11/02/2023

Vector
It is an array whose size is not fixed.
Vector is a dynamic array.

Declaration

vector <int> arr;
This will create an integer vector of name arr. Initially the size of vector is 0. If the vector is full, the size of vector gets doubled and hence we don't have to worry about the size.

However the concept of doubling the size of vector can lead to memory wastage.

Initialization

```
int n;
cin>>n;
vector <int> arr(n); → This can be done in case of vectors.

vector <int> brr (10); → Vector of size = 10
vector <int> crr (10, 1); → Vector of size = 10 all initialized with 1.
```

Insertion in vector

```
vector <int> arr;
arr.push_back(5); Inserts 5 in array
arr.push_back(7); Inserts 7 in array
```

cppreference.com
cplusplus.com

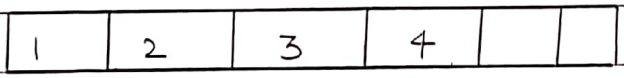
Remove element from vector
arr.pop_back(); This is used to remove an element from vector. (Last element inserted)

Checking size of vector
Function arr.size() is used to check the size of vector.

Checking whether vector is empty or not
Function arr.empty() is used to check whether array / vector is empty or not.

Note → To use vector in our program, we need to include header file of vector which is
#include <vector>

arr.capacity() vs arr.size()



Here size will be 4 and capacity is 6.
Size tells the no. of elements stored and capacity tells that how many elements can be stored.

Printing vector elements

```
for (int i=0 ; i< arr.size() ; i++){
        cout << arr[i] << " ";
}
```

Note → vector <int> brr(10);
brr.size(); → 10 (All initialized to 0)
brr.capacity(); → 10

Note → vector <int> crr (10, 20, 30);

The above line will create vector/array of crr name with 10, 20, 30 values.

Also crr.empty() function will return a boolean value i.e 1 (True) or 0 (False)

* All the concepts of arrays are applicable to vectors also

Problem Solving

① Find unique element in the array. Every element occurs twice except one element. Our task is to find this element.

i/p → {1, 2, 4, 2, 1, (3), 6, 5, 5, 6, 4}
o/p → 3 ↳ occuring only once

Soln We have to find a way by which we can cancel out the elements. This can be done with the help of XOR operator.

1 ^ 2 ^ 4 ^ 2 ^ 1 ^ (3) ^ 6 ^ 5 ^ 5 ^ 6 ^ 4

↳ This is left and rest all are cancelled.

a ^ a = 0 } Properties that we will be
a ^ 0 = a } using in this question.

Code

```
int findUniqueElement (vector <int> arr) {
        int ans = 0;
        for (int i = 0 ; i < arr.size() ; i++){
            ans = ans ^ arr[i];
        }
        return ans;
}
```

We have initialized the ans with 0 as XOR with 0 will give the number itself i.e $a \wedge 0 = a$.

Note → We don't have to pass size explicitly in case of vectors as we have size i.e how many elements are in vector by arr.size() where arr is name of vector.

② Union of two arrays.

i/p → a[] = {2, 4, 6, 8}
b[] = {1, 3, 7}

o/p → {2, 4, 6, 8, 1, 3, 7} → Just put all the elements of both arrays in ans array.

Algorithm

1) Create an ans array / vector
2) Put all elements of a[] into ans array.
3) Put all elements of b[] into ans array.

In the above algorithm we are considering that elements are not repeated i.e no

duplicates.

Code

```
int arr[] = {1, 3, 5, 7, 9};
int sizea = 5;
int brr[] = {2, 4, 6, 8};
int sizeb = 4;
vector<int> ans;

for (int i=0; i<sizea; i++){
        ans.push_back(arr[i]);
}
for (int i=0; i<sizeb; i++){
        ans.push_back(brr[i]);
}
for (int i=0; i<ans.size(); i++){
        cout<<ans[i];
}
```

Note → There can be variations such as ans array is sorted or don't assume duplicates.

(3) Intersection of 2 arrays

i/p → a[] = {1, 2, 3, 4, 6, 8}
b[] = {3, 4, 9, 10}

o/p → {3, 4} → Common elements in the array.

Algorithm

Pick a number from a[] & check whether

it exists in b[] or not. If exists, then push in ans array, if not exists then move to next element in a[] & repeat again.

Code

```
vector <int> arr {1,2,3,4, 6,8};
vector <int> brr { 3, 4, 9, 10};
vector <int> ans;
for (int i=0; i< arr.size(); i++) {
        for (int j=0; j< brr.size(); j++) {
                if (arr[i] == brr[j]) {
                        ans.push_back(arr[i]);
                }
        }
}
```

↗ value belonging to ans

```
for (auto value : ans) {
        cout << value << "  ";
}
```

Note → If the array contain duplicate value say

arr → {1, 2, 3, 3, 4, 6, 8}

brr → {3, 4, 9, 10}

We get o/p as {3, 3, 4} but we don't have 2, 3 in brr. Here we will mark the value if it is visited. Change inside if condition.

```
if (arr[i] == brr[j]) {
        brr[j] = INT_MIN;  → Mark concept
        ans.push_back(arr[i]);
}
```

④ Pair sum in arrays

i/p → a[] = {1, 3, 5, 7, 2, 4, 6}

Find a pair (single pair) whose sum = 9.

o/p → (2,7) or (5,4) or (3,6)
(7,2)

Algorithm
Find all the pairs & check for each pair that whether it is equal to sum or not.
If we get a pair whose sum = 9, just print the pair

{1, 3, 5, 7, 2, 4, 6}

1 → (1,3), (1,5), (1,7), (1,2), (1,4), (1,6)
3 → (3,5), (3,7), (3,2), (3,4), (3,6)
5 → (5,7), (5,2), (5,4), (5,6)
7 → (7,2), (7,4), (7,6)
2 → (2,4), (2,6)
4 → (4,6)

For every element, we will start traversing from the next element till the end for finding the pair.

Code for printing all pairs

```
for (int i=0; i<a.size(); i++) {
        for (int j=i+1; j<a.size(); j++) {
```

```
        cout << "(" << a[i] << ", " << a[j]
                                    << endl;
    }
}
```

Output

(1,3)	(3,5)	(5,7)	(7,2)	(2,4)
(1,5)	(3,7)	(5,2)	(7,4)	(2,6)
(1,7)	(3,2)	(5,4)	(7,6)	(4,6)
(1,2)	(3,4)	(5,6)		
(1,4)	(3,6)			
(1,6)				

Code for pair sum

```
int sum = 9;
for (int i=0; i<a.size(); i++) {
    for (int j = i+1; j<a.size(); j++) {
        if (a[i]+a[j] == sum) {
            cout << "(" << a[i] << ", "
            << a[j] << endl;
        }
    }
}
```

⑤ Triplet sum in arrays. Find a triplet that upon addition gives value equal to sum.

i/p → a[] = {10, 20, 30, 40}
sum = 60
o/p → {10, 20, 30}

Algorithm
This algorithm will be same as that of pair sum but here we use 3 nested loops

Code

```
vector <int> a {1,3,5,7,2,4,6};
int sum = 9;
for (int i=0; i< a.size(); i++){
    for (int j=i+1; j< a.size(); j++){
        for (int k=j+1; k< a.size(); k++){
            if (a[i]+a[j]+a[k]==sum){
                cout << a[i] << " " << a[j]
                << " " << a[k] << endl;
            }
        }
    }
}
```

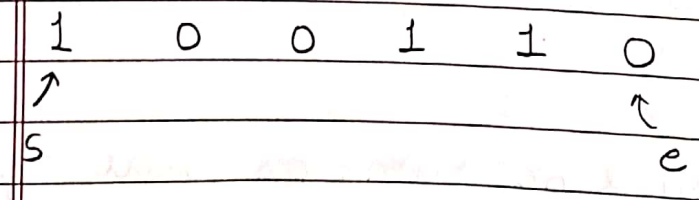
Note → Suppose now we want 4 elements that have sum equal to given value, here 4 nested for loops will be used.

⑥ Sort 0s and 1s in the array. It is also known as Dutch national flag problem

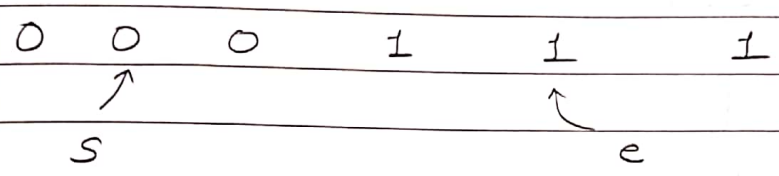
i/p → {1, 0, 0, 1, 1, 0}
o/p → {0, 0, 0, 1, 1, 1}

This can be solved with the help of 2 pointer approach.

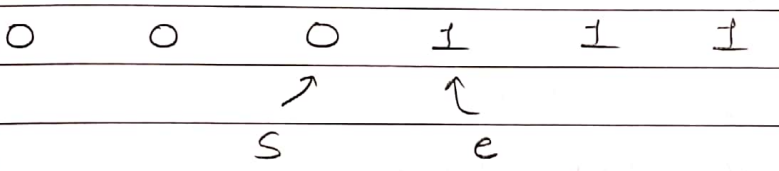
Algorithm



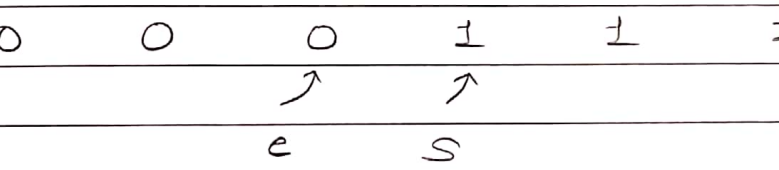
arr[s] = 1 && arr[e] = 0 , swap both & increment s → s++ & decrement end i.e e → e--



arr[s] == 0 → s++
arr[e] == 1 → e--



arr[s] == 0 → s++
arr[e] == 1 → e--



Now s > e } end/exit from loop.

Code

```
int s = 0;
int e = arr.size() - 1;
while (s <= e) {
        if (arr[s] == 0) { s++; }
        if (arr[e] == 1) { e--; }
        if (arr[s] == 1 && arr[e] == 0){
                swap (arr[s], arr[e]);
                s++;
                e--;
        }
}
```

Code - 2

```
int s = 0;
int e = arr.size() - 1;
int i = 0;
while(s <= e) {
    if (arr[i] == 0) {
        swap(arr[s], arr[i]);
        i++;
        s++;
    }
    if (arr[i] == 1) {
        swap(arr[e], arr[i]);
        e--;
    }
}
```

for each loop

```
for (auto i : arr) {

}
```

1) auto is a keyword by which we don't have to tell what is the keyword (data type)

2) i is variable name.

3) arr is vector here.

With the help of this loop we can write loop quickly. Better to use for each loop

however we can use for loop.