La ricerca binaria

```
int ricercaBinariaAux(int inizio, int fine, int array[], int elem)
{
1 if (inizio==fine)
  {
  }
2 if (array[inizio]==elem)
3   return inizio;
  else
4   return -1;
  }

5 int mezzo = (inizio+fine)/2;
6 if (array[mezzo]==elem) return mezzo;

  if (elem > array[mezzo])
    return ricercaBinariaAux(mezzo+1,fine,array,elem);
  else
    return ricercaBinariaAux(inizio,mezzo-1,array,elem);
  }
```

questa
questa

7 operazioni + 1 CHIAMATA RICORSIVA

N.B.
7 operazioni abbondando, poiché, per esempio, si può ricadere solo in 1 if (quindi meno di 7 operaz. verranno svolte

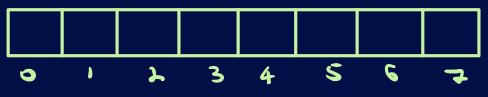
Complessità = num max di operaz (c) ripetuto per ricorsione

DIMENSIONE array = n
ELEM CERCATO = E

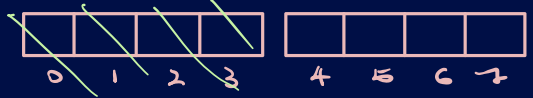
DIMENSIONE porzione DI ARRAY CONSIDERATA	LIVELLO CHIAMATA RICORSIVA
n	prima chiamata ∅ convenzione
* n/2	1
n/4	2
n/8	3

caso base inizio = fine (una cella)

caso peggiore E non c'è



* divido a metà l'array



E > elem in 3? sì, allora volevo controllare nella metà dopo ($n \to \frac{n}{2}$) (8 → 4)

questo xk stiamo supponendo che E sia + grande di tutti gli elem presenti nell'array

E > elem in 5? sì →

etc...

n	j
$\frac{n}{2^0} = 1$	0
$n/2^1 = \frac{n}{2}$	1
...	2
...	3

in relazione

notiamo che $n = \frac{n}{2^j}$ dove j = chiamata ricorsiva

L = livello dell'ultima chiamata ricorsiva → $n / 2^L = 1$ (*) cioè al livello L, dove j = … (L = j xk si parla di livello), noi supponiamo che abbiamo 1 nella cella

(*) $2^L \cdot \frac{n}{2^L} = 1 \cdot 2^L$

$n = 2^L \iff L = \log_2 n$

quindi 0, 1, 2, … $\log_2 n$ chiamate ricorsive

numero (#) di livelli = $\log_2 n + 1$ (xk conta la prima)

complessità: $c \cdot (\log_2 n + 1)$

CASO PEGGIORE $\Theta \log n$

CASO MIGLIORE è si trova nella posizione "mezzo" (trovo subito elem)

quindi Θ_1