11/Feb/2022

Lec-8

Array-2

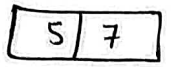
⊙ Vector

↳ Dynamic array.

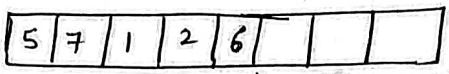
declaration → keyword ↓ vector <int> arr; ← datatype, ← variable name.

default size ⇒ 0 , capacity ⇒ 0

It doubles it's size whenever it is fully filled.

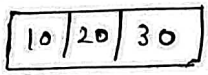
eg, 

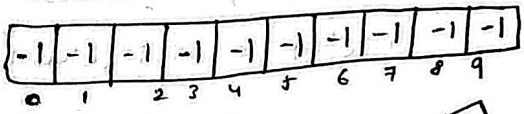
← vector array is full, now it will increase it's size.



← This memory is wasted.

Initialization-

→ vector <int> arr (10,20,30); → 

→ vector <int> brr (10, -1); → 

→
```
int n;
cin>>n;
vector<int> arr[n];
```

- **how to insert elements →**

```
arr.push_back(5);
arr.push_back(7);
```

- **remove the last element -**

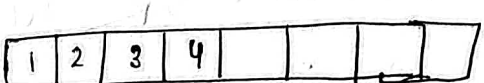
```
arr.pop_back();
```

- **size -** returns the number of elements in vector.

```
arr.size();
```

- **empty -** arr.empty();

- **capacity()** → returns the size of the storage space currently allocated for the vector.

eg, 

capacity() → 8

size() → 4

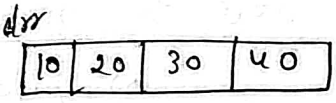
websites
↳ cppreference.com
↳ c++.com

$\frac{\text{sizeof(arr)}}{\text{sizeof(int)}} \rightarrow \frac{6,3}{\hookrightarrow \text{compiler dependent.}}$

⇉
```
int n;
cin>>n;
```

→
```
vector<int> brr(10);
cout << "size of brr " << brr.size() << endl;        // 10
cout << "capacity of brr" << brr.capacity() << endl; // 10
// printing the vector.
for (int i=0; i< brr.size(); i++){
        cout << brr[i] << " ";      // o/p → 0 0 0 0 0 0 0 0 0 0
}
```

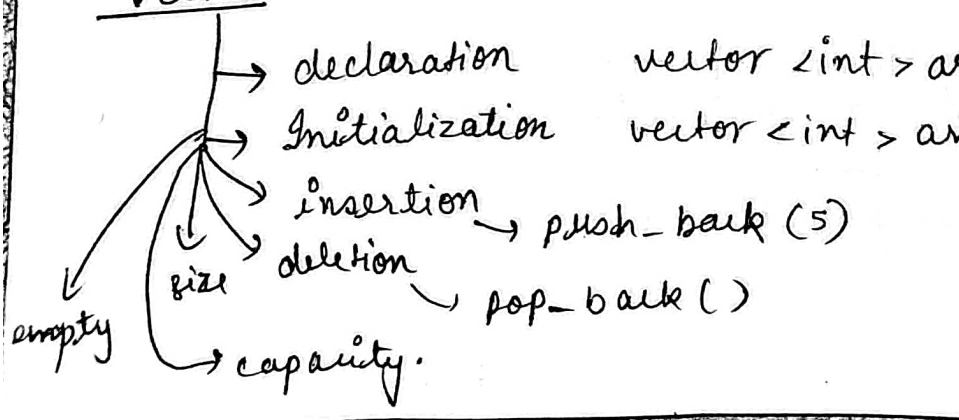
// Taking input for the vector :-
```
int n;
cin >> n;
vector <int> crr (n, -10);
```
// Create an n size vector with having value of -10 each.

→ `vector <int> drr {10,20,30,40};` → 

⊙ empty or not ?
```
cout << drr.empty() << endl;
```
// 0 → False
⇓ that means it's not empty.

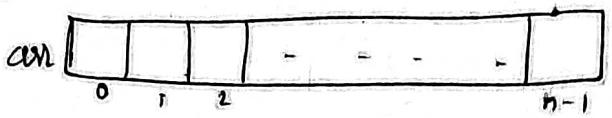
→
```
vector <int> err;
cout << err.empty() << endl;
```
// 1 → true means it's empty.

• Vector



// how to access n^{th} element.

↳ by indexing. , arr [n-1]



Q- Find Unique element.

i/p → arr | 1 | 2 | 4 | 2 | 1 | 3 | 6 | 5 | 5 | 6 | 4 |

every element occurs twice, only 1 element occurs once.

⇒ XOR

1 ^ 2 ^ 4 ^ 2 ^ 1 ^ 3 ^ 6 ^ 5 ^ 5 ^ 6 ^ 4

= 3 ans

```
main(){
    int n;
    cin >> n;
    vector <int> arr(n);
    for (int i = 0; i < n; i++){
        cin >> arr[i];
    }
    int uniqueElement = findUnique(arr);
                        ↳ func
    return 0;
}

int findUnique(vector<int> arr){
    int ans = 0;
    for (int i = 0; i < n; i++){
        ans = ans ^ arr[i];   ⟶ // ans = ans ^ arr[i];
    }
    return ans;
}
```

Why we initialized ans with 0.

⇒ because →

0 ^ 0 → 0

0 ^ n → n eg → 0 ^ 1 = 1

so it is ineffective to XOR with 0.

⑥ Union of 2 array →

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

o/p Union → {2, 4, 6, 8, 1, 3, 7}
↑ all elements with no duplicate

Algo:
→ create an ans array/vector.
→ Put all elements of a into ans array.
→ Put all elements of b into ans array.
→ done. (Assuming there are no duplicates).

```
int n, m;
vector<int> arr(n);
vector<int> brr(m);
for (int i=0; i<n; i++){
        cin>>arr[i];
}
for (int i=0; i<m; i++){
        cin>>brr[i];
}
```

} Taking Input

```
vector<int> ans;
for (int i=0; i<n; i++){
        ans.push_back(arr[i]);
}
```

} storing arr into ans array.

```
for (int i=0; i<m; i++){
        ans.push_back(brr[i]);
}
```

} storing brr into ans array.

```
// Printing the ans vector.
for (int i=0; i<n+m; i++){
        cout<<ans[i];
}
```

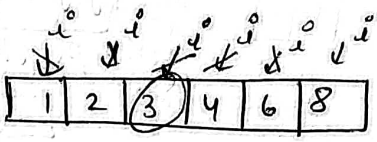
} Printing the array.

↳ Some modified questions.
  ↳ When duplicates exist
  ↳ Answer should be in # increasing order.

⊙ Intersection of two arrays.

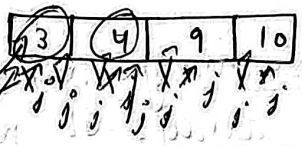
i/p → a = { 1, 2, 3, 4, 6, 8 }
      b = { 3, 4, 9, 10 }

o/p → { 3, 4 }



Algo:

→ Pick an element from array a.
→ compare with each element of b array.
→ if they are same then store it in ans array.
loop



ans = | 3 | 4 |

⇒ **Code -**

```
vector<int> ans;
for (int i=0; i<n; i++){
    int element = arr[i];
    for (int j=0; j<m; j++){
        if (element == brr[j])
            ans.push_back(element);
    }
}
// print the ans vector.
// for (int i=0; i<   // for-each loop.
for (auto value : ans){
    cout << value << " ";
}
```

→ What if we have duplicate element in a array.

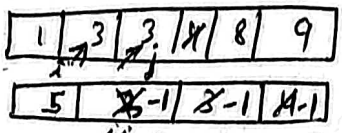
eg → a = { 1, 3, 3, 4, 6, 8 }
     b = { 3, 3, 4 }

output according to this code → { 3, 3, 3, 3, 4 }

But it is wrong. o/p should be { 3, 3, 4 }.

so we will just add mark to show that the element is already matched with an element of another array.

```
for (i → 0 to n-1){
    int element = arr[i];
    for (j → 0 to m-1){
        if (element == brr[j]){
            brr[j] = -1;      // mark
            ans.push_back(arr[i]);
        }
    }
}
```



Here assuming that only positive integer exist in our array. If negative integers also exist then we can mark with INT_MIN.

→ now we can easily avoid duplication in union.

```
if (brr[j] != -1)
    ans.push_back(element);
}
```

code in H.W

⑥ Pair Sum →

i/p → a = {1, 3, 5, 7, 2, 4, 6}, find two elements (a pair) which adds up gives value equal to target.

Brute force →

→ find out all pairs. Check their sum is target or not.

(1,3)4 (3,5)8 (5,7)12 (7,2)9 (2,4)6 (4,6)10
(1,5)6 (3,7)10 (5,2)7 (7,4)11 (2,6)8
(1,7)8 (3,2)5 (5,4)9 (7,6)13
(1,2)3 (3,4)7 (5,6)11
(1,4)5 (3,6)9
(1,6)7

target = 40
arr[] → {10, 20, 30, 40}
element = 10
40 - 20 30 40 50
element = 30
outer loop 40 - 70
element = 20
40 - 30 40 60
ans = (10, 30) inner loop

```
int target = 80;
vector <int> arr{10, 20, 30, 40, 60, 70};
// print pairs, outer loop will traverse for each element.
for (int i = 0; i < arr.size(); i++){
    int element = arr[i];
    // Traverse from (i+1)th element.
    for (int j = i+1; j < arr.size(); j++){
        cout << "(" << element << ", " << arr[j] << ")" << endl;
    }
}
```

↑ This will print all pairs.

```
// now pair sum.
for (int i = 0; i < arr.size(); i++){
    for (int j = i+1; j < arr.size(); j++){
        if (arr[i] + arr[j] == target)
            cout << arr[i] << " " << arr[j]
                 << endl;
    }
}
```

② ~~Triplet Sum~~ -

outer loop → (x, y) → inner loop.

Dry Run -

arr[] → {10, 20, 30, 40} target = 40

(10, y)

10, 20 → 10+20=30

10, 30 → 40

10, 40 → 50

{10, 20, 30, 40}
 ↑i

(20, y)

↳ (20, 30) → 50

↳ (20, 40) → 60

{10, 20, 30, 40}
 ↑i

(30, y)

↳ (30, 40) → 70

③ Triplet Sum -

a[] → {10, 20, 30, 40}, sum = 80.

(10, 20, 30) 60 — (20, 30, 40) 90

(10, 20, 40) 70

(10, 30, 40) 80 ✓ ans

{10, 20, 30, 40}
 ↑i
 (40, y)

loop i = 0 → n-1

(x, y, z) → loop k = · → n-1

loop j = i+1 → n-1

```
vector <int> arr {10, 20, 30, 40};
for (int i = 0; i < arr.size(); i++){
    int element1 = arr[i];
    for (int j = i+1; j < arr.size(); j++){
        int element2 = arr[j];
        for (int k = j+1; k < arr.size; k++){
            if (element1 + element2 + arr[k] == sum)
                cout << element1 << " " << element2
                << " " << arr[k] << endl;
        }
    }
}
```

⊙ Find 4 nums that adds up to sum :

Algo/ pseudo code.

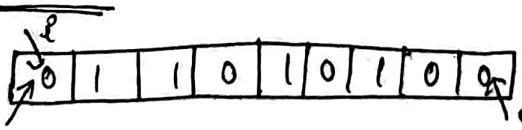
```
vector <int> arr {10, 5, 15, 20, 30};
int sum = 20;
for (int i = 0 → < arr.size()){
    for (int j = i+1 → < arr.size()){
        for (int k = j+1 → < arr.size()){
            for (int l = k+1 → < arr.size()){
                if (arr[i] + arr[j] + arr[k] + arr[l]
                    == sum){
                cout << arr[i] << arr[j]
                << arr[k] << arr[l];
            }
        }
    }
}
```

⊙ Sort 0's & 1's →

i/p → arr → | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

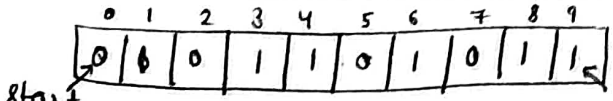
o/p → | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Two pointers approach →



put 0 on start ← l

r ⇒ put 1 on end.



size = 10

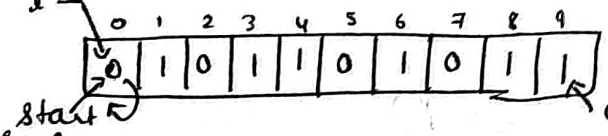
start ↳ shows a position where we can insert next 0.

end ↳ shows a position where we can insert next 1.

```
if (arr[i] == 0)
    swap(arr[i], arr[start])
    i++;
    start++;
}
```

```
if (arr[i] == 1){
    swap(arr[i], arr[end]);
    end--;
}
```

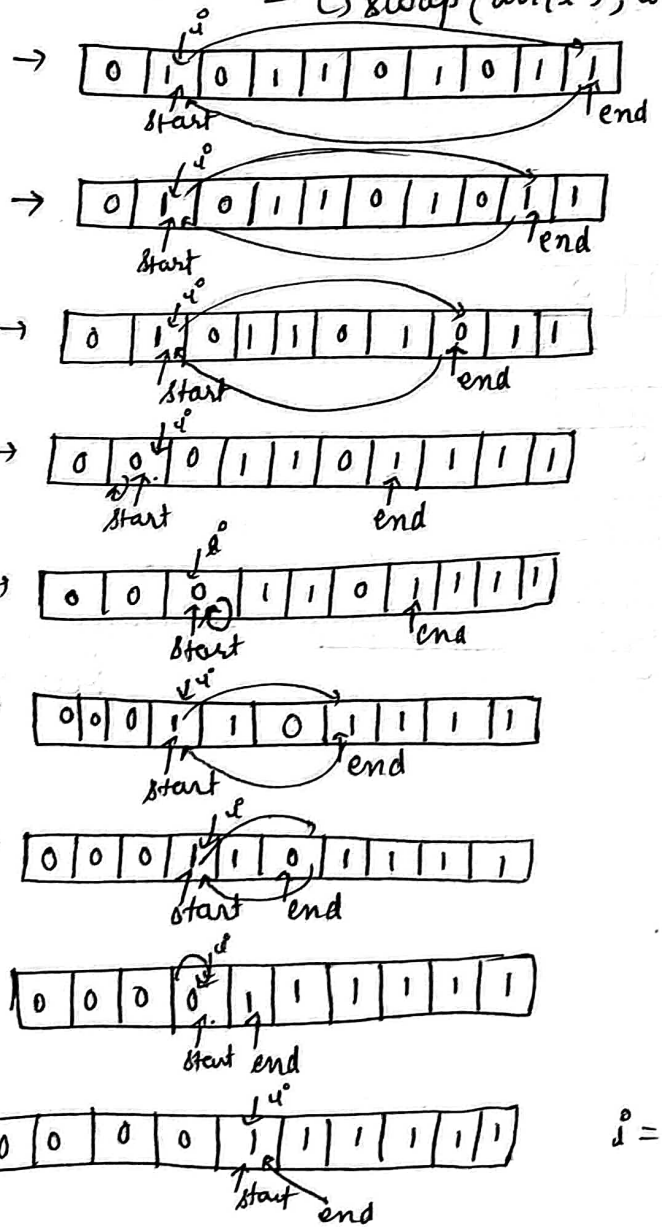
dry run - i → i is used to traverse the array.



i = 0

arr[start] = 0

arr[i] = 0 ↳ swap(arr[i], arr[start]), i++, start++;

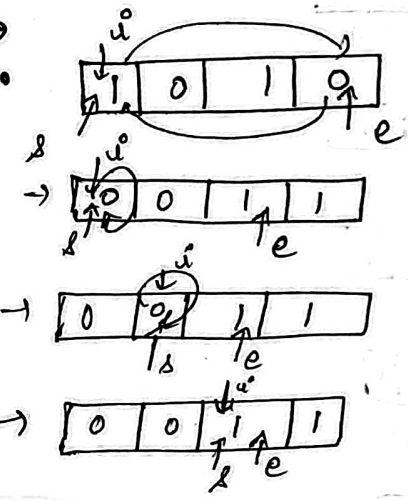


i == end → stop

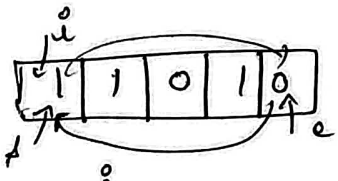
code -

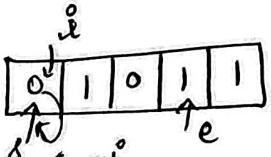
```
vector <int> arr{0,1,0,1,1,0,1,0,1,1};
int start = 0;
int end = arr.size()-1;
int i = 0;
while (i != end){
    if(arr[i] == 0)
        swap(arr[i], arr[start]);
        i++;
        start++;
    }
    if (arr[i] == 1)
        swap(arr[i], arr[end]);
        end--;
    }
}
```

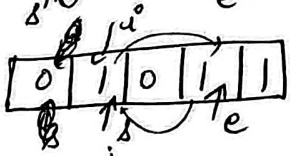
dry run →

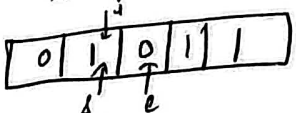


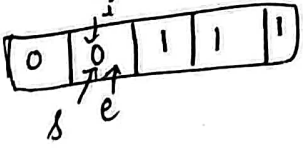
i == end → array is sorted. (out of loop).











i == e → Stop the loop.

① for - each loop -

```
for (auto i : ar)
    cout << i;
}
```

keyword.

By using auto keyword we don't need to tell the datatype explicitely.

Questions -

① Left Rotate an array by 1 element.
② Majority element in an array.
③ Buy & Sell Stock → Level-1