1	6	8	10	12	14	16	29	40	42	53
0	1	2	3	4	5	6	7	8	9	10

ei

si

data = 45

```
while (si <= ei) {
int m = (si + ei)/2;
if (arr[m] == data) {
return m;
} else if (arr[m] < data) {
si = m+1;
} else {
ei = m-1;
}
}
```

0 1 2 3 4 5 6 7 8 9 10 11

```
int[] arr = { -200, -190, -180, 0, 7, 15, 25, 150, 250, 450, 700, 3750 };
int ans = binarySearch(arr, scn.nextInt());
System.out.println(ans);
```

h, h/2, h/4, h/8, h/16, h/32 ..... 8, 4, 2, 1

1 2 3 4 5 6 ..... K

$a_0 = h$, $a_K = 1$, $r = (1/2)$

Formula: $a_K = a_0 \times r^{K-1}$

$\therefore 1 = h\left(\frac{1}{2}\right)^{K-1}$

$1 = \frac{h}{2^{K-1}}$

$2^{K-1} = h$

$\log_2 2^{(K-1)} = \log_2(h)$

$(K-1)\log_2(2) = \log_2(h)$

$K = \log_2(h) + 1 \simeq K = \log_2(n) + C$

$T = O(\log_2(h))$

$10^4 = 1s$

$10^{-9}s$

$h = 10^{1000}$

$\log_2 10^{1000}$

$\simeq (1000)\log_2 10$

$= 3000$

1000 / 2 / 500 / 50 ~

50x

```java
public static int binarySearch(int[] arr, int data) {
    int n = arr.length, si = 0, ei = n - 1;
    while (si <= ei) {
        int mid = (si + ei) / 2;
        if (arr[mid] == data) {
            return mid;
        } else if (arr[mid] < data) {
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }

    return -1;
}
```

```
{ -200, -190, -180, 0, 7, 15, 25, 150, 250, 450, 700, 3750 };
```

0 1 2 3 4 5 6 7 8 9 10 11

pair sum 46 (8, 40) (12, 36)

[constraints: all numbers are unique, Array is sorted]

Q.

2	4	8	12	13	14	20	22	30	36	40

si ei

```
while (si < ei) {
    int sum = arr[si] + arr[ei]
    if (sum == target) {
        sysO(arr[si] + ", " + arr[ei]); si++, ei--;
    } else if (sum < target) {
        si++;
    } else {
        ei--;
    }
}
```

4 + 6 = 2