

Problem6: BinarySearch

Akira MATSUI

March 27, 2017

Introduction

In this exercise, we will learn binary search algorithm, which is to find a specific number in a sorted list. To use this search algorithm, you have to make sure that your list is ordered numerically. This exercise is related to bubble sort exercise.

If you have a number A that you need to find out where it is in a list, binary search algorithm tell you that in following steps. Suppose the list has N elements.

- compare A with the mean number R_m , of the list ($m := N/2$)
- if $A > R_m$, compare A with R_{m_2} ($m_2 := 3N/4$)
- if $A < R_m$, compare A with R_{m_2} ($m_2 := N/4$)

Here is the simple explanation.

There is a list that has numbers. Suppose that you want to find where $T = 9$ is, but cannot find it at a glance. This might happens if you list has a hundreds of elements.

(1,2,3,6,7,8,9)

First Step

Compare T with 6. In this case, $T > 6$ then go to next step.

Divide the list into half,

(1,2,3), (7,8,9).

Because $T > 6$, we are interested in the latter list (7,8,9).

Second Step

You have

(7,8,9).

Compare T with 8. In this case, $T > 8$ then we have found where $T = 9$ is in the list.

Question

- you have `BS <- sort(as.integer(runif(100, min = 1, max = 99)))`
- Implement `binary search` find where T is in `BS`
- In this excersise, set `T <- as.integer(runif(1, min = 1, max = 99).`
- In the case `BS` dose not have the same number as T , `print('can't find')`

Sample Answer

```
binary <- function(BS){  
  
  N <- length(BS) #Number of factor  
  Start <- 1  
  End   <- N  
  
  while(Start<End){  
    Mid <- as.integer((Start + End)/2)  
    if(BS[Mid] == T){  
      return(Mid)  
    }else if(T < BS[Mid]){  
      End <- Mid  
    }else{  
      Start <- Mid + 1  
    }  
  }  
  print('Not found')  
}  
  
BS <- sort(as.integer( runif(100, min = 1, max = 99) ))  
T <- as.integer( runif(1, min = 1, max = 99) )  
  
print(T)  
  
## [1] 86  
BS[binary(BS)]  
  
## [1] "Not found"  
## [1] NA  
print(BS)  
  
## [1] 1 2 4 9 9 9 9 10 10 10 10 11 12 12 15 19 19 19 23 27 27 28 28  
## [24] 28 30 30 30 31 31 32 33 35 36 37 38 40 41 42 43 45 45 45 47 49 49 50  
## [47] 52 52 52 53 54 54 55 56 59 59 60 61 61 62 62 63 63 63 63 64 65 67 67  
## [70] 68 68 69 69 71 72 75 76 78 78 78 80 80 80 80 80 81 81 84 88 89 90 90  
## [93] 91 91 93 95 96 97 97 98
```