5] Infinite Array (Without using array length) Find target

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| arr = [ | 2, | 3, | 5, | 6, | 7, | 8, | 10, | 11, | 12, | 15, | 20, | 23, | 30 ] |

- assume above array is infinite we don't know the length of an array.

- We have to find target Element without using length of array.

Let's look How can we solve it ?

- First we have to find size of an array

↓

we can search target in the chunk (smaller part)

↓

if target is not present in that chunk than we double the size of chunk and apply binary search in that chunk.

- continue this process until we reach at target.

For Example,

chunk size in starting phase is:

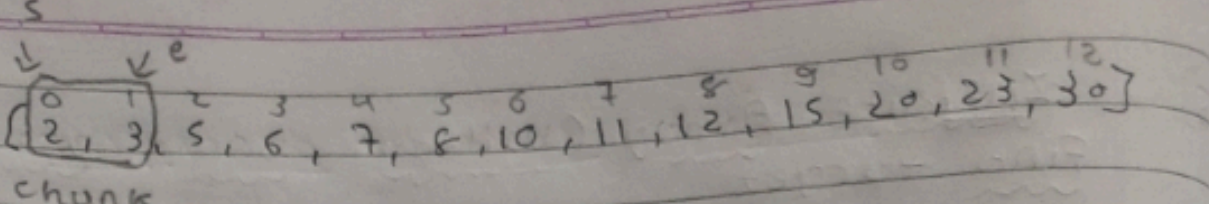
int start = 0;

int end = 2;

0 1

[ , ]

Chunk



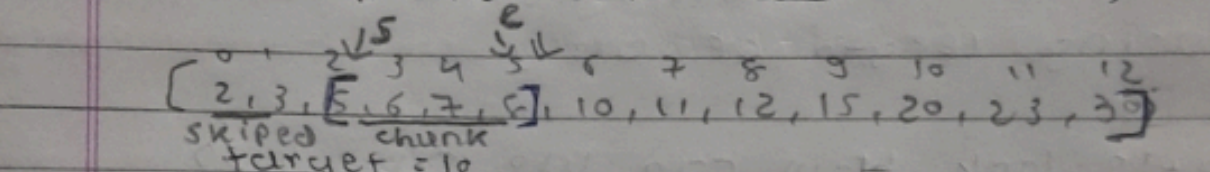

- we double the size of chunk until we sure about target is lie in this range.

by,

// double the size of chunk
while ( target > arr[end] ) {

end = length

int newStart = end + 1;




// Formula = Previous end + size of chunk * 2

end = end + (end - start + 1) * 2;

start = newStart;

}

Now,

target > arr[end] => 10 > 8 => yes

⇓

double the size of chunk by.

start = end + 1 = 5 + 1 = 6

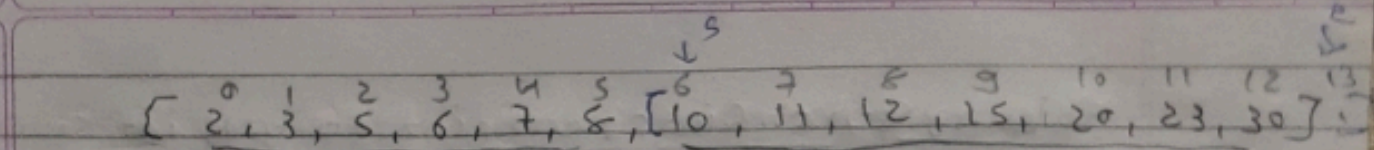
end = end + (end - start + 1) * 2

= 5 + (5 - 2 + 1) * 2

= 5 + (4 * 2)

= 5 + 8

end = 13




Now,

target > arr[end] => No!

↓

So target lies in this range

↓

apply binary search

Let's code

```
InfiniteArray(int[] arr, int target){

  //First we need to find the range where
our target may lie. If not lie than
double the size
   int start = 0;
   int end = 1;


While(end < length of arr. && target > arr[end]){
    int newstart = end + 1;

    end = (end + (end - start + 1) * 2)
    start = newstart;
}
//now apply binary search
return binarySearch(arr, target, start, end);
}
```

```
binarySearch(int[] arr, int target, int start,
  int end) {

    while (start <= end) {
        int mid = start + (end - start) / 2;

        if (target > arr[mid]) {
            start = mid + 1;
        }
        else if (target < arr[mid]) {
            end = mid - 1;
        }
        else {
            return mid; // target found
        }
    }
    return -1; // if target not found
}
```