2 sum

array = [5 ② 11 ⑦ 15]   tar = 9
index: 0 1 2 3 4

return [1, 3]

① BF

all pair → sum = tar

```
for (i=0 to n) {
    first = arr[i]
    for (j=i+1 to n) {
        second = arr[j]
        sum = first + second
        if (sum == tar) {
            return ans
        }
    }
}
```

$O(n^2)$

② Better

sorting → $O(n \log n)$

[2 5 7 11 15]

↑ smallest, ↑ largest

sum = smallest + largest = 17

17 > 9

largest --

[2 5 7 11 15]

↑ smallest, ↑ largest

sum = 2 + 11 = 13

13 > 9

largest --

[2 5 7 11 15]

↑ smallest, ↑ largest

sum = 2 + 7 = 9

target ← O(n)

Total TC : $O(n \log n)$

③ Optimized

hashing

arr = [5 2 11 7 15]

tar = 9

pair sum = first + second = tar

(second = tar − first)

<value, idx>
int int
unordered map

5 2 11 7 15
↑ first

second = 9 − 5 = 4

4 not there

element	idx
5	0

5 2 11 7 15
↑ first

second = 7

store 2 in um

element	idx
5	0
2	1

5 2 11 7 15
↑ first

second = −2

store 11 in um

element	idx
5	0
2	1
11	2

5 2 11 7 15
↑ first

second = 2

2 exists in um

element	idx
5	0
2	1
11	2

make pair of 7 ka index and 2 ka index

that is your result

Psuedo Code

```
unordered_map<i, i> m
for (i=0 to n) {          ← TC=O(n)
    first = arr[i]
    sec = tar − first
    if (m.find(sec) != m.end()) {     ← value mil gayi hai
        return [i, m[sec]]
    }
    m[first] = i;
      key      value
}
```

↑ we consider that find function takes constant time for practical solutions

TC : O(n)