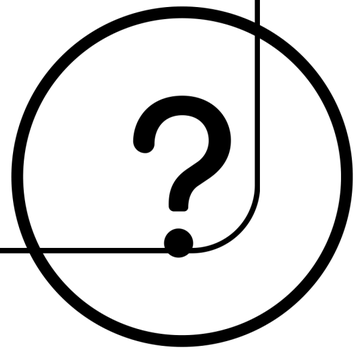
Check Pair Sum in an Array

Let's Understand the problem

-Given an array of n integers and a value targetSum, write a program to check whether there is a pair of elements in the array whose sum is equal to targetSum. If yes, return true; otherwise, return false

-Assume all elements are distinct.

-Values in the array can be both negative and positive



EX1:

Input

TargetSum =15 Output : True

Explanation : (7 , 8) or (-5 , 20) are the pairs with the sum of 15

EX2:

Input

X[]=

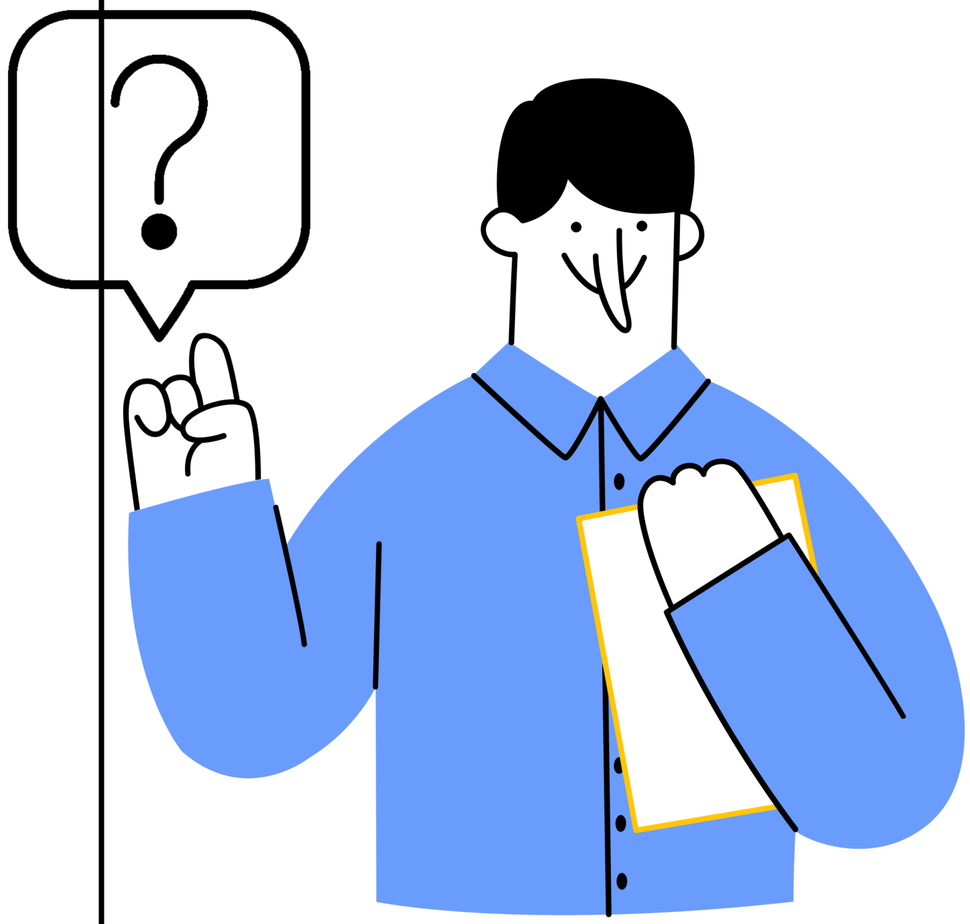
-5	4	-2	16	8	9

TargetSum =15 **Output : False**

Explanation : There is no pair of elements whose sum is equal to 15.

Discussed solution approaches:

-Efficient approach using hash table



Solution idea using hash table:

- The best idea is to use hash table to improve the time complexity further because it performs searching efficiently in the O(1) average.
- So the idea would be to iterate over the array and insert element X[i] into the hash table. Before inserting X[i], we check if the targetSum — X[i] already exists in the table. If it exists, we have found a pair with sum equal to the targetSum and we return true.

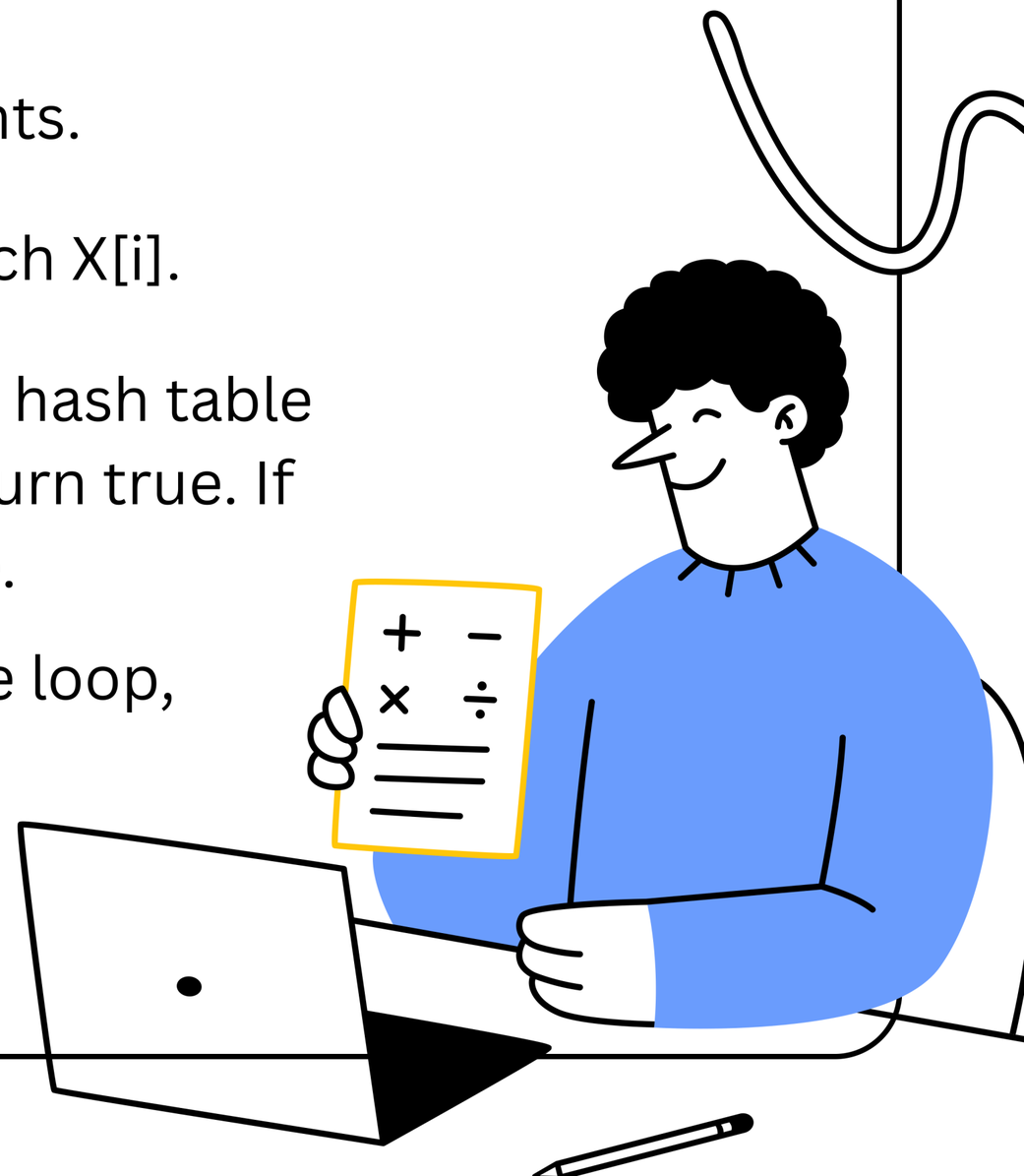
Solution steps:

1. We create a hash table to store array elements.

2. Now we run a loop and scan the array for each X[i].

3. We check if targetSum - X[i] is present in the hash table or not. If yes, we have found a pair and we return true. If no, we insert X[i] into the Hash table.

4. If we didn't find any such pair by the end the loop, return false.



Solution code python

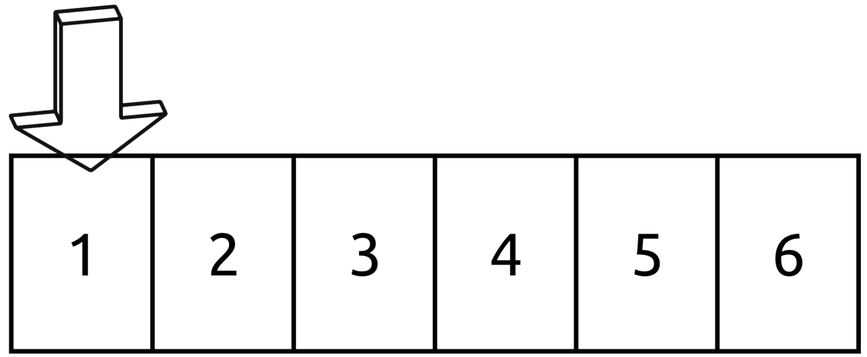
```python
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False

# Example usage:
X = [1, 2, 3, 4, 5, 6]
n = len(X)
targetSum = 9

result = checkPairSum(X, n, targetSum)
```

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

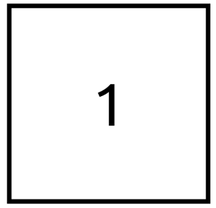
hashTable: Any = set()



i=0

k = 9 - 1=8

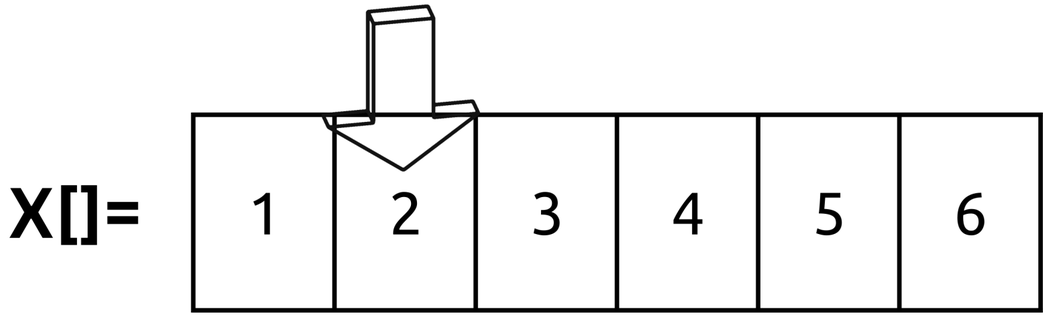
hashTable{}=



n = 6 **targetSum = 9**

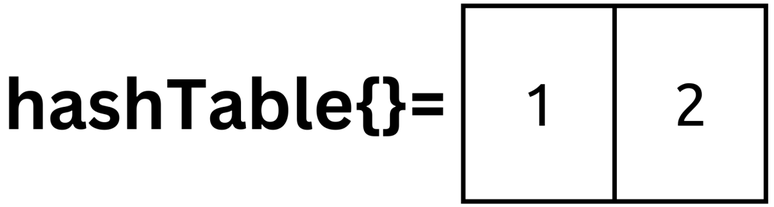
```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

```
hashTable: Any = set()
```



i=1

k = 9 - 2=7

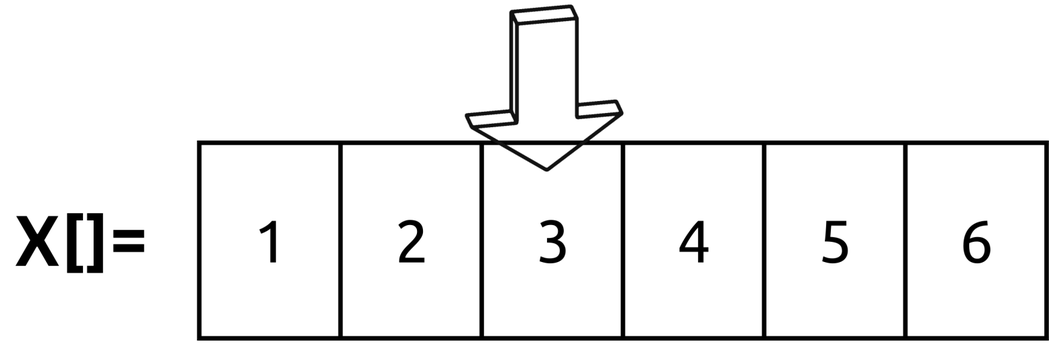


n = 6

targetSum = 9

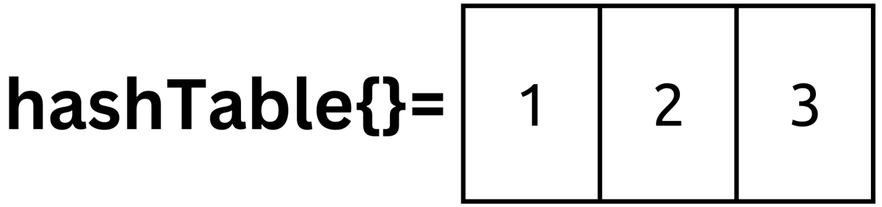
```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()



i=2

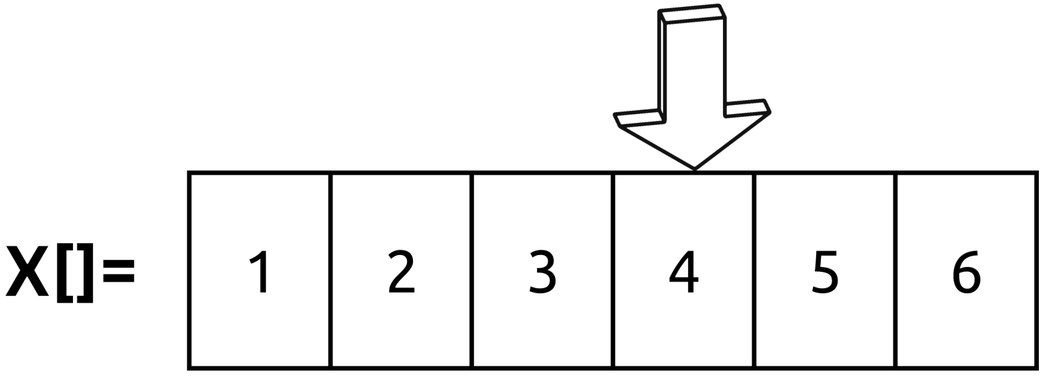
k = 9 - 3=6



n = 6 **targetSum = 9**

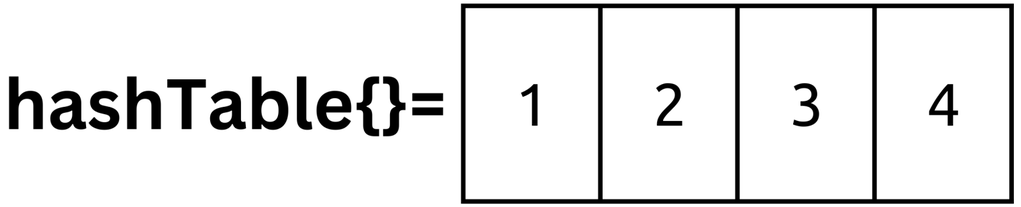
```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()



i=3

k = 9 - 4=5

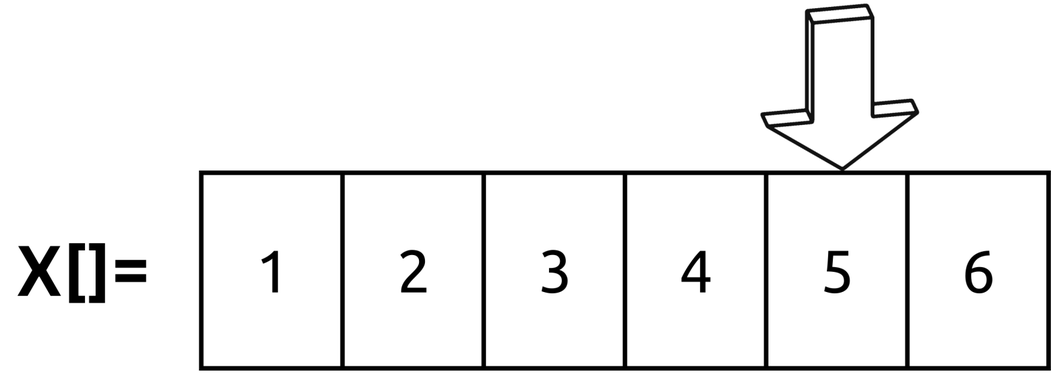


n = 6

targetSum = 9

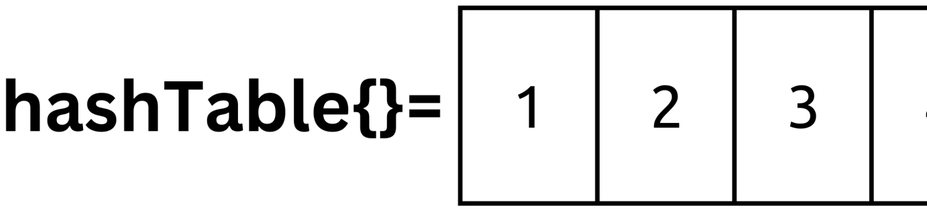
```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()



i=4

k = 9 - 5=4



n = 6 **targetSum = 9**

Solution analysis

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(\ ) =$$

Solution analysis

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) =$$

Solution analysis

```python
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) = 1$$

Solution analysis

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) = 1$$

Solution analysis

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) = \sum 1$$

Solution analysis

```
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) = \sum_{i=} 1$$

Solution analysis

```python
def checkPairSum(X, n, targetSum):
    hashTable = set()
    for i in range(n):
        k = targetSum - X[i]
        if k in hashTable:
            return True
        hashTable.add(X[i])
    return False
```

hashTable: Any = set()

$$T(n) = \sum_{i=0}^{n-1} 1$$

Solution analysis

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{n-1} 1 && , \sum_{i=l}^{u} 1 \\
&= n - 1 - 0 + 1 && , u - l + 1 \\
&= n \\
&\in \Theta(n)
\end{aligned}
$$

Comparisons of time and space complexities

Nestled inape :Time O(n^2), Space = O(1)

Sorting and binary search :Time O(nlogn), Space = O(1)

Sorting and Two pointers:Time O(nlogn), Space = O(1)

Hash Table:Time O(n), Space = O(n)

Thank You!

for Watching