Practice 2.1: Paradigm
Artificial Intelligence

G.Guérard

Divide and conquer

We want to search a value in a sorted list. An obvious approach is to use a loop until we find the value. Here, time depends on the size of the list, or from where we start the loop. The idea behind the divide&conquer's search algorithm is similar to the game "find my number" when this one is between 1 and 100. The idea is to start at 50, and then you know if you are over or under the secret value. Depending of the response, you will continue to cut by half the feasible set. Propose a divide&conquer's algorithm to search a value in a sorted list.

A binary search begins by comparing the middle element of the array with the target value. If the target value matches the middle element, its position in the array is returned. If the target value is less or more than the middle element, the search continues in the lower or upper half of the array with a new middle element, eliminating the other half from consideration.

Given an array A of n elements with values or records $A_0 \ldots A_{n-1}$ and target value T, the following subroutine uses binary search to find the index of T in A.

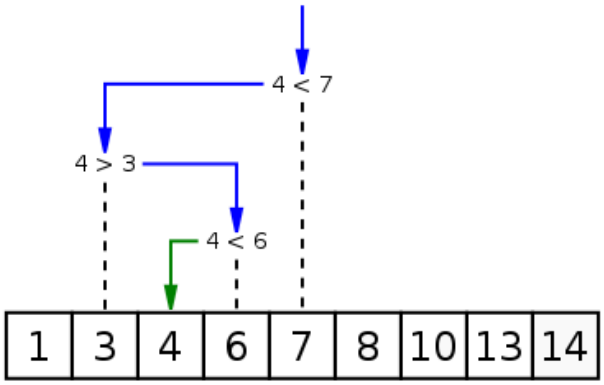
Set L to 0 and R to $n-1$.

If $L > R$, the search terminates as unsuccessful. Set m to the floor of $(L + R) / 2$.

If $A_m = T$, the search is done; return m.

If $A_m < T$, set L to $m + 1$ and go to step 2.

If $A_m > T$, set R to $m - 1$ and go to step 2.



We deduce the divide&conquer's algorithm:

Binary_search(list,T,n)

If(T==null) return null

Pivot ← n/2

If(list[Pivot]==T return Pivot

Elseif list[Pivot]>T return Binary_search(list[1 … Pivot-1],T,Pivot-1)

Else return Binary_search(list[n-Pivot…n],T,n)

In this algorithm, we only use a part of the list, it is also possible to use the left bound L and right bound R with the full list.