINEERING, BANGALORE**

**DESIGN AND ANALYSIS OF ALGORITHMS (18CS42)**

**LABORATORY PART-B COMPONENT**

**TOPIC:**

**IMPLEMENTATION OF NUTS AND BOLTS PROBLEM IN n\*log n TIME**

**NAME: GIRIDHAR K. SHANBHAG**

**SEMESTER: 4TH**

**SECTION: B-1**

**USN: 1RV18CS056**

**IMPLEMENTATION OF NUTS AND BOLTS PROBLEM IN n\*log n TIME**

1. **OVERVIEW OF THE NUTS AND BOLTS PROBLEM:**

The basic idea of the nuts and bolts problem is that we have been given n number of nuts and number of bolts and the task is to match the nuts with the corresponding bolts.

These nuts and bolts can be represented in the form of characters or whole numbers/ integers.

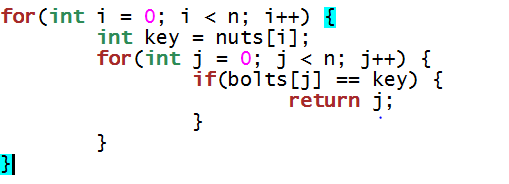
1. **AIM OF THE TASK:**

The nuts and the bolts are given to us in the form of two arrays corresponding to each. The goal is to match the elements of the array nuts and the elements of array bolts in asymptotic time of the order of n\*log(n), where n is the number of the nuts/bolts.

1. **NAÏVE APPROACH TO THE PROBLEM:**

The simplest method to implement the solution is to find out the index in the bolt array corresponding to every element in the nuts array.

The loop structure will be like:



This approach will give us a solution to the problem that can be implemented in an asymptotic time of the order of n squared i.e., O(n2).

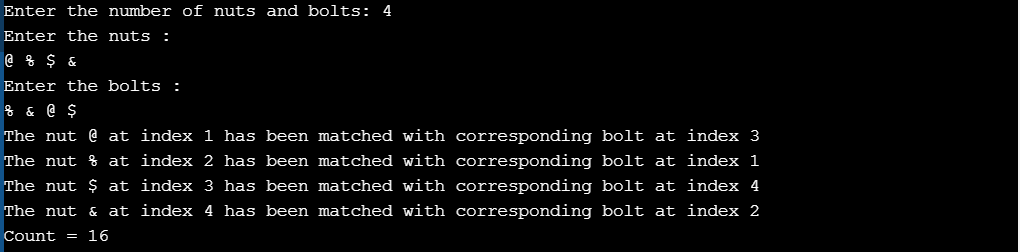
Writing a brute force solution always gives us an idea as to how we can optimize it.

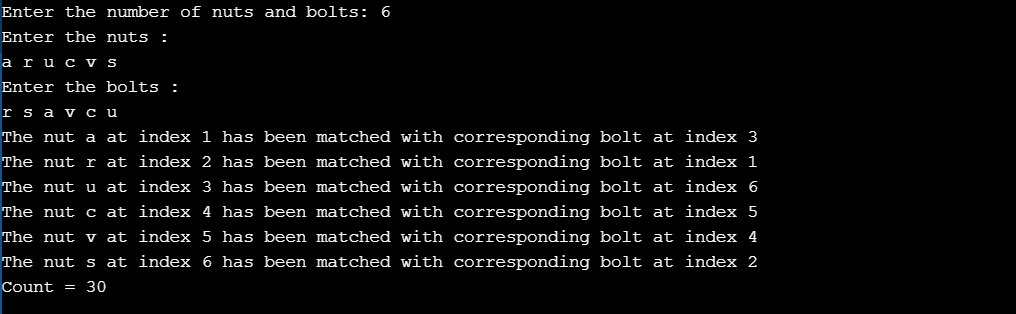
1. **OPTIMISED SOLUTION (n\*logn SOLUTION) IMPLEMENTATION:**

In the method adopted as above we understand that for every index from 0 till n-1 (total of n indices), we set the element of the nuts array at that particular index as our key and look for it in the bolts array through linear search. Linear search takes time of the order O(n) and we perform this task for all n elements thereby causing a complexity of O(n2). To make this number better we need to adopt a technique that increases the efficiency of linear search as we need to run the loop or the algorithm for all n elements of the array. Hence we resort to binary search algorithm which performs each search in time of the order of log n i.e., O(log n). Therefore the task of searching all elements of the nuts array in the bolts array takes a total time of O(n\*log n).

But for binary search to be implemented on the bolts array it is necessary that the elements are in a sorted order. Keeping in mind the fact that the sorting algorithm should also have a total time complexity of O(n\*log n), we can resort to mergesort as for all cases of input, best, worst and average it runs in n\*log n time.

1. **OUTPUT SNAPSHOTS:**

****

****