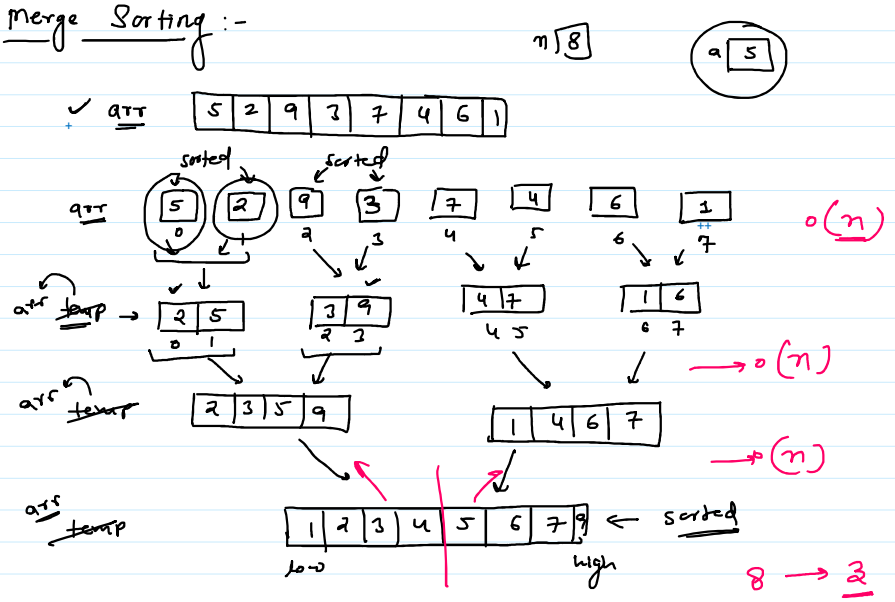
Day 13

01 April 2025 08:30 AM

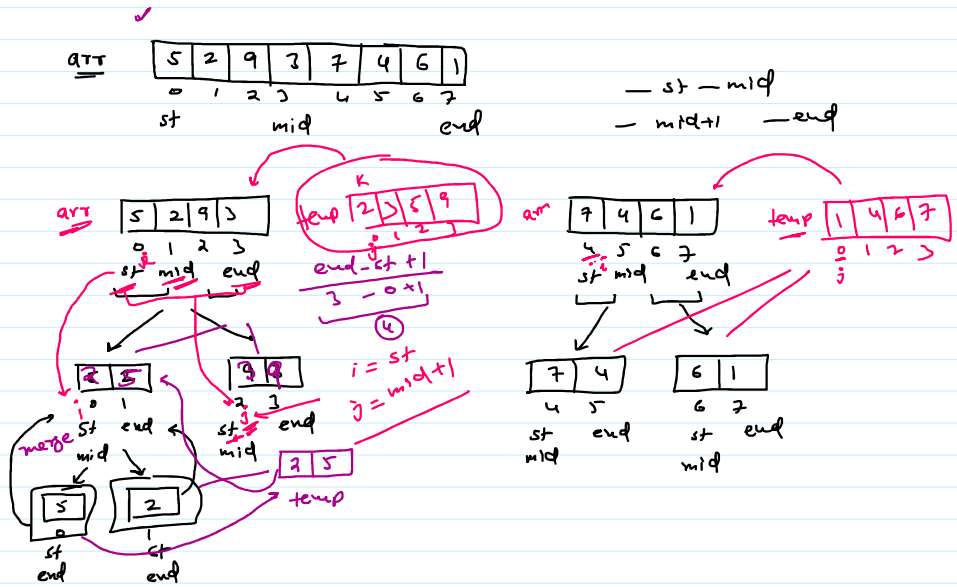
Merge Sorting :-



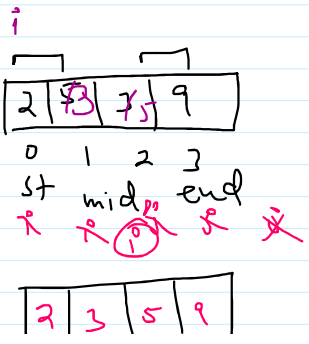
time → $o(n \log_2 n)$

space → $o(n)$

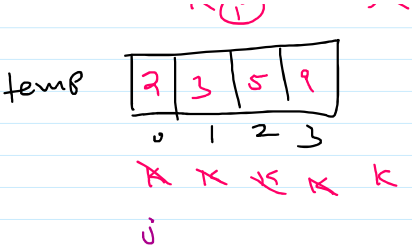
mergeSorting (arr, st, end)

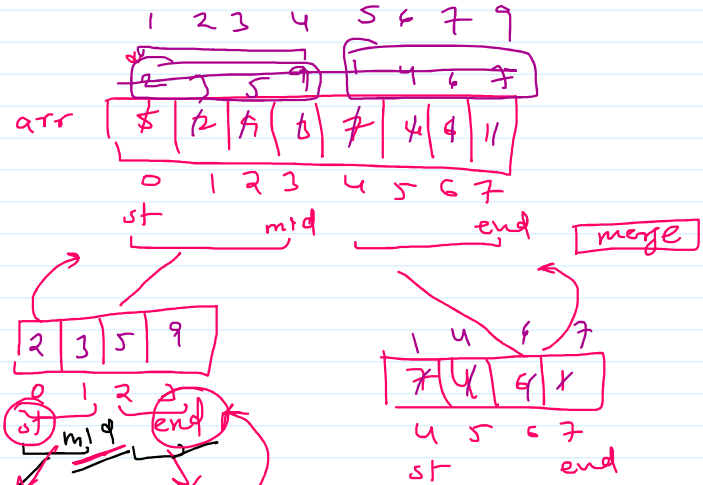


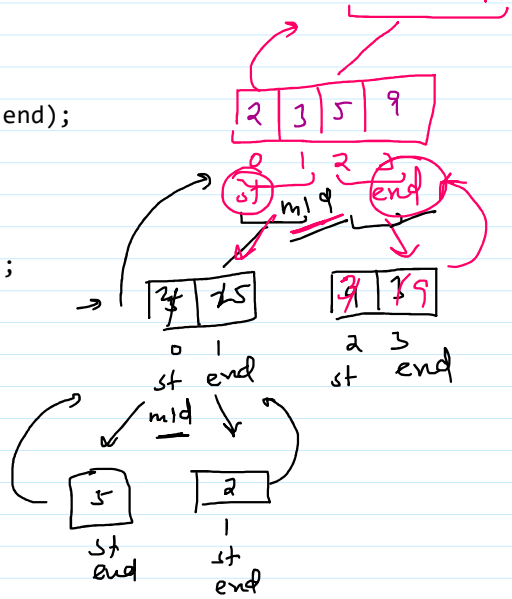

```
#include<iostream>
using namespace std;
void mergeTwoSortedParts(int arr[], int st, int mid, int end)
{
    int temp[end-st+1];
    int i,j,k;
    for(i=st,j=mid+1,k=0;i<=mid && j<=end ; k++)
    {
        if(arr[i] < arr[j])
            temp[k]=arr[i++];
        else
            temp[k] = arr[j++];
    }
```

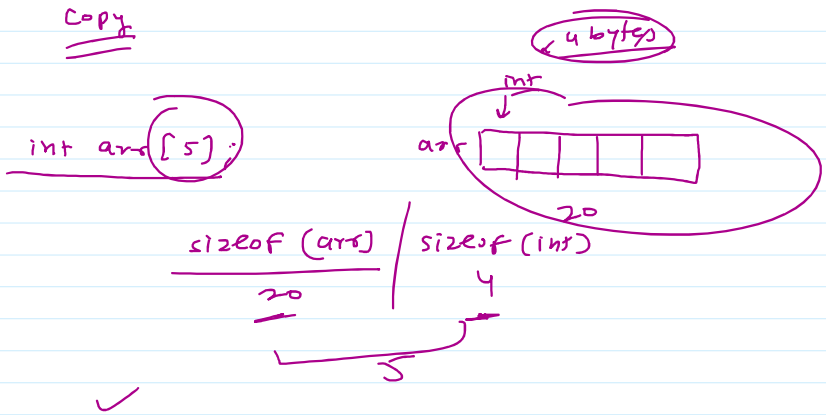


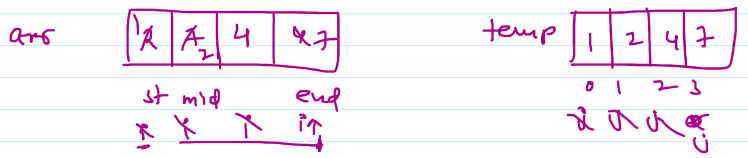
```
        else
            temp[k] = arr[j++];
    }
    while(i<=mid)
    {
        temp[k++] = arr[i++];
    }
    while(j<=end)
    {
        temp[k++] = arr[j++];
    }
    for(i=st,j=0 ; i<=end ; i++,j++)
        arr[i]=temp[j];
}
void mergeSort(int arr[], int st, int end)
{
    if(st>=end)
        return;
    int mid = (st+end)/2;
    mergeSort(arr,st,mid);
    mergeSort(arr,mid+1,end);
    mergeTwoSortedParts(arr,st,mid,end);
}
int main()
{
    int arr[]={5,2,9,3,7,4,6,1};
    int n = sizeof(arr)/sizeof(int);
    mergeSort(arr,0,n-1);
    for(int i=0 ; i<n; i++)
        cout<<arr[i]<<" ";
    return 0;
}
```





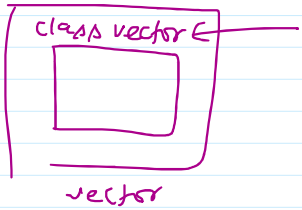




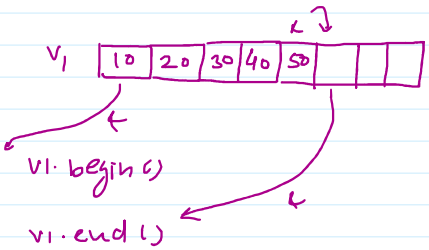


Vector → Dynamic Array :-

#include <vector>

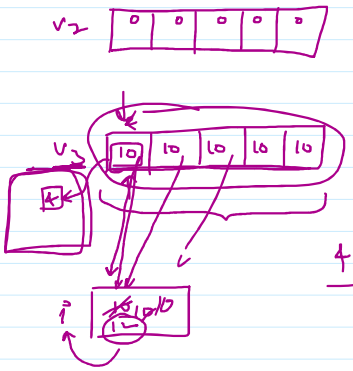


```
vector <int> v1;
v1.push_back(10);
v1.push_back(20);
v1.push_back(30);
v1.push_back(40);
v1.push_back(50);
v1.size(); ⟶ 5
v1.capacity() ⟶ 8
```




```
vector <int> v2(5);
```
size

```
vector <int> v3(5, 10);
```



for each loop :-

```
for (int i : v3)
    cout<<i;
```

auto keyword for stl

```
for (auto i : v3)
```

```
int i;
auto i = ;
```

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    // vector<int> v1={10,20,30,40,50};
    // vector<int> v1;
    // vector<int> v1(5);
    vector<int> v1(5,10);
    v1.push_back(20);

    cout<<"Size = "<<v1.size()<<endl;
    for(int i:v1)
        cout<<i<<" ";
    return 0;
}
```

Iterator (pointer)

begin → add of first element

end → add of last+1 element

STL

1. Iterator → (pointer)
2. Container
   - vector, array
   - list
   - stack
   - queue

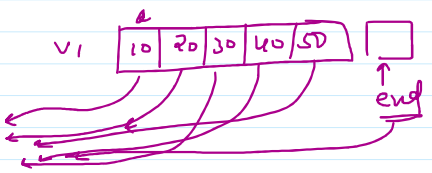
- map
- set

③ Algorithm

④ Functors

① iterators :-

vector<int> v1 = {10, 20, 30, 40, 50};



vector<int>:: iterator i;

for (i = v1.begin() ; i != v1.end() ; i++)
{
    cout << *i;
}

10 20 30 40 50

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> v1={10,20,30,40,50};
    vector<int>::iterator i;
    for(i=v1.begin() ; i!=v1.end() ; i++)
    {
        *i += 2;
        cout<<*i<<" ";
    }
    cout<<endl<<endl;
    vector<int>::reverse_iterator j;
    for(j=v1.rbegin() ; j!=v1.rend() ; j++)
    {
        *j -= 2;
        cout<<*j<<" ";
    }
    cout<<endl<<endl;
    vector<int>::const_iterator k;
    for(k=v1.cbegin() ; k!=v1.cend() ; k++)
    {
        // *k += 2;
        cout<<*k<<" ";
    }
    cout<<endl<<endl;
    vector<int>::const_reverse_iterator l;
    for(l=v1.crbegin() ; l!=v1.crend() ; l++)
    {
        // *l += 2;
        cout<<*l<<" ";
    }
    cout<<endl<<endl;
    //auto keyword
    for(auto x=v1.crbegin() ; x!=v1.crend() ; x++)
        cout<<*x<<"  ";
    return 0;
}
```

```
#include<iostream>
#include<vector>
using namespace std;
void output(vector<int> &r)
{
    for(auto i:r)
        cout<<i<<" ";
    cout<<endl;
}
int main()
```

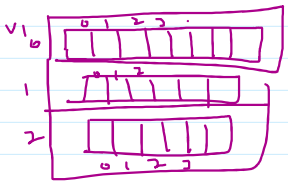
```
{
    vector<int> v1={10,20,30,40,50};
    output(v1);
    v1.push_back(60);
    output(v1);
    v1.pop_back();
    output(v1);
    v1.insert(v1.begin(),100);
    output(v1);
    v1.insert(v1.begin()+2,200);
    output(v1);

    v1.erase(v1.begin());
    output(v1);
    v1.erase(v1.begin()+1,v1.begin()+4);
    output(v1);
    cout<<v1.at(1)<<endl;
    cout<<v1[1]<<endl;
    v1.clear();
    cout<<v1.empty()<<endl;
    return 0;
}
```

2-D Array (Nested vector)

vector <int>

vector < vector <int> > v2;



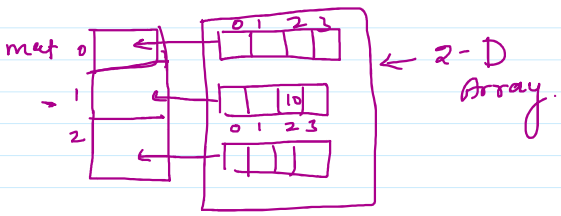
vector < string > v3;

3X4

method-1

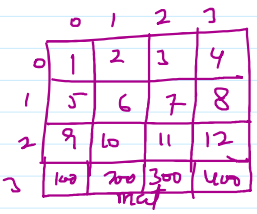
vector < vector <int> > mat(3, vector <int>(4))

mat[1][2] = 10;



method-2

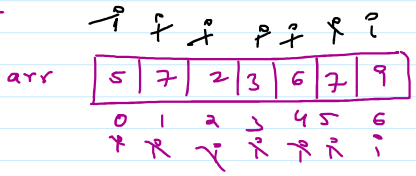
vector < vector <int> > mat = { { 1,2,3,4 },
{ 5,6,7,8 },
{ 9,10,11,12 }
};



vector<int> temp = { 100, 200, 300, 400 };

mat.push_back(temp);

Counting Sort :-



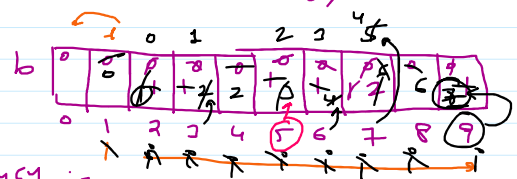
n = 7

max = 9

time → O(n + max) (n = no. of elements, max = max elem)

space → O(n + max)

```
int b[max+1] = {0};
```



step-1 count frequency :-

```
for(i=0; i<n; i++)
{
    b[arr[i]]++;
```

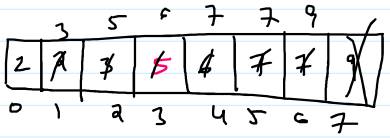
step-2 left sum of b array

```
for(i=1; i<=max; i++)
{
    b[i] += b[i-1];
```

step-3 match index :-

```
int c[n]
for(i=0; i<n; i++)
{
    c[b[arr[i]]-1] = arr[i];
    b[arr[i]]--;
```

c



step-4 copy from c to arr →

```
for(i=0; i<n; i++)
{
    arr[i] = c[i];
}
```

```
#include<iostream>
#include<vector>
using namespace std;
void countingSort(vector<int> &arr)
{
    int max = 0;
```

```
    for(int i=0;i<arr.size();i++)
    {
        if(arr[i] > max )
            max = arr[i];
    }
    int b[max+1]={0};
    //count frequency
    for(int i=0;i<arr.size();i++)
    {
        b[arr[i]]++;
    }
    //left sum
    for(int i=1;i<=max;i++)
        b[i] += b[i-1];
    //match index
    int c[arr.size()];
    for(int i=0;i<arr.size();i++)
    {
        c[b[arr[i]]-1]=arr[i];
        b[arr[i]]--;
    }
    //copy from c to arr
    for(int i=0;i<arr.size();i++)
        arr[i] = c[i];
}
int main()
{
    vector<int> v1 = {5,1,2,6,4,3,7,5,4};
    countingSort(v1);
    for(int i:v1)
        cout<<i<<" ";
    return 0;
}
```