1.1 all odd squares can be written as $(2n+1)^2$ $n \in \mathbb{N}$ (incl 0)

if this is true $\forall n$

$\exists m \in \mathbb{N}$ s.t

$$(2n+1)^2 = 8m+1$$
$$4n^2 + 4n \cancel{+1} = 8m \cancel{+1}$$
$$n^2 + n = 2m$$

this works as long as n^2+n is even; factoring we have

$$n(n+1) = 2m$$

an $n \in \mathbb{N}$ if n is odd $n+1$ is even so we will have a factor of 2 and if n is even we obviously have an factor of 2

1.2
$$(2n)^2 = 8m+1$$
$$4n^2 = 8m+1$$

when $n = 1$

$$4 = 8m+1$$

and $\nexists$ $m \in \mathbb{N}$ s.t $4 = 8m+1$

$\Rightarrow \Leftarrow$ doesn't work

3) O(n): is linear time, one extra input increases the worst case by 1 unit per unit size in the search space e.g. searching a list by checking each element sequentially or searching an unordered array

O(1): ~~linear~~ constant time, time is constant regardless of search space

O(log n): time increases with the log of the number of inputs e.g. binary search. if there are m^n inputs it takes n time. Notably this is base agnostic so is an approximation

* this can also be size

=> linear increase in memory

=> constant increase in memory

=> log increase in memory

O(1) as it is the most space efficient and perfectly scalable