Titolo unità didattica: Algoritmi ricorsivi [14]

Titolo modulo : Algoritmo ricorsivo per la ricerca binaria [02-T]

Tecnica di programmazione ricorsiva e approccio divide et impera

Argomenti trattati:

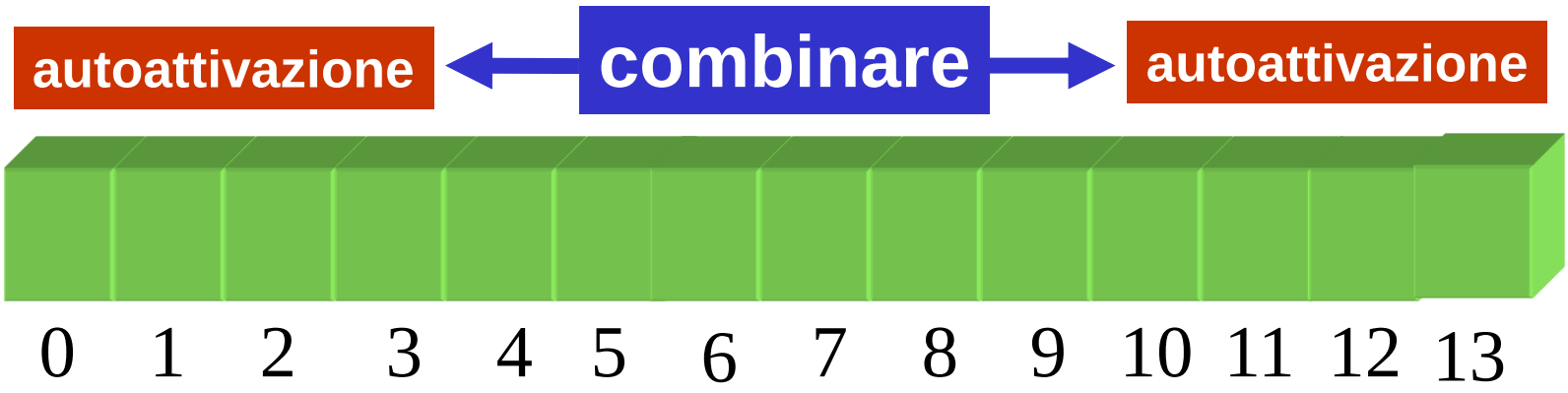
- ✓ suddivisione e autoattivazioni
- ✓ istanza banale e caso base
- ✓ algoritmo ricorsivo per la ricerca binaria

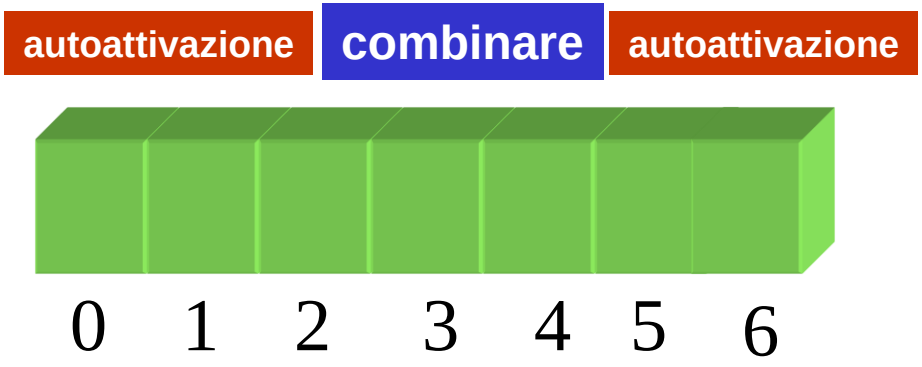
Prerequisiti richiesti: P1-13-01-T, P1-14-01-T

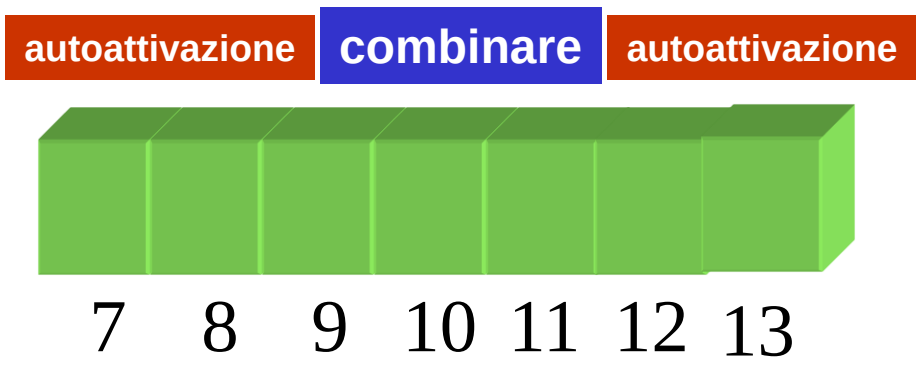
la tecnica di **programmazione ricorsiva** è molto utile nella descrizione di algoritmi basati sull'approccio **divide et impera**

suddivisione dell'istanza di interesse in istanze più semplici e loro **risoluzione**	→	**suddivisione** e **autoattivazioni** della function
istanza banale	→	caso base

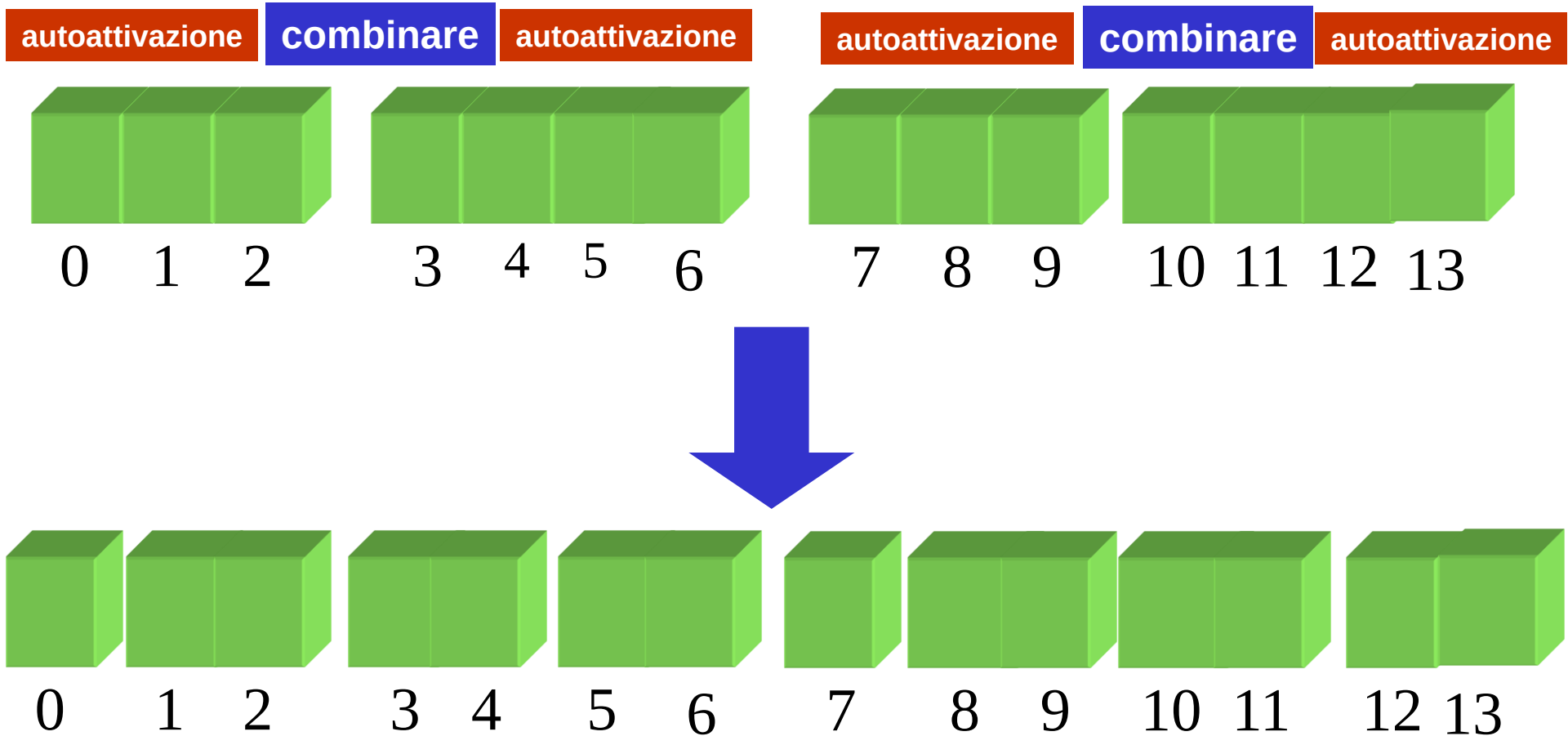
ricorsione e divide et impera







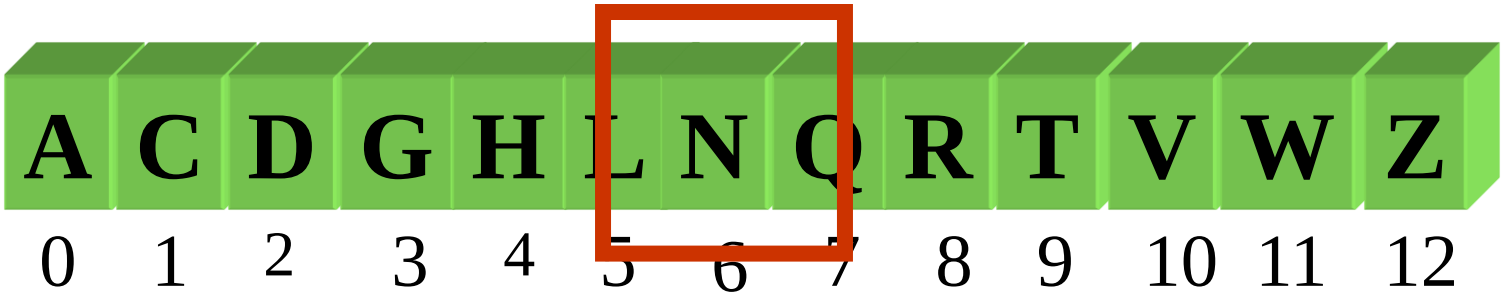
ricorsione e divide et impera



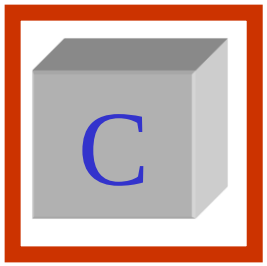
caso base (istanze banali)

ricorsione e divide et impera

algoritmo ricorsivo di ricerca binaria



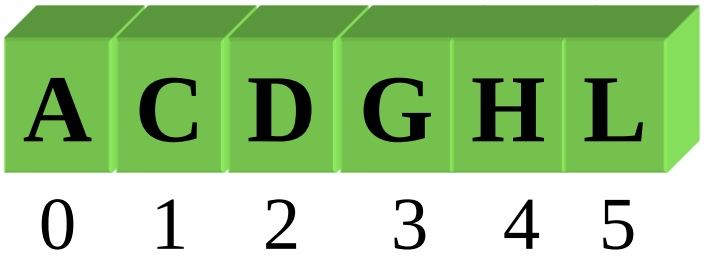
valore da ricercare (**chiave**)

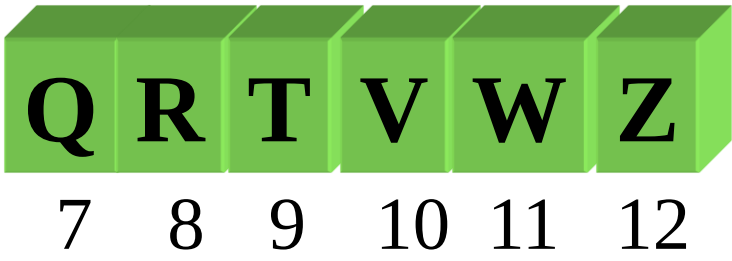


$C \neq N$ ⇒ continuare la ricerca

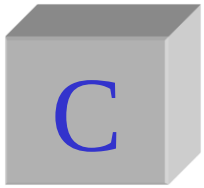
ricorsione e divide et impera

algoritmo ricorsivo di ricerca binaria





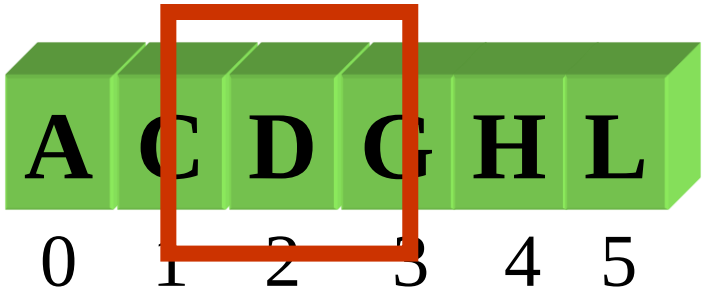
valore da ricercare (**chiave**)



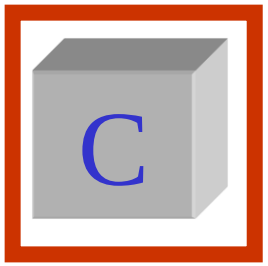
C < N ⇒ C < Q,R,S,T,V,W,Z

ricorsione e divide et impera

algoritmo ricorsivo di ricerca binaria



valore da ricercare (**chiave**)



C < D ⇒ C < G,H,L

algoritmo iterativo di ricerca binaria

```
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 ;
}
```

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

$$T(n) = \lfloor \log_2 n \rfloor + 1$$

algoritmo **ricorsivo** di **ricerca binaria** (versione semplice)

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

caso base

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

autoattivazione su porzione di sinistra

autoattivazione su porzione di destra

$$T(n) = \lfloor \log_2 n \rfloor + 1$$

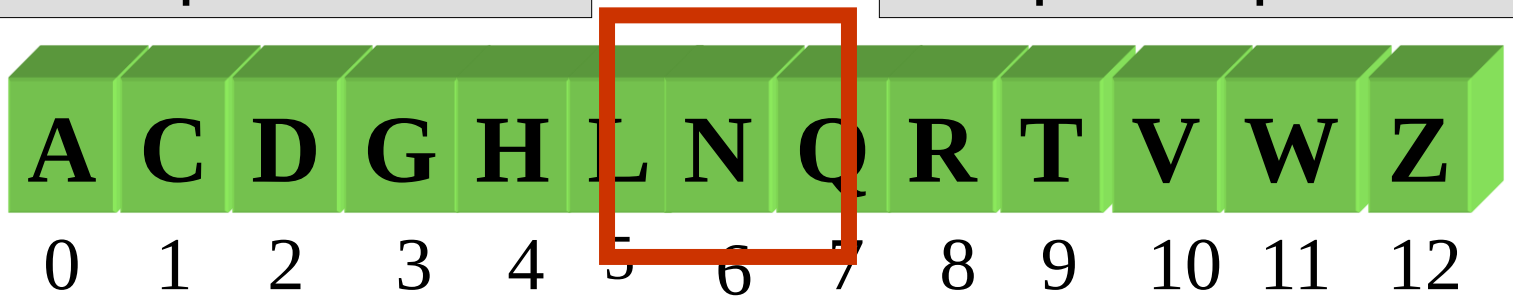
ricorsione e divide et impera

algoritmo ricorsivo di ricerca binaria

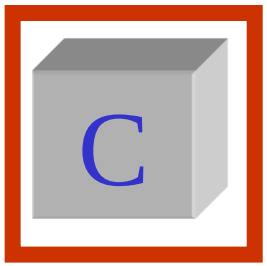
autoattivazione su questa porzione

oppure

autoattivazione su questa porzione



valore da ricercare (**chiave**)



algoritmo ricorsivo di ricerca binaria – versione vero-falso

```
logical ric_bin_ricTF(char chiave, char elenco[], int n) {
 int mediano;
 if (n == 0)
     return false;
  else
    { mediano = (n-1)/2 ;
     if (chiave == elenco[mediano]
         return true;
      else if (chiave < elenco[mediano])
            return ric_bin_ricTF(chiave, elenco, mediano) ;
          else
            return ric_bin_ricTF(chiave,elenco+mediano+1,n-mediano-1);
      }
}
```

```
typedef enum{false,true} logical;
```

caso base

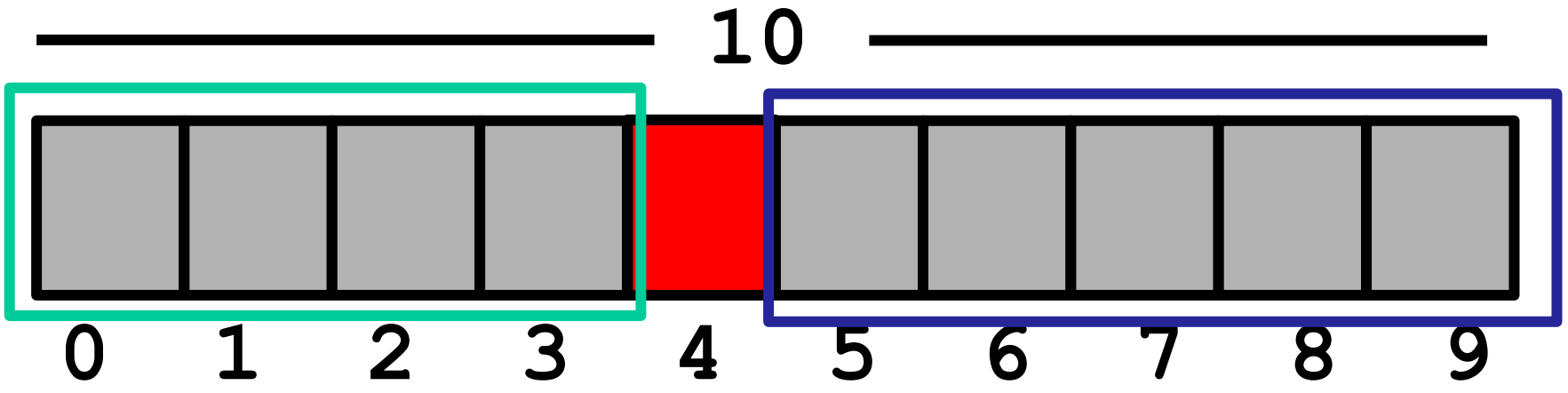
size porzione di sinistra

size porzione di destra

autoattivazione su porzione di sinistra

indirizzo inizio porzione di destra

autoattivazione su porzione di destra



mediano = (10-1)/2 = 4

size porzione sinistra: 4 (0..3)

size porzione destra: (10-4-1) = 5 (5..9)

```
...
   mediano = (n-1)/2 ;
   ...
            return ric_bin_ricTF(chiave, elenco, mediano) ;
      ..
            return ric_bin_ricTF(chiave, elenco+mediano+1, n-mediano-1);
```

algoritmo ricorsivo di ricerca binaria – versione standard

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

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

globalizzazione dell'indice

la globalizzazione (spiazzamento dell'indice) deve essere fatta **solo** nell'autoattivazione sulla porzione di destra dell'array