* Descrierea programului:

Abordare Divide et Impera pentru rezolvarea problemei 1, var. 3;

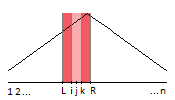
Complexitate – *O(log n)*, unde *n* este lungimea secvenței de intrare.

* Descrierea funcției recursive:

Considerăm parametrii funcției *L* și *R* (cel mai din stânga, respectiv cel mai din dreapta index al razei de căutare) și variabilele locale 3 numere naturale consecutive *i*, *j*, *k* cu *j = ⌊(L+R)/2⌋* ;

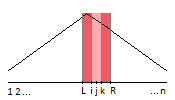
* Cazuri de continuare:

1. *v[i]<v[j]<v[k]* :



În acest caz, vârful vectorului sigur se află în a doua jumătate a razei de căutare; se reapelează funcția cu parametrii *L/2*, respectiv *R*;

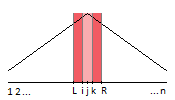
1. *v[i]>v[j]>v[k]* :



În acest caz, vârful vectorului sigur se află în prima jumătate a razei de căutare; se reapelează funcția cu parametrii *L*, respectiv *R/2*;

* Cazul de oprire:

*v[i]<v[j]>v[k]* :



În acest caz, vârful vectorului este elementul de pe poziția *j*.

* Demonstrația complexității:

Folosind Teorema Master: *T(n) = aT(n/b) + f(n)*

În particular, pentru acest algoritm:

*a = 1* : se apelează funcția *search(int,int)* o singură dată în ea însăși;

*b = 2* : raza de căutare devine de două ori mai restrânsă cu fiecare apelare recursivă a funcției *search(int,int)* și, în particular, când *L = R – 2*, algoritmul sigur se încheie (pentru că vârful este elementul dintre L și R);

*f(n) = Θ(1)* : în funcția *search(int,int)* se mai apeleazăfuncția de comparare: O(1);

Rezultă din Teorema Master *T(n) = O(log n)*.