BINARY SEARCH (A: (a_0 , a_1 ..., a_n))

Input: A sort sequence of integers of size n such that one of the possible integers x is missing

output: The missing integer x

---

if len(A) is 2 : do,

if $a_0 + 1$ is not a_1,

return $a_0 + 1$

else do:

if $a_{mid} == a_{mid-1} + 2$, do

return $a_{mid} + 1$

Let mid = $\lfloor n/2 \rfloor$

Let left be (a_0, a_1 ... a_{mid})

Let right be (a_{mid} , a_{mid+1} ..., a_n)

if length of left != $a_{mid} - a_0$:

return BINARYSEARCH (a_0, a_1 ... a_{mid})

else do:

return BINARYSEARCH (a_{mid}, a_{mid+1}, ... a_n)

Thm: Binary Search is correct

Lemma: Let A be a sort sequence of integer that is finite, non empty and of size n. Suppose that there exist a missing integer in the sequence. Let $low = 0$ and $high = n$. Binary Search will return the missing integer x such that $x \notin A$.

Proof:

Suppose a missing element x exists in A

Basis:

Let mid be $\left\lfloor \frac{high + low}{2} \right\rfloor$

Let high − low = 2. Since the sequence only has 2 element, the missing element must be $a_0 + 1$.

Hypothesis:

Suppose $1 \leq high - low \leq k$ s.t. Binary Search returns x, the missing element.

Inductive:

let n be $n = k+1$ let left ~~be~~ mid - low

let mid be $\left\lfloor \frac{high + low}{2} \right\rfloor$ let right be high - mid

Two possibility exist:

If $a_{mid} = a_{mid-1} + 2$, then $a_{mid} + 1$ is the missing element, BinarySearch returns $a_{mid} + 1$

suppose WLOG, if left is not equal $a_{mid} - a_0$, then there exist a missing element in the left sublist. We then call BinarySearch on the left sublist, which has length greater than 1 and ~~lesser~~ than k. By the hypothesis, BinarySearch returns X correctly

conclusion:

By PMI, BinarySearch is correct.

Time complexity

$$T(n) = 1\,T(n/2) + O(1)$$

master theorem

$A = 1, \quad B = 2, \quad D = 0$

$\log_2 1 = 0$

$0 = 0$

$$\boxed{O(\log n)}$$