Printing all Subsequences

→ a contigous / non-contigous sequences which follows the order.

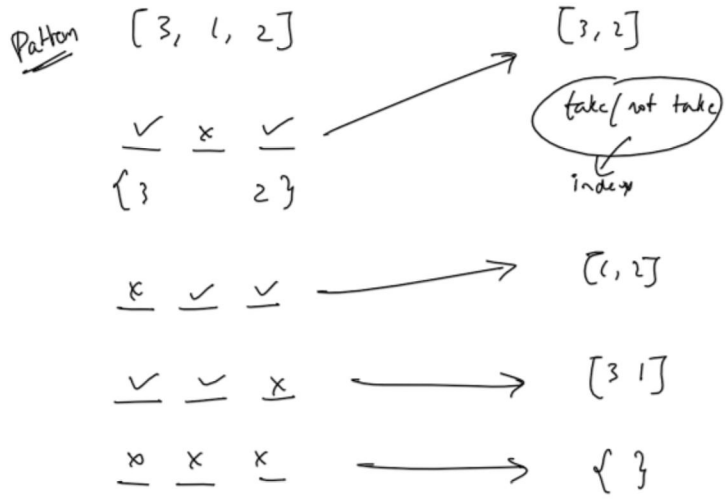
arr → [3, 1, 2]

n = 3, 8 subseq.

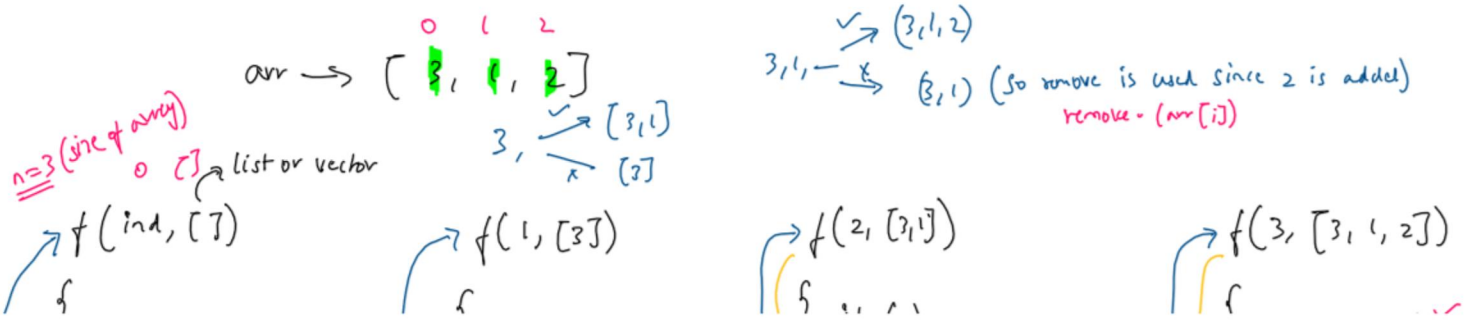
{} ✓
3 ✓
1 ✓
2 ✓
3 1 ✓
1 2 ✓
3 2 ✓ (non-contigous)
3 1 2 ✓

3 2 1 ✗ (Not subsequence because not following an order)

+ A subarray can be Subsequences.



Structure of code / pattern :



```
f(ind, [])
{
  if (ind >= n)
    print([])
    return;
  [].add(arr[i])  [3]
  f(ind+1, []); // take
  [].remove(arr[i]);
  f(ind+1, []); // not take
}
main()
{
  arr → [3,1,2]
  f(0, [])
}
```

```
f(1, [3])
{
  if() x
  [3].add(arr[i])  [3,1]
  f(ind+1, []);
}
```

```
f(2, [3,1])
{
  if() x
  [3,1].add(arr[i])  [3,1,2]
  f(ind+1, [])
}
```

```
f(3, [3,1,2])
{
  if(ind >= n)
    print(3,1,2)
    return
}
```

[3,1,2].remove(arr[2])

[3].remove(arr[1])

[3,1]

```
f(ind+1, [])
}
```

f(2, [3])

```
f(2, [3])
{
  Same thing
}
```

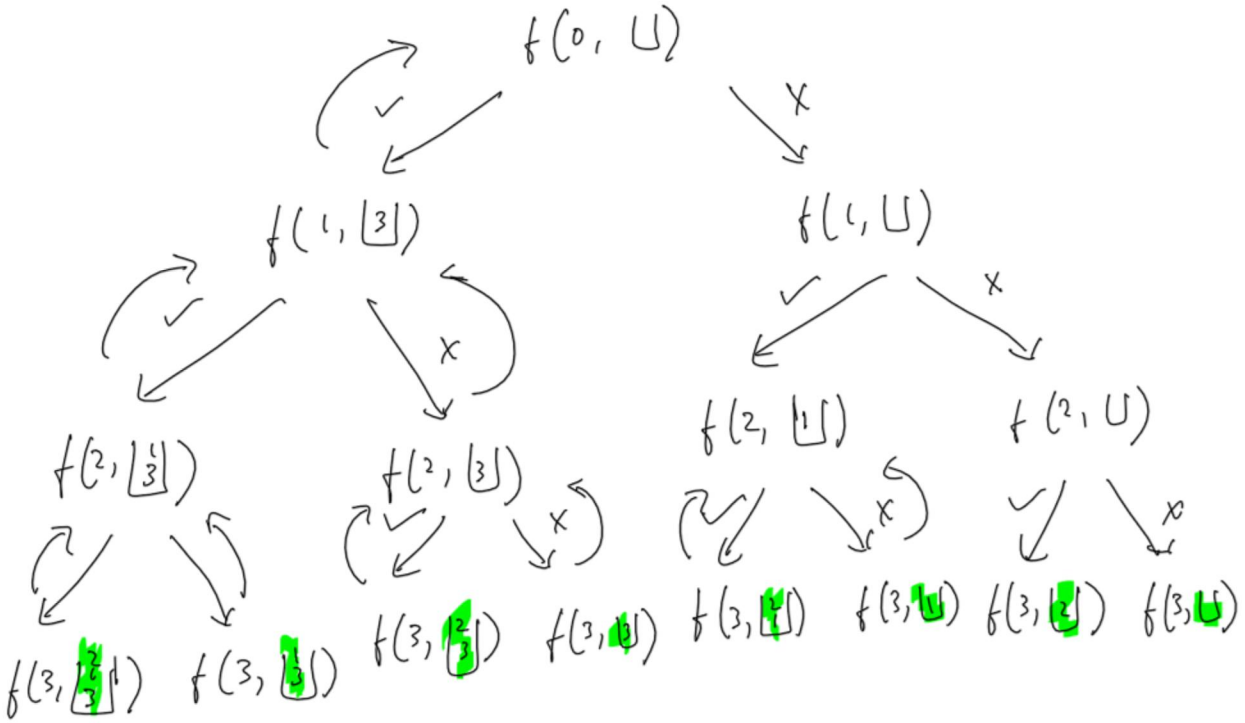
```
f(3, [3,1])
{
  if(ind >= 3)
    print() // 3,1
    return
}
```

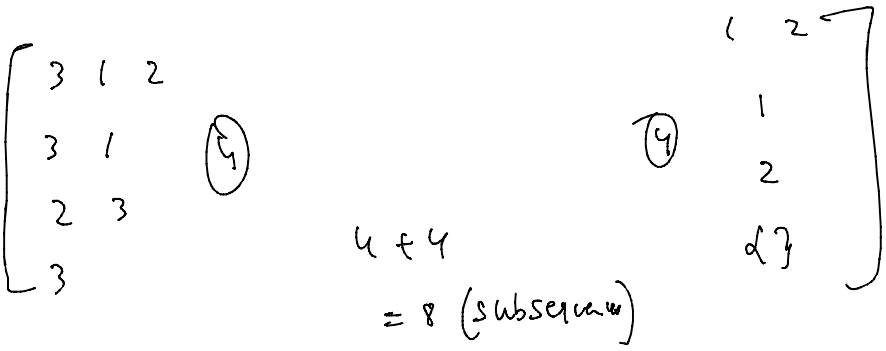
Output

3, 1, 2

3, 1

{3, 1, 2}






```cpp
#include <bits/stdc++.h>
using namespace std;
void printF(int ind, vector<int> &ds, int arr[], int n){
    if(ind == n){
        for(auto it: ds){
            cout << it << " ";
        }
        cout << endl;
        return;
    }
    // take or pick the particular index into the subsequences
    ds.push_back(arr[ind]);
    printF(ind+1, ds, arr, n);
    ds.pop_back();

    // not pick, or not take condition, this element is not added
    // to your subseq
    printF(ind+1, ds, arr, n);
}
int main(){
    int arr[] = {3, 1, 2};
    int n = 3;
    vector<int> ds;
    printF(0, ds, arr, n);
    return 0;
}
```

Remove

```
5
1 2 3 4 5
```

output.in

```
3 1 2
3 1
3 2
3
1 2
1
2
```

T.C ⇒ $O(2^n)$

S.C ⇒ $O(n)$