Titolo unità didattica: Approccio divide et impera [13]

Titolo modulo : Idea di base del “divide et impera” [01-T]

L’epitome del divide et impera: algoritmo di ricerca binaria

Argomenti trattati:

- ✓ divide et impera e sequenze di istanze “più semplici”
- ✓ istanza banale
- ✓ divide et impera per la ricerca in array ordinati: algoritmo di ricerca binaria
- ✓ algoritmo di ricerca binaria

Prerequisiti richiesti: AP-07-03-T, AP-12-01-T

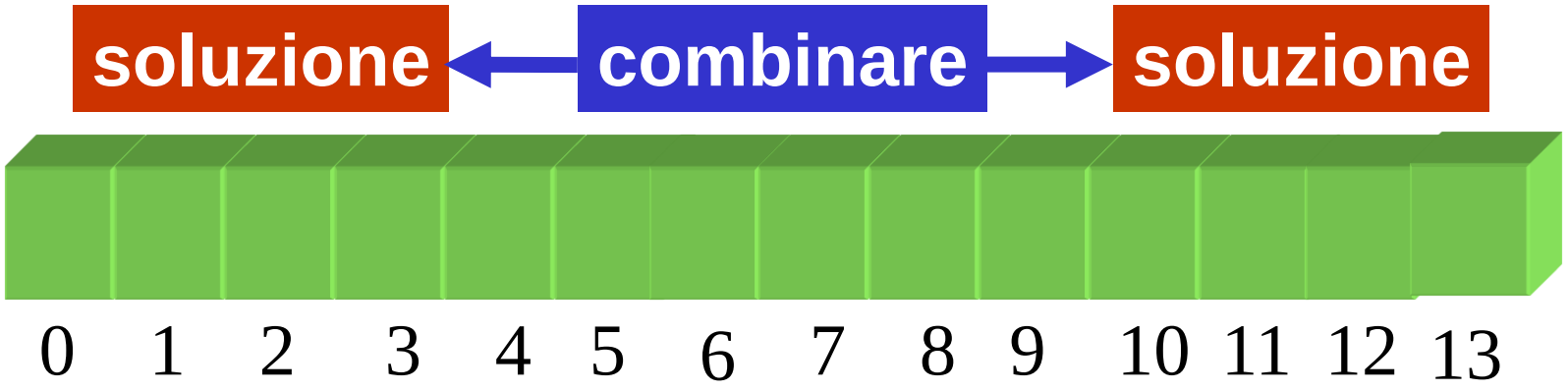
idea!

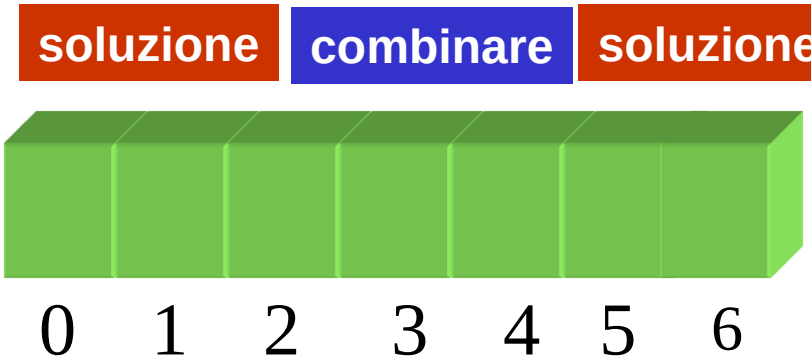
partendo dell'istanza da risolvere, si genera una **sequenza di istanze** via via più **semplici** del problema, fino all'istanza che non è più suddivisibile e che ha **soluzione banale**

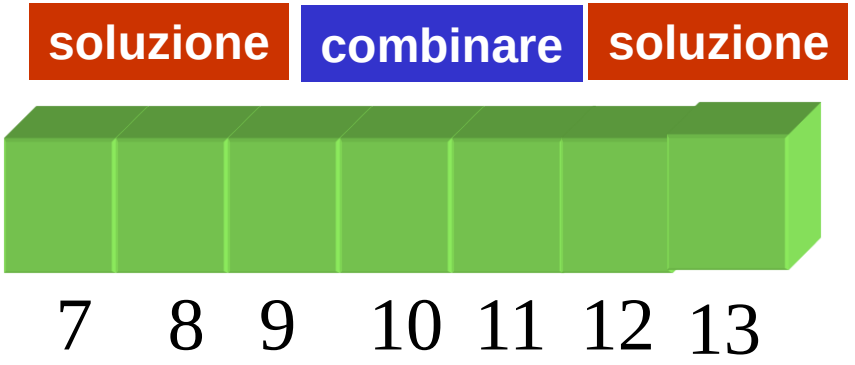
approccio divide et impera
(divide and conquer)

a ogni passo, la **soluzione dell'istanza** da risolvere viene espressa in termini delle **soluzioni delle istanze più semplici** in cui essa è **decomposta**

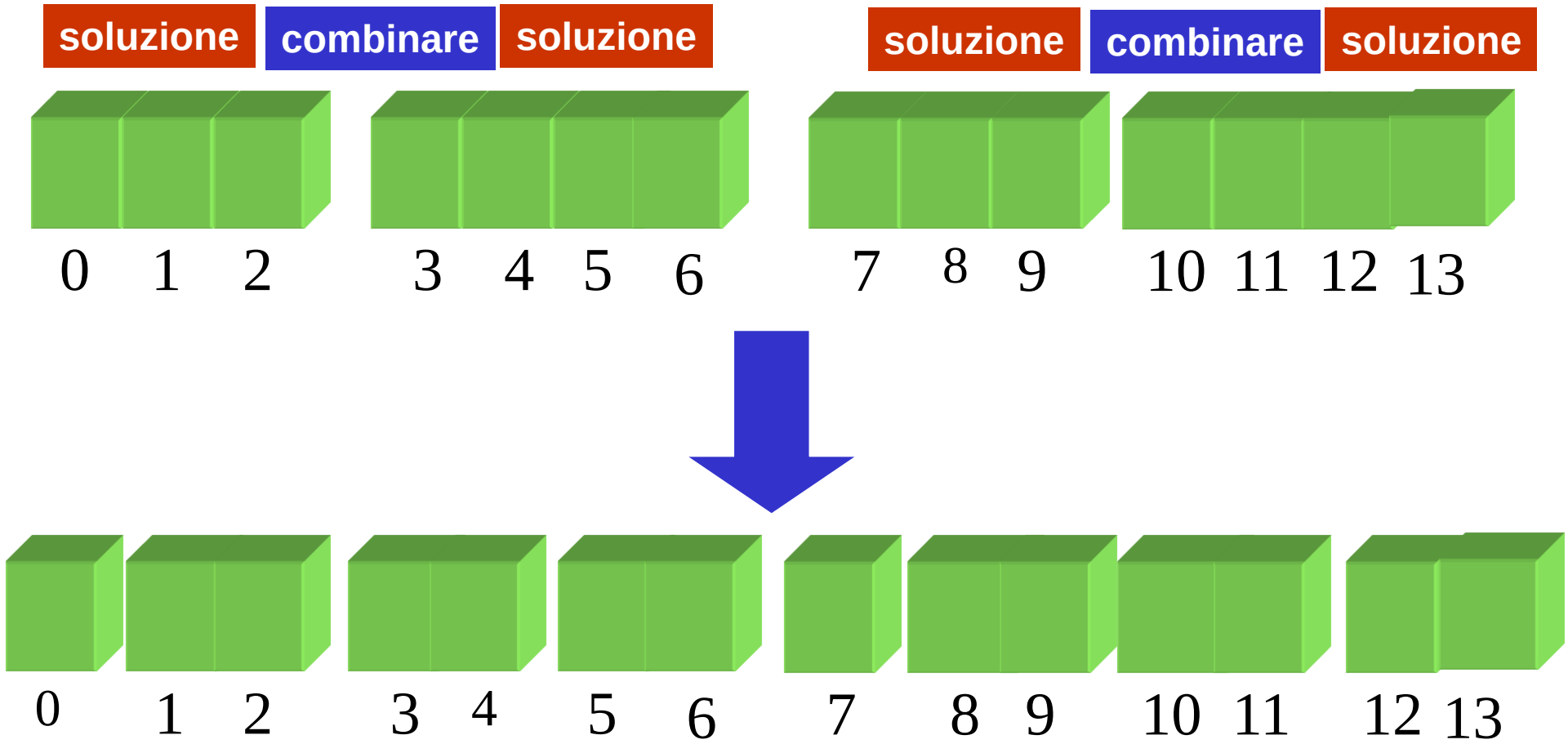
idea approccio divide et impera



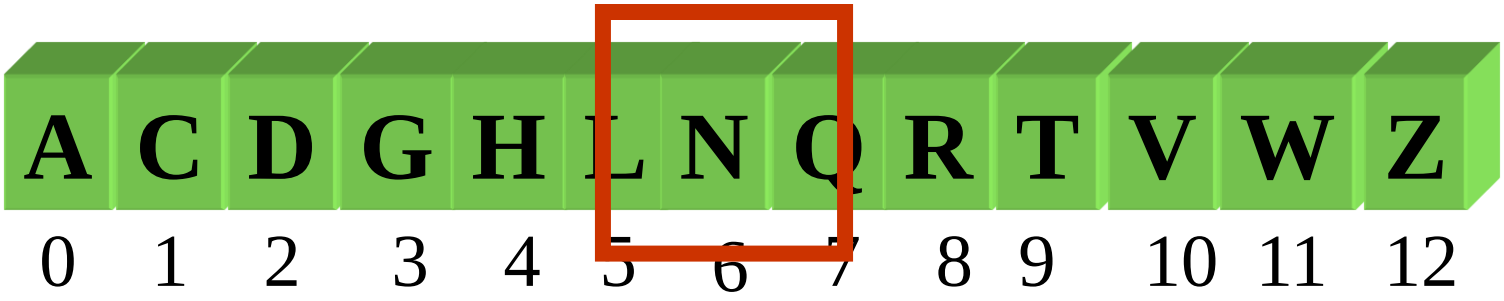




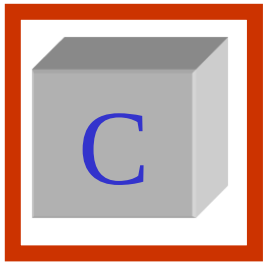
idea approccio divide et impera



ricerca in un array **ordinato** (**ricerca binaria**)

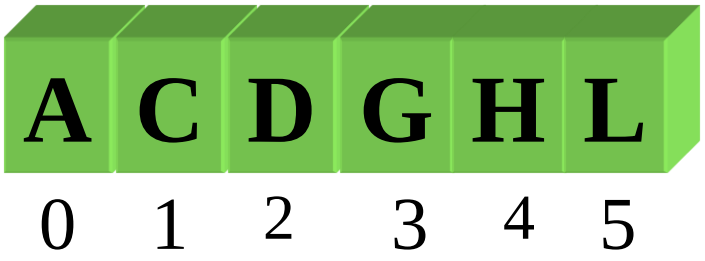


valore da ricercare (**chiave**)

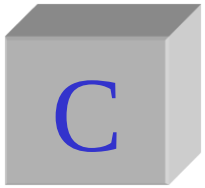


$C \neq N$ ⇒ continuare la ricerca

ricerca in un array **ordinato** (**ricerca binaria**)

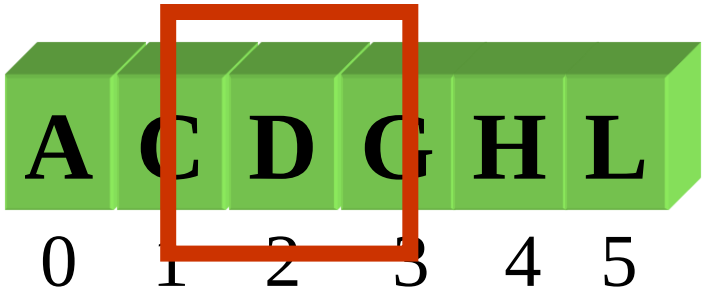


valore da ricercare (**chiave**)

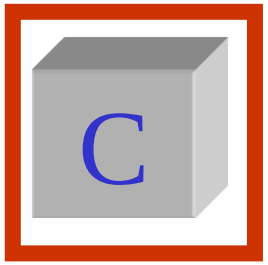


C < N ⇒ C < Q,R,S,T,V,W,Z

ricerca in un array **ordinato** (**ricerca binaria)**

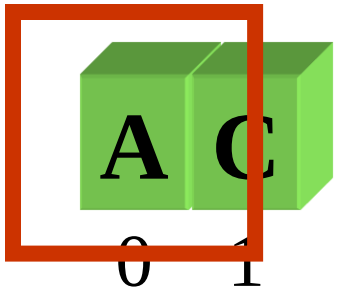


valore da ricercare (**chiave**)

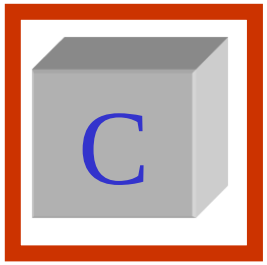


C < D ⇒ C < G,H,L

ricerca in un array **ordinato** (**ricerca binaria)**

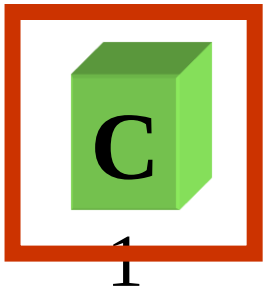


valore da ricercare (**chiave**)

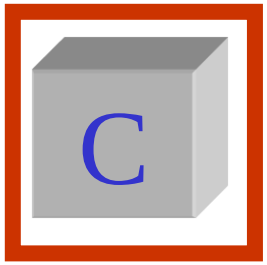


C > A

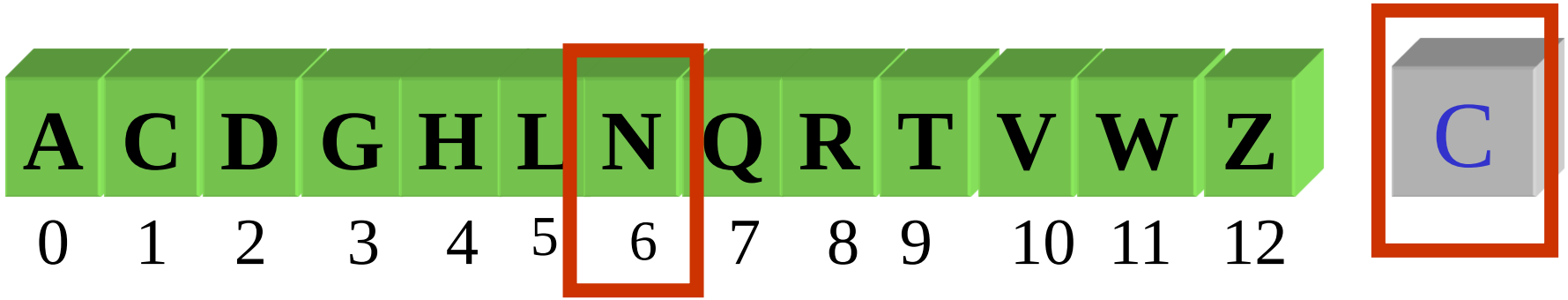
ricerca in un array **ordinato** (**ricerca binaria)**



valore da ricercare (**chiave**)



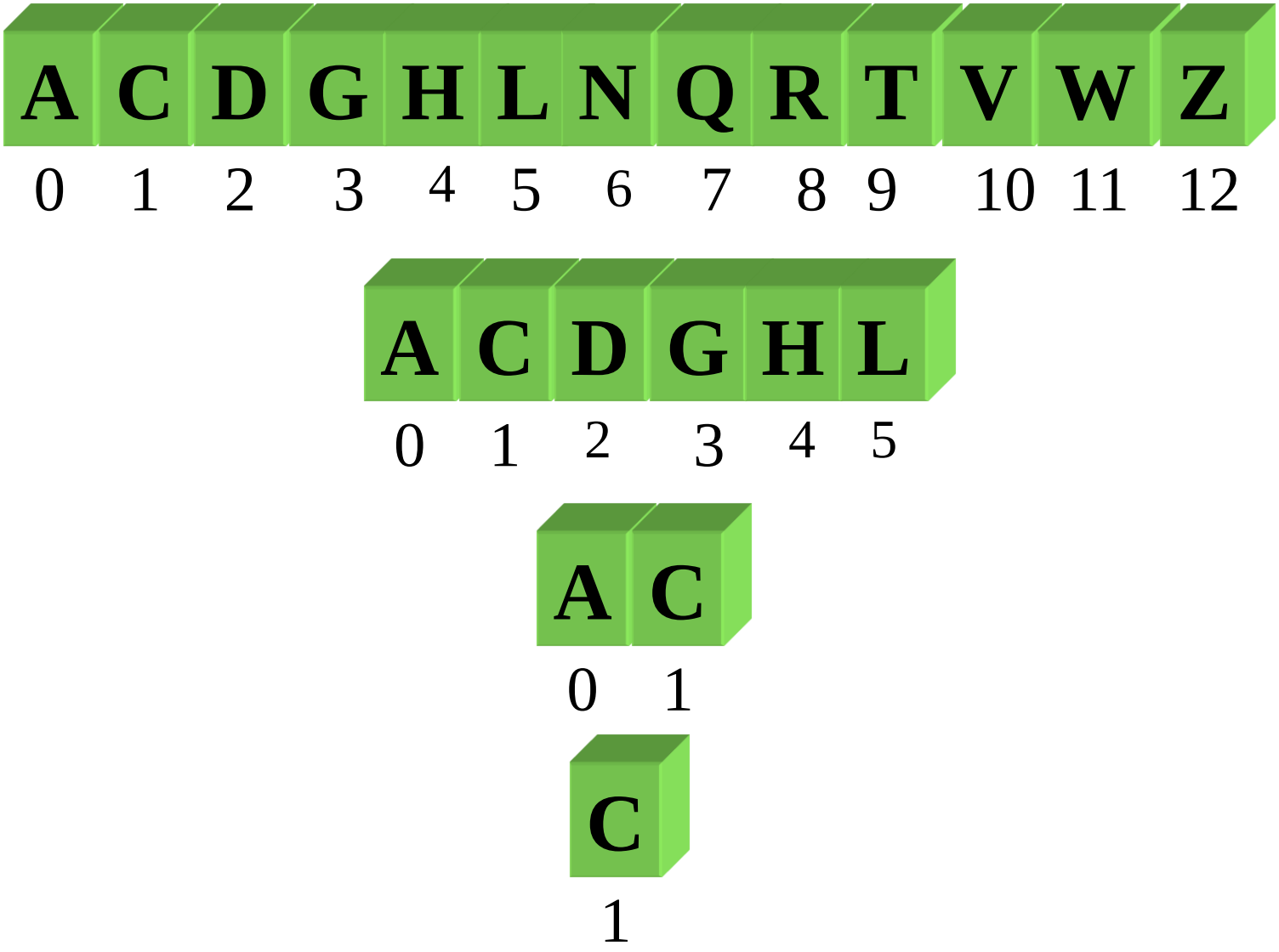
ricerca in un array **ordinato** (**ricerca binaria**)

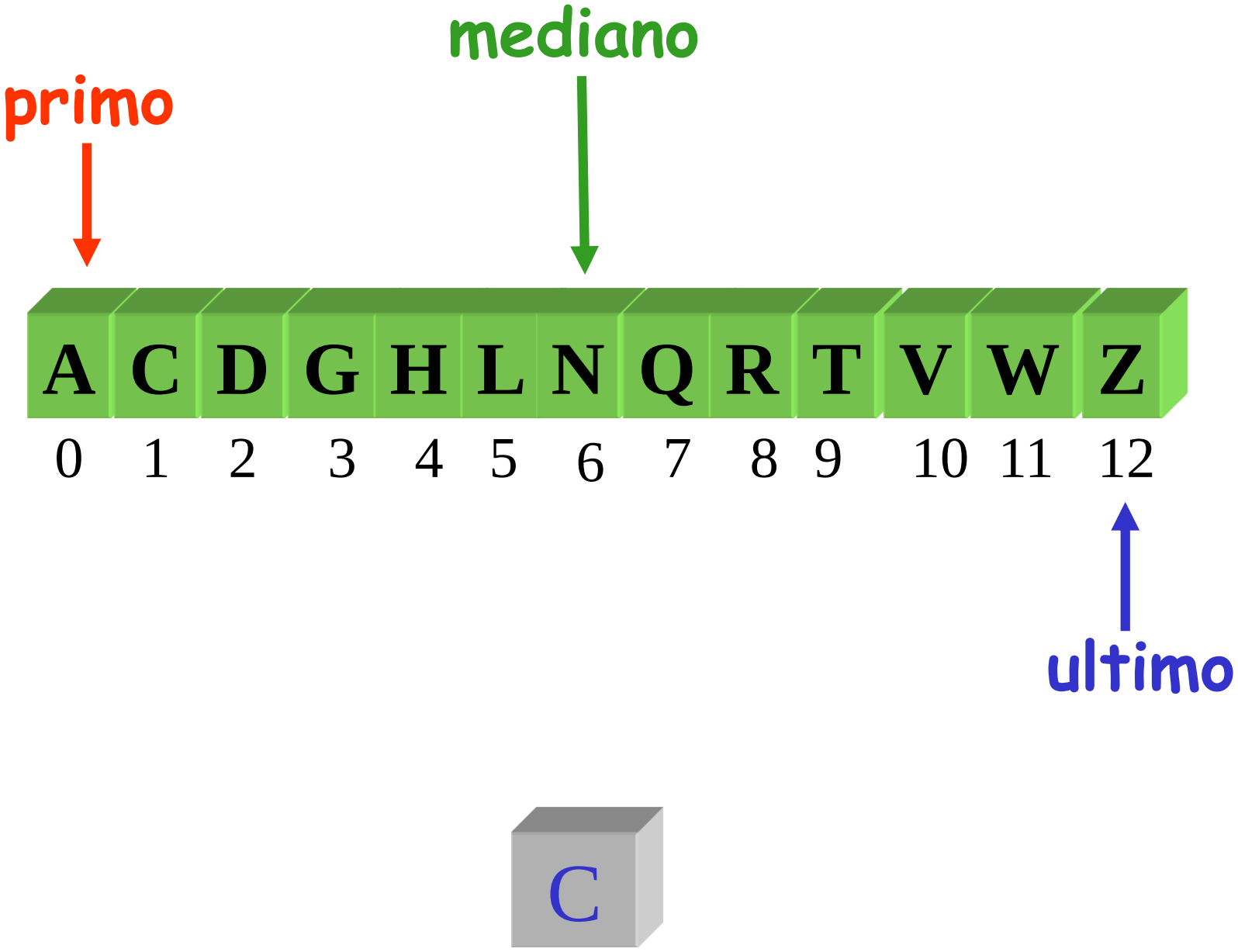


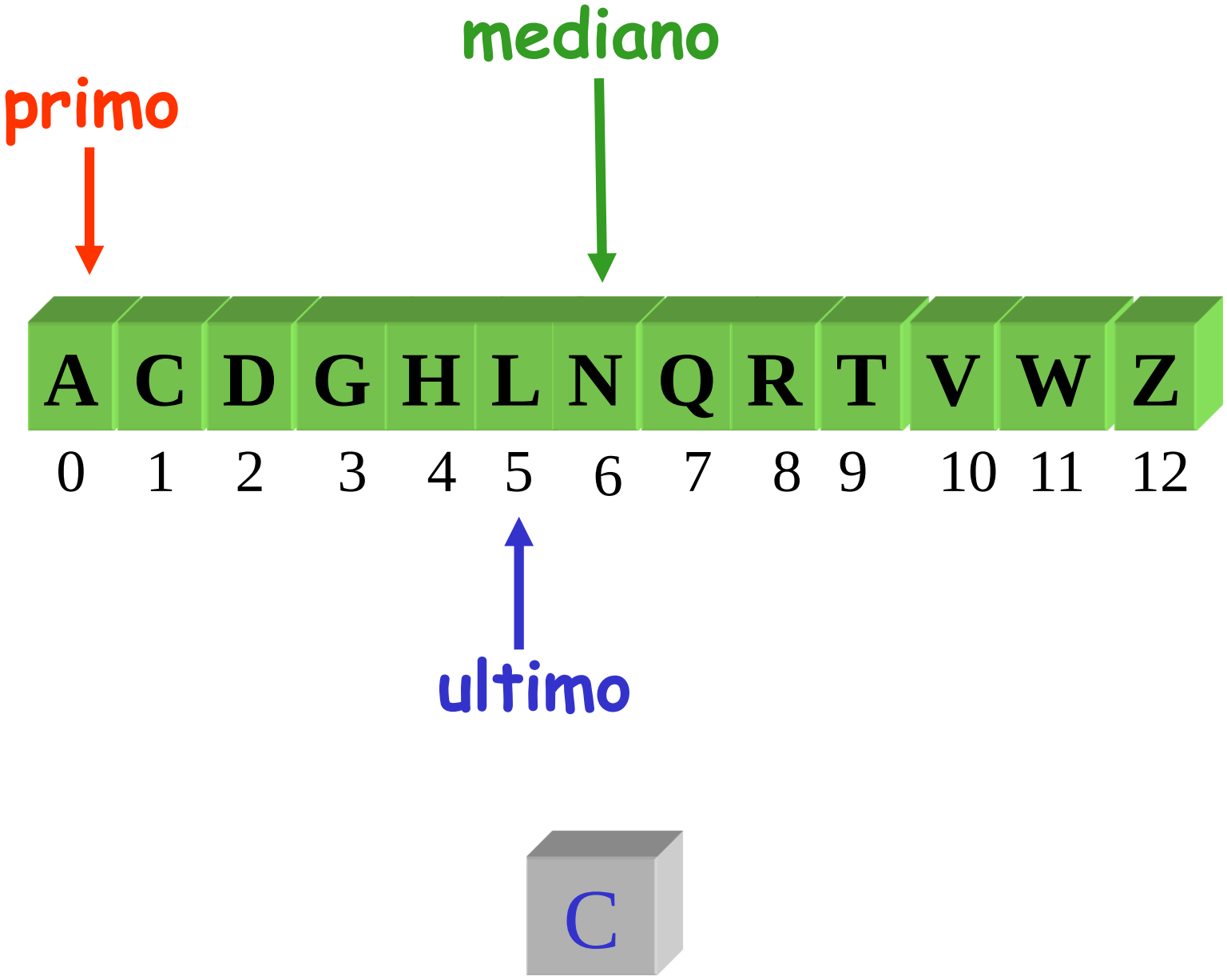
- confrontare il valore da ricercare con il valore dell'elemento di indice **mediano**
- suddividere l'array in due parti
- **eliminare** una delle due parti dell'array dal procedimento successivo di ricerca

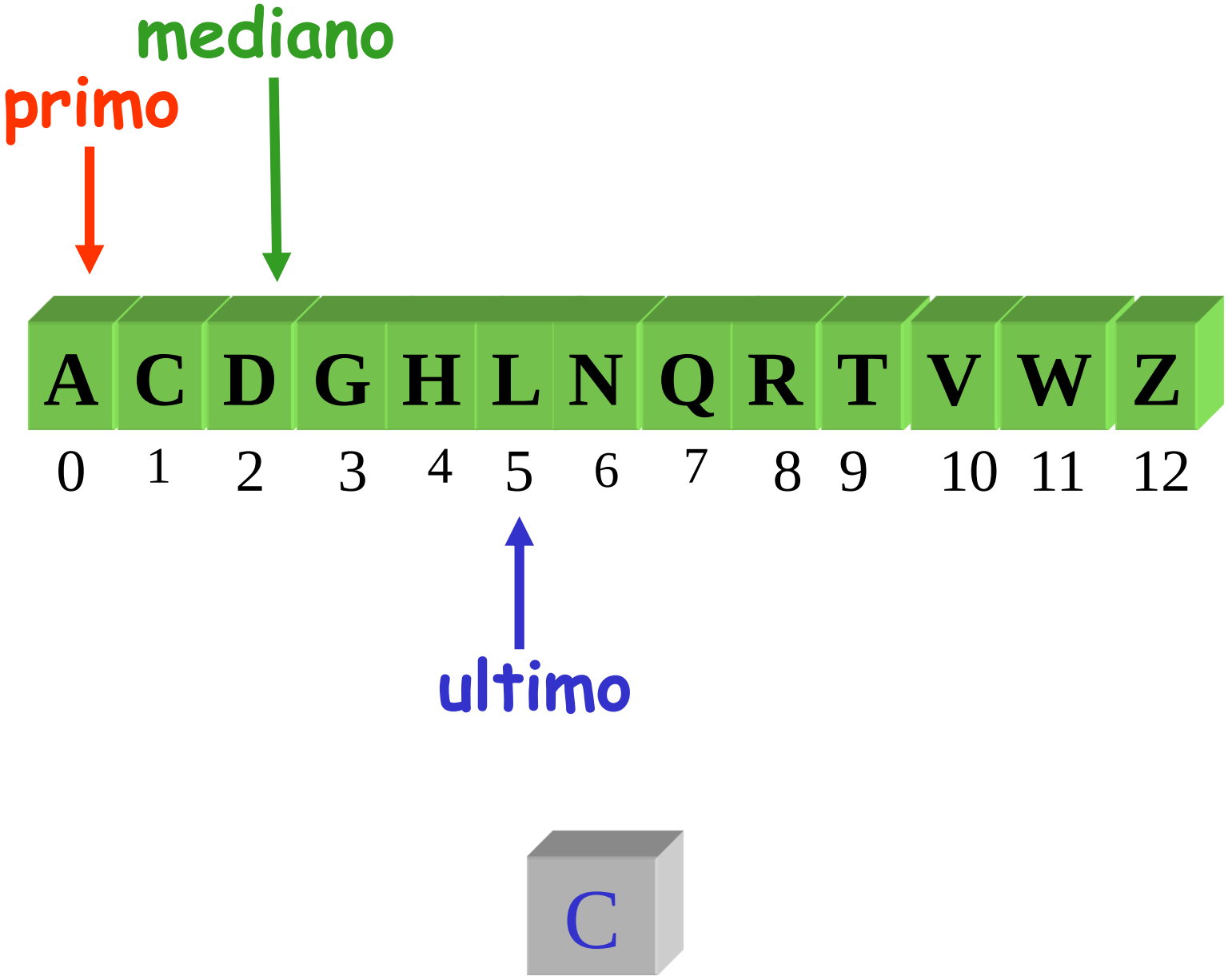
ricerca in un array **ordinato** (**ricerca binaria**)

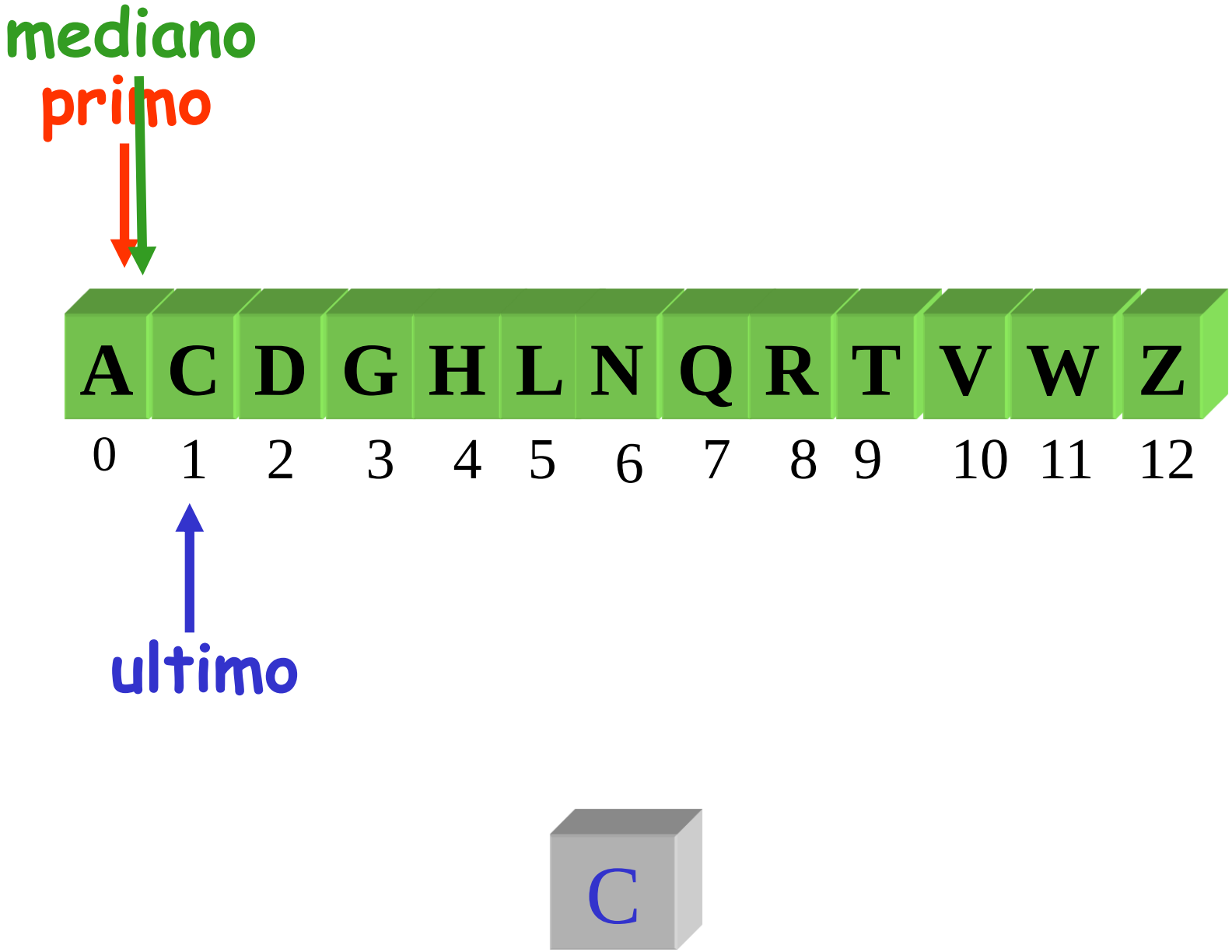
- ogni passo genera una istanza di dimensione dimezzata

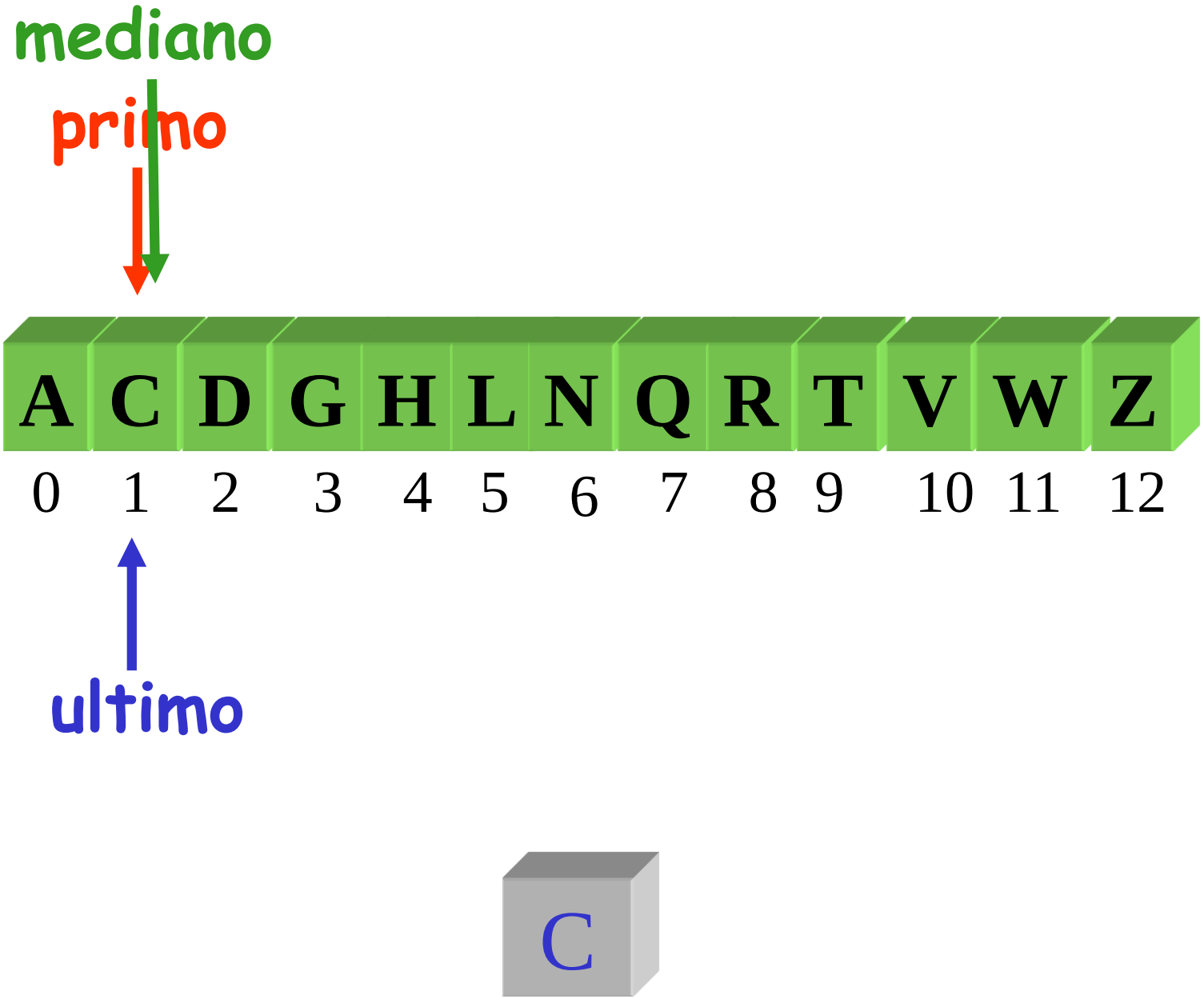



mediano
primo
A C D G H L N Q R T V W Z
0 1 2 3 4 5 6 7 8 9 10 11 12
ultimo
C


mediano
primo
A C D G H L N Q R T V W Z
0 1 2 3 4 5 6 7 8 9 10 11 12
ultimo
C


mediano
primo
A C D G H L N Q R T V W Z
0 1 2 3 4 5 6 7 8 9 10 11 12
ultimo
C


mediano
primo
A C D G H L N Q R T V W Z
0 1 2 3 4 5 6 7 8 9 10 11 12
ultimo
C


mediano
primo
A C D G H L N Q R T V W Z
0 1 2 3 4 5 6 7 8 9 10 11 12
ultimo
C

dati di input: chiave (variabile **chiave**), array ordinato (variabile **elenco**), size dell'array (variabile **n**)

dato di output: il valore dell'indice (se la chiave appartiene all'array); -1 (se la chiave non appartiene all'array)

costrutto ripetitivo: while

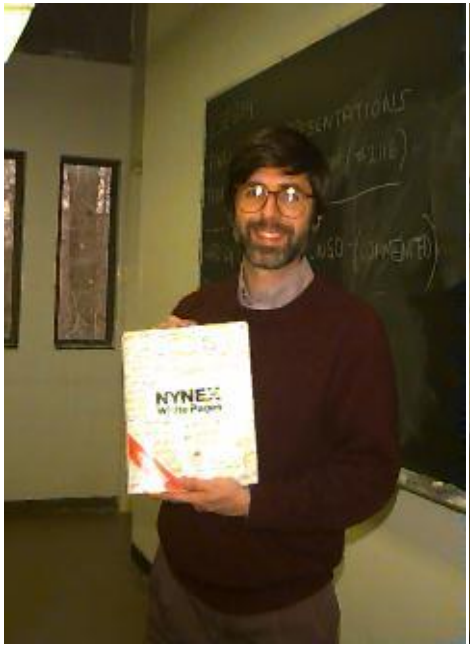
operazione ripetuta (al generico passo)**:**

calcolo dell'indice mediano della porzione di array in esame;

confronto (a tre vie) chiave – elemento array di indice mediano

eventuale determinazione della nuova porzione di array da esaminare

```
int ric_bin (char chiave, char elenco[], int n) {
 int mediano, primo, ultimo;
 primo = 0 ;
 ultimo = n-1 ;
 while (primo <= ultimo) {
   mediano = (primo + ultimo)/2 ;
   if (chiave == elenco[mediano])
        { return mediano ; }
     else if (chiave < elenco[mediano])
             { ultimo = mediano - 1 ; }
             else
             { primo = mediano + 1 ; }
 }
 return -1 ;
}
```


NYNEX