4 SUM PROBLEM

☆ In this problem, we are supposed to return 4 different indices whose numbers when added give a target sum. Numbers within a set of 4 can be duplicated but their indices cannot.

Brute force solution is obviously running 4 loops and checking all quadruplets in the array. This gives us an unbearable time complexity of $O(N^4)$. Optionally we are supposed to sort the array of each answer.

Better solution is to involve the use of hashmaps for the last loop. We will create a hashmap in the first iteration and then in the last iteration look for the remaining difference. Time complexity is N^3 for three loops and $\log M$ for hashmap.

Optimal solution is quite the same as 3 sum problem where we used the 3 pointer approach. Here we use 4 pointers i, j, k and l. i and j are fixed at the start, $i = 0, j = i+1, k = j+1$ and $l = N-1$. We find the sum, if it is lesser than required sum, we

increment k and if it is greater than required then we decrement l. If it is equal then we add the answers array and we move both k and l to the next and previous elements respectively. If k and l cross each other we just increase j.

Pseudocode :

```
fourSumProblem (arr, N, K) {
    int ans []
    for (i = 0 → N - 3) {
        for (j = i + 1 → N - 2) {
            k = j + 1
            l = N - 1
            while (k < l) {
                sum = arr[i] + arr[j] + arr[k]
                                        + arr[l]
                if (sum < K) {
                    k++
                } else if (sum > K) {
                    l--
                } else {
                    ans.append({i, j, k, l})
                    k++
                    l--
                    while(arr[k] == arr[k-1]) {
                        k++
                    }
                    while(arr[l] == arr[l+1]) {
                        l--
                    }
```

```
    }
  }
 }
}
return ans
}
```