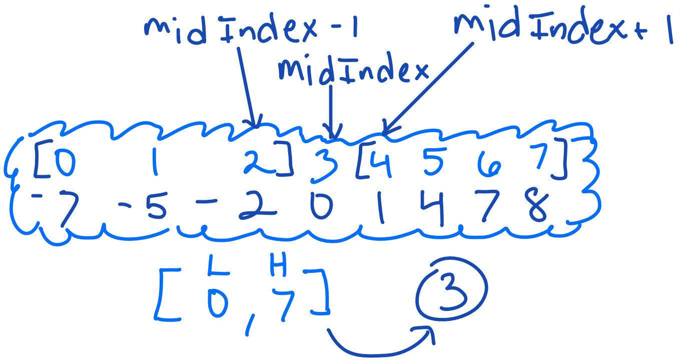
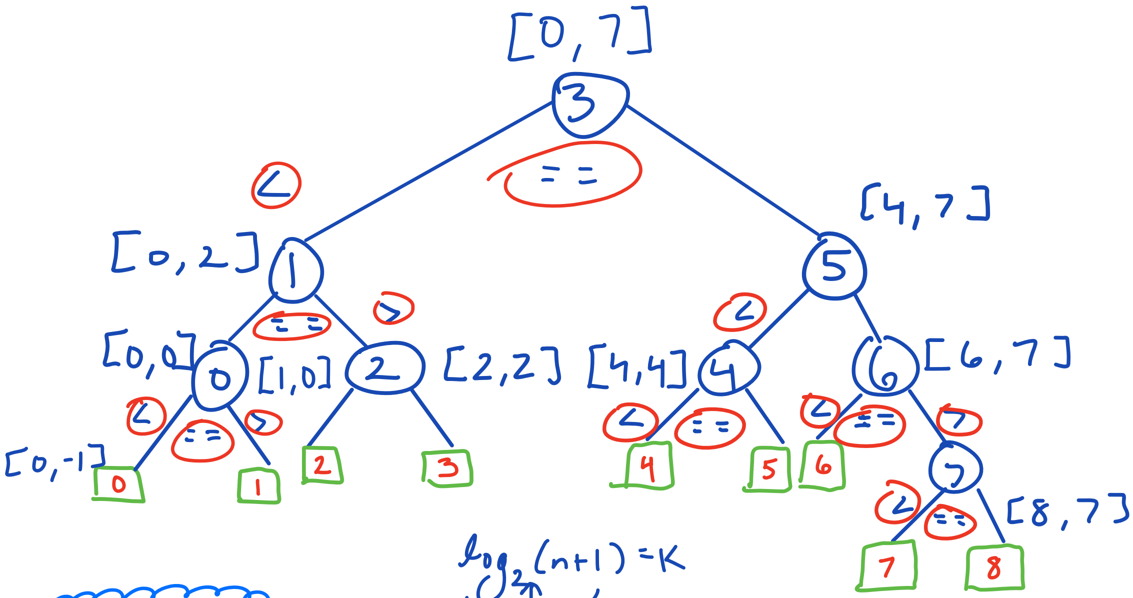
I. Ordered collection using sequential search

II. ordered coll. (↗) Modified Seq search

M. Seq. S. I. → eagerly checks for match ==
II. → checks for unsucc <
III. → eagerly advancing >



```
int low = 0
int high = coll size - 1
while (low ≤ high) // range is not empty
{ midIndex = $\frac{Low + High}{2}$
  if( Key == midKey)
      Stop(succ, midIndex)
  else if (Key < mid Key)
      high = mid Index - 1
  else
      low = midIndex + 1
}
Stop(unsucc, Pos? )
```



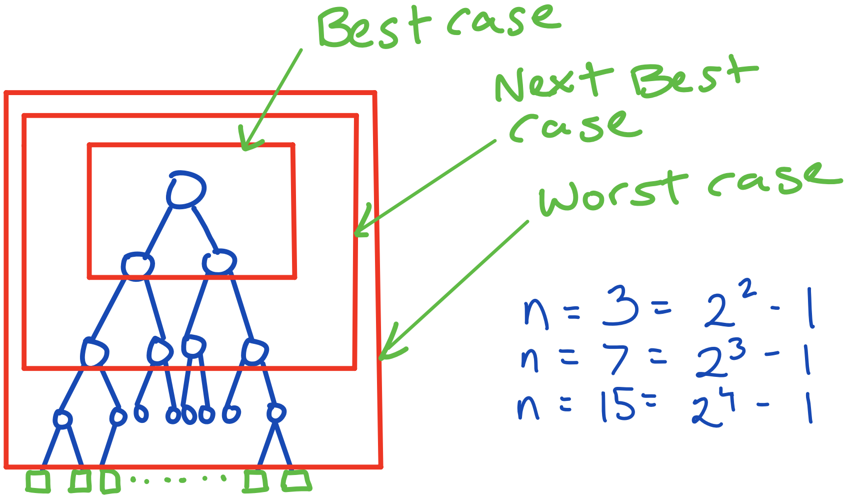
$\log_2(n+1) = K$

$n+1 = 2^K$

$n = 2^K - 1$

Binary Search I (eager == )

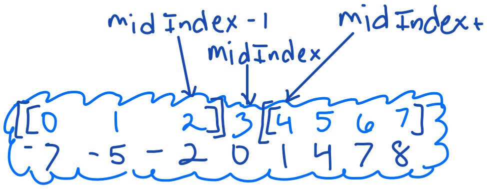
|  | 1 | 2 | 4 | 8 | $\frac{n+1}{2} = \frac{n}{2} + \frac{1}{2}$ | AC |
|---|---|---|---|---|---|---|
|  | Bc | nBc | nnBc | nBC | WC | $2\log_2(n+1)-3$ |
| Succ | 1 | 3 | 5 | 7 . . . | $2\log_2(n+1)-1$ |  |
| unsucc |  |  |  | $2\log_2(n+1)$ |  |  |



$n = 3 = 2^2 - 1$

$n = 7 = 2^3 - 1$

$n = 15 = 2^4 - 1$



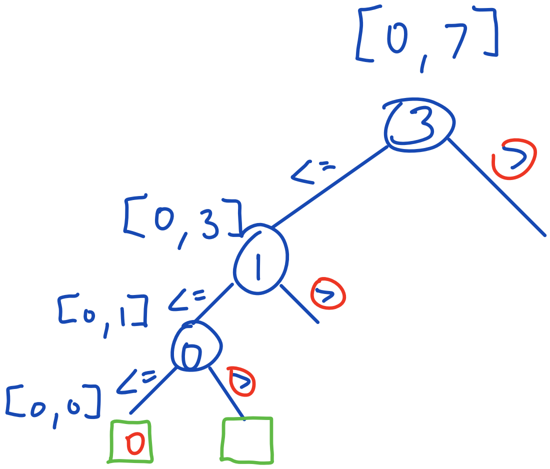
# Binary Search II

```
while (low < high)
{ midIndex = $\frac{low + high}{2}$
  if (Key > midKey)
     low = midIndex + 1

  else
     high = midIndex
}

if (Key == currKey)
   Stop(Succ, pos ? )

else
   Stop(unsucc, pos? )
```



```
int search
{

}
```

(succ, pos) → [0, size-1]
(unsucc, pos) → [0, size]