Algorithm To Find A Random Number

Vinicios Barretos Batista
Instituto de Biociências, Letras e Ciências Exatas, Unesp -
Univ Estadual Paulista (São Paulo State University) Rua Cristóvão
Colombo 2265, Jd Nazareth, 15054-000, São José do Rio Preto - SP,
Brazil. e-mail: vinicios.barretos@unesp.br

March 18, 2018

Abstract

In this article will be introduced a algorithm that discovers a random number generated by the computer based on Binary Search.

1 Discussion

Basically, the idea is build a code that find a random number generated from 1 to 10 with the least number of attempts. For this is necessary a method better than simply checking number by number with if/else commands, especially because this is not applicable in cases with many elements.

2 Method

The idea to use Binary Search has come by your performance, simplicity and by the fact that sequence already is sorted, since than computer will generate an integer between 1 and 10. By the way, this algorithm only works in sorted sequences.

Binary search compares the target value to the middle element of the sequence, if they are unequal, the half in which the target cannot lie is eliminated and the search continues on the remaining half until it is successful. If the search ends with the remaining half being empty, the target is not in the sequence. If the target value matches the middle element, the search is stopped and the value finded.

Basically, Binary Search has the following procedure:

1. Set L to 0 and R to n (Number of elements).
2. If L >R, the search terminates as unsuccessful.
3. Set m (the position of the middle element) to (L + R)/2.
4. If m <T (Target Element), set L to m + 1 and go to step 2.
5. If m >T (Target Element), set R to m - 1 and go to step 2.
6. Now m = T, the search is done;

The programming language used in the following code is Java, and to generate random numbers is necessry import the java.util.Random Class.

```java
import java.util.Random;

public class random {
   public static void main (String args[]) {

   Random randomNumber = new Random();
   int attempt=0, low=1, up=10, guess;
   int num = randomNumber.nextInt(10) + 1;

   while (low <= up) {
      attempt++;
      guess = (low + up) / 2;
      if (guess == num)
         break;
    if (guess > num)
         up = guess-1;
    else
         low = guess+1;
   }

   System.out.println("System found the number " + num + " in " +
       attempt + " attempts");

   System.exit(0);
   }
}
```

The number of iterations in the worst case is the one of equation 1.

$$\log_2(n) + 1 \tag{1}$$

In the best case, this code will find the number 5 in only one attempt. And in the worst case, the number 10 will be found in 4 attempts.

3 Conclusion

The method applied in this article makes much less comparisons than checking number by number, and can be used in applications with many more elements.

References

[1] https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.html
Last view: 18 March 2018.

[2] https://en.wikipedia.org/wiki/Binary_search_algorithm
Last view: 18 March 2018.