Assignment No. A1

Roll no. 4202

1 Title

Binary Search using Divide and Conquer.

2 Problem Definition

Using Divide and Conquer Strategies design a function for Binary Search using Scala.

3 Objective

- To implement binary search using divide and conquer strategy.
- To learn Scala programming.

4 Pre-requisite

- Basic knowledge of object oriented programming and functional programming.
- Knowledge of divide and conquer strategy.
- Knowledge about the binary search algorithm.

5 Software and Hardware Requirements

1. 64 bit Machine i3/i5/i7
2. 64-bit open source Linux OS Fedora 20
3. Scala
4. Eclipse

Theory

Divide and Conquer Algorithm:

- Dynamic Programming is typically applied to optimization problem. In this dynamic programming, the word programming stands for planning and it does not mean by computer programming.
- Dynamic programming is a technique for solving problem with overlapping subproblems.
- In this method each subproblem is solved only once. The result of each subproblem is recorded in the table from which we can obtain a solution to the original problem.
  For any given problem, we may get any number of solutions we seek for optimal solution (i.e. minimum value or maximum value solution) and such an optimal solution becomes the solution to the given problem.
- Dynamic programming design involves four major steps :
  1. Characterize the structure of optimal solution. That means develop a mathematical notation that can express any solution and subsolution for the given problem.
  2. Recursively define the value of an optimal solution.
  3. By using bottom up technique compute value of optimal solution. For that you have to develop a recurrence relation that a solution to its subsolution using the mathematical notation.
  4. Compute an optimal solution from computed information.

Binary Search Algorithm:

- A binary search or half-interval search algorithm finds the position of a specified value (the input key) within a sorted array.
- In each step, the algorithm compares the input key value with the key value of the middle element of the array. If the keys match, then a matching element has been found so its index, or position, is returned. Otherwise, if the sought key is less than the middle elements key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the input key is greater, on the sub-array to the right.
- If the remaining array to be searched is reduced to zero, then the key cannot be found in the array and a specialNot found indication is returned.
- Every iteration eliminates half of the remaining possibilities. This makes binary searches very efficient - even for large collections.
- Binary search requires a sorted collection. Also, binary searching can only be applied to a collection that allows random access (indexing).

  Worst case performance: O(log n)
  Best case performance: O(1)

Algorithm:

1. Start
2. Create sorted list l
3. Accept key
4. binarysearch(l,key,low,high)
```
     begin
     Calculate mid as (low+high)/2
     if n < l[mid]
         begin
         high = mid  1
         binarysearch(l,key,low,high)
     end
     if key > l[mid]
         begin
         low = mid + 1
         binarysearch(l,key,low,high)
     end
     if key = l[mid]
         print found
```
5. Stop

   where, key is the element to be searched and low and high the lower and upper bounds of the list

Scala:

- Scala is an acronym for Scalable Language.
- Scala is a modern multi-paradigm programming language designed to express common programming patterns in a concise, elegant, and type-safe way.
- Scala smoothly integrates features of object-oriented and functional languages.
- Scala is a pure object-oriented language in the sense that every value is an object. Types and behavior of objects are described by classes.
  Classes are extended by subclassing and a flexible mixin-based composition mechanism as a clean replacement for multiple inheritance.
- Scala is also a functional language in the sense that every function is a value and because every value is an object so ultimately every function is an object.

6 Mathematical Model :

Let S be the solution perspective of the system such that,
S =$\{S_t, E_t,$I,O,DD,NDD,F_{me},Sc,Fc$\}$
where,

S_t = start state list should be sorted.

I represents set of input
I = $\{$key, sorted array$\}$
key $\in$ I+
sorted array $\in$ I+

O represents set of output
O = status
where, status=found or not found

F_{me}=set of functions.
F_{me}=$\{$f1,f2,f3$\}$
where,
f1= f1 represents the function to read the input.
f2= f2 represents the function to perform binary search.
f3= f3 to display the result.

E_t=display result.

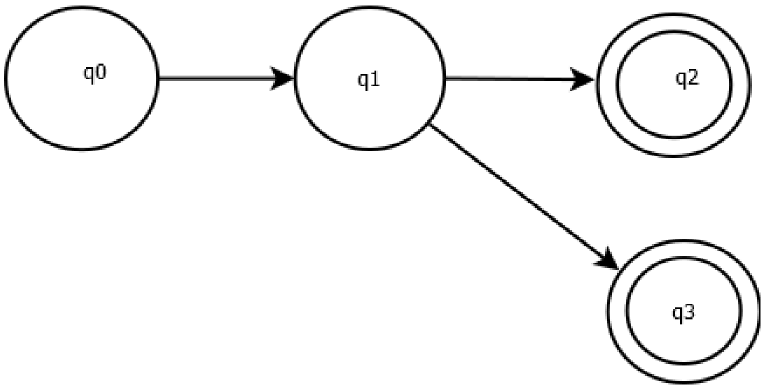
DD = Deterministic Data
= list

NDD (Non Deterministic Data)
= the key which may or may not be present in the list

Sc = status=found if key in list else status=not found

Fc = status=found if key not in list else status=not found

7 State Diagram :



q0 → start state. Accept the key and list

q1 → perform binary search on the list

q2 → display the status if key found.

q3 → display the status if key not found.

8 Conclusion

We have thus successfully implemented binary search Algorithm using Scala programming and understood divide and conquer strategy .