Problem

[2,3,4,5] taget = 5;

find two indexes whos sum = target;

BRUTE Force O(n^2)

```
vector<int> twoSum(vector<int>& nums, int target) {
        vector<int>ans;
        for(int i = 0; i< nums.size();i++){
            for(int j = i+1; j < nums.size(); j++){
                if(nums[i]+ nums[j]== target){
                    ans.push_back(i);
                    ans.push_back(j);
                    return ans;
                }
            }
        }
        return ans;
    }
```

To optimize it to O(n)

we need to do it in one iteration

i.e we need to find one more element that gives the target

eg: if target = 14 , first element = 8;

8 + __ = 14

we can use a hashmap to store the key values and the fetch the data;

```
vector<int> twoSum(vector<int>& nums, int target) {
       vector<int>ans;
```

```
    unordered_map<int,int>mp;
    for(int i = 0 ; i<nums.size(); i++){
         int targetNum = target- nums[i];
         if(mp.find(targetNum) != mp.end()){
             ans.push_back(i);
             ans.push_back(mp[targetNum]);
             return ans;
         }else{
             mp[nums[i]]= i;
         }
    }
    return ans;

}
```