DAY 11

06 May 2024 01:01 PM

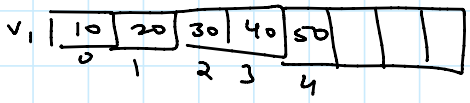
DSA

STL vector | Array | stack | Queue | list | set | map

set :-

vector :- Dynamic Array

```
vector<int> v1;
v1.push_back(10);
v1.push_back(20);
v1.push_back(30);
v1.  ————————  (40)
v1.  ————————  (50)
```

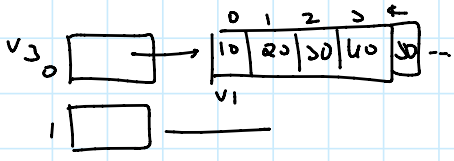


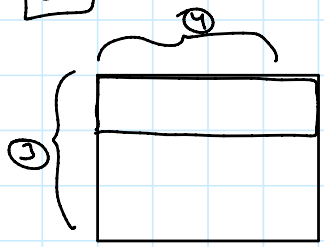
size 5

capacity 8

```
vector<float> v2;
```

```
vector<vector<int>> v3;
v3.push_back(v1);
cout<< v3[0][2]; → 30
```






```
vector<vector<int>> v4(3, vector<int>(4));
```

Row

colp

stack | Queue | array ← static

int a[10] ✓

array <int, 10> a1;

set :→ duplicate elements not allowed

```
#include <set>
```

```
set <int> s1;
```

```
s1.insert (10);
s1.insert (50);
s1.insert (20);
s1.insert (10);
```

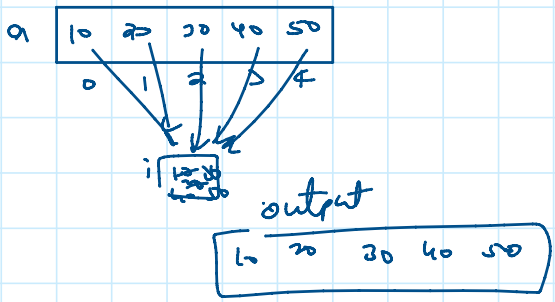
× ×

s1 | 10 | 20 | 50

```
int a[] = {10, 20, 30, 40, 50};
```

for each loop

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




```
int a[5] = {10, 20, 30};

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

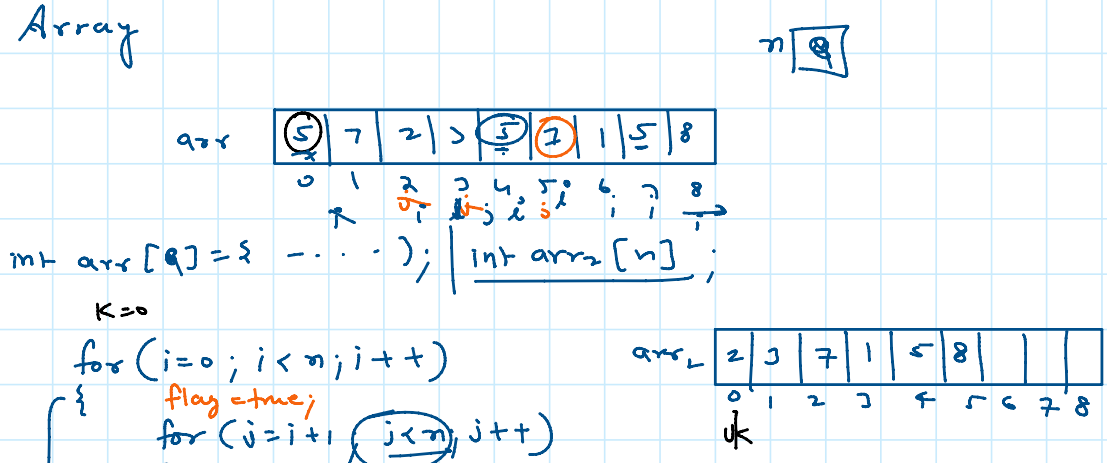
→ 10 20 30 0 0

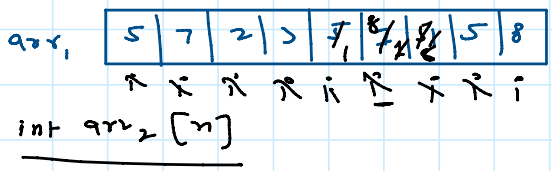
.

```
#include<iostream>
#include<set>
using namespace std;
```

```cpp
void output(set<int> s1)
{
    for(int i:s1)
        cout<<i<<" ";
    cout<<endl;
}
int main()
{
    set<int> s1;
    s1.insert(10);
    s1.insert(50);
    s1.insert(20);
    s1.insert(40);
    s1.insert(10);
    s1.insert(20);
    output(s1);
    if(s1.find(200)==s1.end())
        cout<<"Not found\n";
    else
        cout<<"Found\n";
    s1.erase(10);
    output(s1);
}
```

W.A.P to Remove duplicate element from an Array





j=0

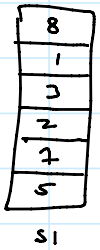
```cpp
for(i=0; i<n; i++)
{
```

int arr2[n]

arr2 | 5 | 7 | 2 | 3 | 1 | 8 | ——x——

i i i i i j

set<int> s1;



Time (n)

Space (< n)

```
for(i=0; i<n; i++)
{
    if (s1.find(arr1[i]) == s1.end())
    {
        s1.insert(arr1[i]);
        arr2[j++] = arr1[i];
    }
```

Multiset :- duplicate elements allow

#include<set>

multiset<int> s1;

s1.insert(10);

s1.insert(20);

s1.insert(10);

s1 | 10 | 10 | 20

.

```
#include<iostream>
#include<set>
using namespace std;
void output(multiset<int> s1)
{
    for(int i:s1)
        cout<<i<<" ";
    cout<<endl;
}
int main()
{
    multiset<int> s1;
    s1.insert(10);
    s1.insert(50);
    s1.insert(20);
    s1.insert(40);
    s1.insert(10);
    s1.insert(20);
    output(s1);
    if(s1.find(20)==s1.end())
        cout<<"Not found\n";
    else
```

```
        cout<<"Found\n";
    s1.erase(10);
    output(s1);
}
```

→ template < typename T1, typename T2 >
class pair
{
T1 first;
T2 second;

class data
{ public:
int a;
float b;
};

d1
a 10
b 7.5

data d1;
d1.a = 10;
d1.b = 7.5f;

data d2;
d2.a = 100
d2.b = 75.25

d2
a 100
b 75.25

map :-

pair

pair <int, float> P1;

P1
int → first
float → second

pair <float, string> P2;

P2
float → first
string → second

```
#include<iostream>
#include<set>
using namespace std;
void output(multiset<int> s1)
{
    for(int i:s1)
        cout<<i<<" ";
    cout<<endl;
}
int main()
{
    pair<int, int> p1;
    p1.first=101;
    p1.second=8005;
    cout<<p1.first<<' '<<p1.second<<endl;
    pair<int,float> p2(1001,25.52);
    cout<<p2.first<<' '<<p2.second<<endl;

}
```

Map → sort According to Key

Key must be unique → (first) key

value (second)

(int) roll per (float)

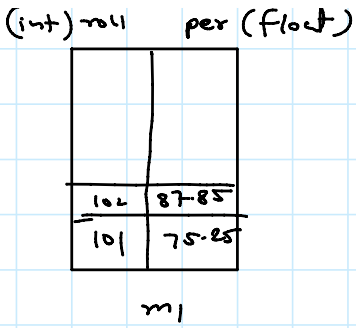
map < int, float > m1;

Map → Key

```
map<int, float> m1;

m1.insert(pair<int, float>(101, 75.25));

m1[102] = 87.85;
   ↑        ↑
  key     value
```




```
#include<iostream>
#include<map>
using namespace std;
void output(map<int,float> m1)
{
    for(pair<int,float> i:m1)
        cout<<i.first<<" "<<i.second<<endl;
    cout<<endl;
}
int main()
{
    map<int,float> m1;
    m1.insert(pair<int,float>(101,75.28));
    m1.insert(pair<int,float>(103,95.25));
    m1.insert(pair<int,float>(101,65.54));
    m1.insert(pair<int,float>(102,74.14));
    m1[107]=85.14;
    m1[105]=45.54;
    m1[101]=35.25;      //edit
    output(m1);
    map<int,float>::iterator i=m1.find(101);
    if(i==m1.end())
        cout<<"Not found";
    else
        cout<<"Found Value = "<<i->second;
}
```

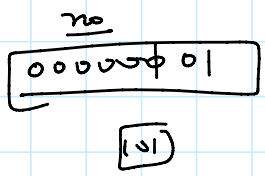
Multi map :— duplicate values allowed

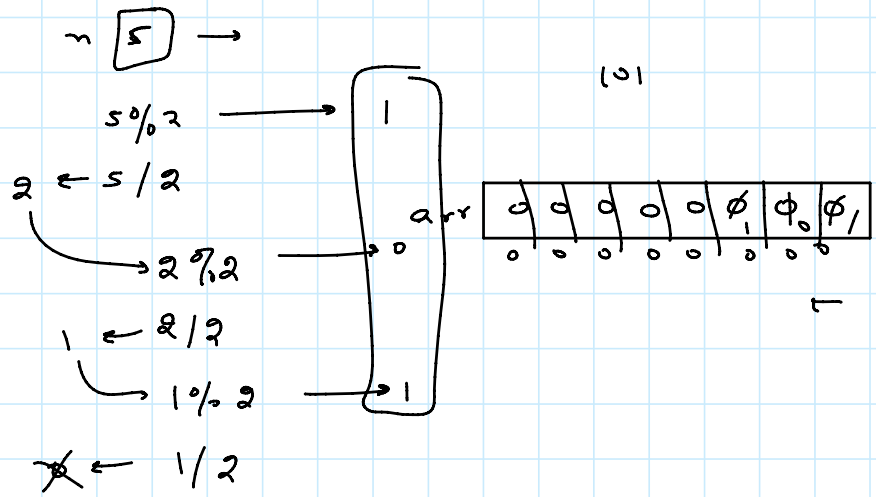
```
#include<iostream>
#include<map>
using namespace std;
void output(multimap<int,float> m1)
{
    for(pair<int,float> i:m1)
        cout<<i.first<<" "<<i.second<<endl;
    cout<<endl;
}
```

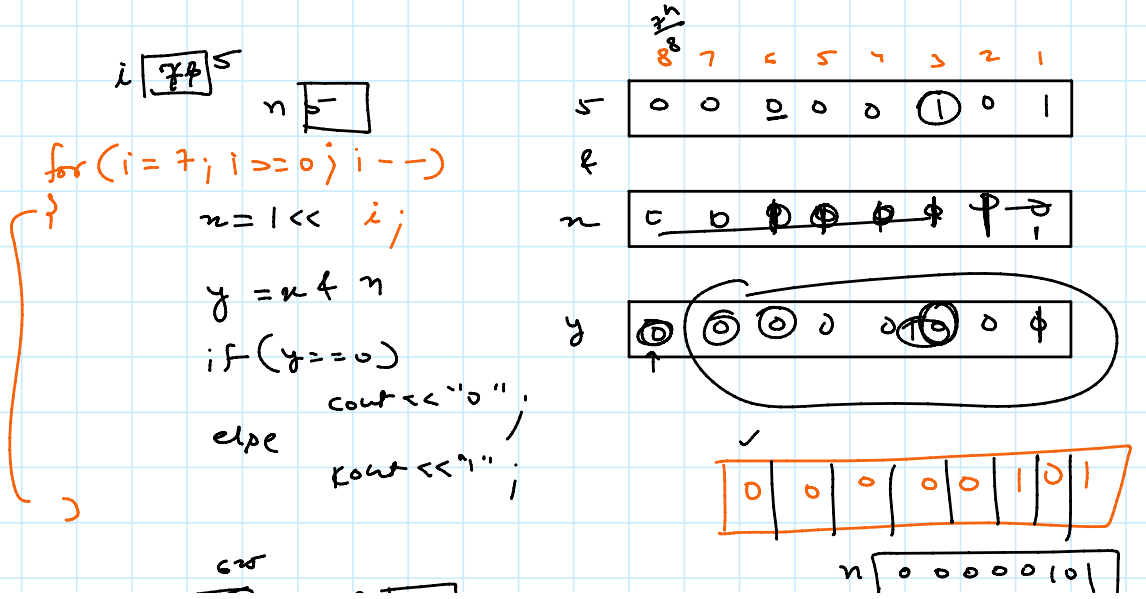
```cpp
int main()
{
    multimap<int,float> m1;
    m1.insert(pair<int,float>(101,75.28));
    m1.insert(pair<int,float>(103,95.25));
    m1.insert(pair<int,float>(101,75.28));
    m1.insert(pair<int,float>(102,74.14));
    output(m1);
    map<int,float>::iterator i=m1.find(101);
    if(i==m1.end())
        cout<<"Not found";
    else
        cout<<"Found Value = "<<i->second;
}
```

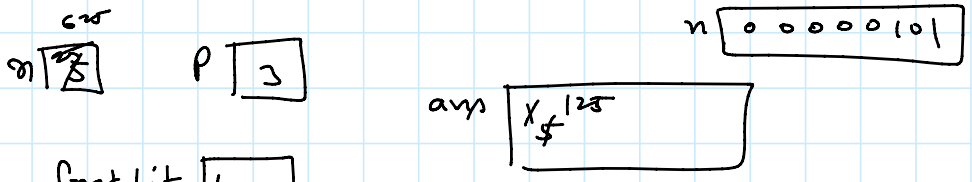
<< >> & | ^ ~

print Binary no of a decimal no

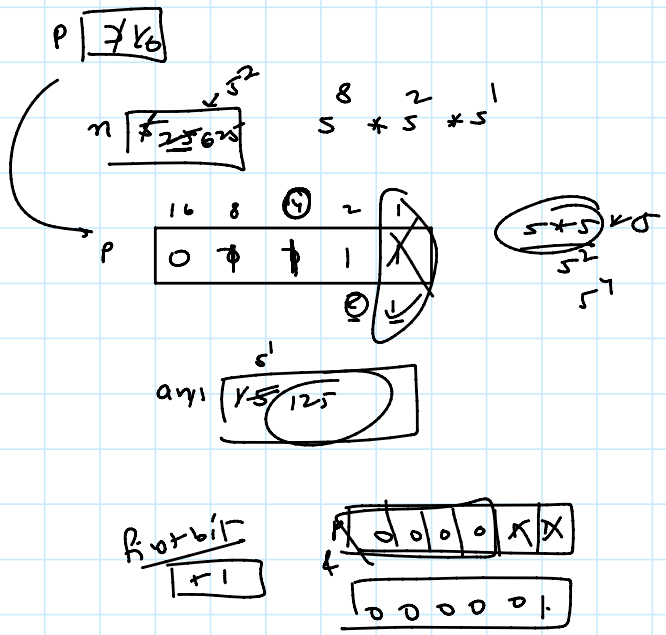









```cpp
#include<iostream>
#include<map>
using namespace std;
int mypower(int n, int p)
{
    int ans=1;
    while(p)
    {
        int first_bit=p&1;
        if(first_bit)
        {
            ans *= n;
        }
        n=n*n;
        p=p>>1;
    }
    return ans;
}
int main()
{
    int n=5,p=3;
    cout<<mypower(n,p);
}
```



Time Complexity :-

for (1 ——— n)

O(n)

```
for (1 ——— n)
{
    =
    =
}
```

$O(n)$

```
for (1 ——— n)                    n
{                                +
    for (1 ——— n)          n
    {
    }
}
```

$n * n$

$O(n^2)$

```
for (1 ——— n)                    n
{
    for (1 ——— n)                n
    {
        for (1 ——— n)            n
        {
        }
    }
}
```

$\rightarrow O(n^3)$

```
for (1 ——— n)
{
    for (1 ——— n)
    {
        .
        .
        .
        n
    }
}
```

$O(n^n)$

n 8

```
for (1 ——— log2 n)
{
    =
}
```

$O(\log_2 n)$

n [8] ⟶ 3

$8 = 2^p$

$p = \log_2 8n$

2
4
8
16
32

```
for( 1 ——— n )
{
    for( 1 —— log2n )
    {
    }
}
```

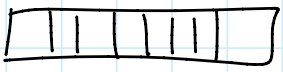
$O(n \log_2 n)$

```
if( ——— )
——
else ——
```

⟶ $O(1)$

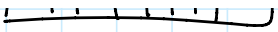
$O(1)$
$O(\log_2 n)$
$O(n)$
$O(n \log_2 n)$
$O(n^2)$
$O(n^2 \log_2 n)$
$O(n^3)$
$O(n^n)$

Space Complexity :-

problem → 

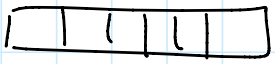
space
$O(1)$

no extra memory



no extra memory

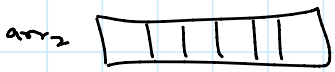
O(1)



n

space

O(n)



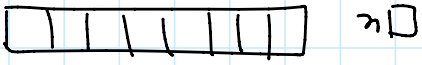
n

n

space

$O(n^2)$

arr[n][n] ← Extra



n

space

O(nm)

arr[n][m]

```
for(1 ——— n)
{
    for(1 ——— m)
    {
    }
}
```

time

O(n·m)