Example of computation of worst case (Binary Search)

Result : worst - case running time

$T(n) = \lfloor \lg_2 n \rfloor + 1$.

Binary search, ordered.

Find an item x in an ordered array I based only of comparisons of x to elements of I.

size (I) - number of elements to be searched (= n).

$I[0] \leq I[1] \leq I[2] \leq \ldots \leq I[i-1] \leq$
$I[i] \leq \ldots \leq I[n-1]$
^
midpoint $i = \left\lfloor \frac{n-1}{2} \right\rfloor$

Formulas we will use :

For every $m \in \mathbb{Z}$,

$m = \left\lfloor \frac{m}{2} \right\rfloor + \left\lceil \frac{m}{2} \right\rceil$ $\left(\text{same as } m - \left\lfloor \frac{m}{2} \right\rfloor = \left\lceil \frac{m}{2} \right\rceil\right)$

$\left\lceil \frac{m}{2} \right\rceil = \left\lfloor \frac{m+1}{2} \right\rfloor$ $\left(\text{hence } m - \left\lfloor \frac{m}{2} \right\rfloor = \left\lfloor \frac{m+1}{2} \right\rfloor\right)$

Exercise : prove both (should be easy).

Note : $\left\lfloor \frac{n}{2} \right\rfloor$ is performed by a binary operation "shift one position to the right".

Unsucessful search for $x > I[n-1]$ yields an example of worst - case number of comparisons.

If I contains no duplicates then sucessful search for $x = I[n-1]$ yields an example of worst - case number of comparisons as well.

Optional exercise for smart students
Prove the above two statements.

We will prove that this worst - case number of comparisons is **exactly** equal to $\lfloor \lg_2 n \rfloor + 1$.

First, we will show, by induction on k, that the number of elements that still need to be serched (those are the elements after the last midpoint) after k comparisons is given by this formula

$\left\lfloor \frac{n}{2^k} \right\rfloor$

Proof by induction on k.

Basis step, k = 1.

The midpoint is $\left\lfloor \frac{n-1}{2} \right\rfloor$, and $x > I\left[\left\lfloor \frac{n-1}{2} \right\rfloor\right]$.

The number of elements of I after the midpoint is

$(n-1)$ (*the greatest index*) $- \left(\left\lfloor \frac{n-1}{2} \right\rfloor + 1\right)$ (*the least index*) $+ 1 =$

$= n - 1 - \left\lfloor \frac{n-1}{2} \right\rfloor = \left\lceil \frac{n-1}{2} \right\rceil = \left\lfloor \frac{n}{2} \right\rfloor = \left\lfloor \frac{n}{2^1} \right\rfloor$

This completes the basis step.

Side conclusion. While searching for an element larger than any element of the array of m elements, after one (and unsuccessful) trial, $\left\lfloor \frac{m}{2} \right\rfloor$ elements (those after the midpoint) still need to be searched.

Inductive step, k ≥ 1.

Inductive hypothesis.

Assume that after k comparisons there are still $\left\lfloor \frac{n}{2^k} \right\rfloor$ elements (after the last midpoint) to be searched.

Using the above side conclusion we infer that after next comparison the number of elements that still need to be searched is :

$\left\lfloor \frac{\left\lfloor \frac{n}{2^k} \right\rfloor}{2} \right\rfloor$ = (* because $\left\lfloor \frac{n}{2^k} \right\rfloor$ is shifting n

k positions to the right and

$\left\lfloor \frac{n}{2^{k+1}} \right\rfloor$ is shifting n

k+1 positions to the right *) $\left\lfloor \frac{n}{2^{k+1}} \right\rfloor$

This completes the inductive step.

This completes the proof that the number of elements that still need to be serched (those are the elements after the last midpoint) after k comparisons is given by this formula

$\left\lfloor \frac{n}{2^k} \right\rfloor$

Now, what is the smallest k that makes the number of elements (after the last midpoint) that still need to be searched equal to 0 ?

What is the smallest k that makes

$\left\lfloor \frac{n}{2^k} \right\rfloor = 0$?

What is the smallest k that makes

$0 \le \frac{n}{2^k} < 1$?

What is the smallest k that makes

$n < 2^k$?

What is the smallest k that makes

$n + 1 \le 2^k$?

What is the smallest k that satisfies

$lg_2 (n + 1) \le lg_2 2^k$?

What is the smallest k that satisfies

$lg_2 (n + 1) \le k$?

Answer :

$\lceil lg_2 (n + 1) \rceil$

same as :

$\lfloor lg_2 n \rfloor + 1$

```
Plot[{⌈Log2[n + 1]⌉, Log2[n + 1]}, {n, 1, 127}]
```