CLASS 46

30 March 2023 21:08

CLASS-46

* Binary search Using Recursion

(s - e) → element findout ?

(0 - n-1) (x)

ar = [1, 2, 3, 4, 5, 6, 7]

x = 2

Problem statement ⇒

1) Find index of (x) in the whole array

0 - (n-1)

2) Subproblem ?

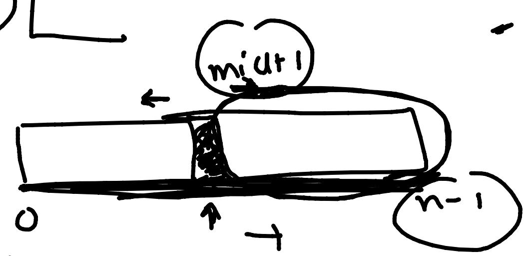
(i, j)

(i - j)

(0, n-1) use Binary search → Mid

```
int s = 0;
int e = n-1;
while(s <= e)
{ int mid = (s+e)/2;
  if(a[mid] == x)
    return mid;
  else if(a[mid] < x)
  { mid+1;
  }
  else
  { mid-;
  }
}
```

```
(int left, int right)
{   if(left <= right)
    { int mid = (left+right)/2;
      if(a[mid] == x)
      { return mid;
      }
      else if(a[mid] < x)
      { // int ans = binary search(mid+1, right);
```



Find x in right
(mid+1) → (n-1)

Recursion(int n)
{
// some work you should do
// some work from recursion

```
{ //
int ans = binarysearch(mid+1, right);
if (ans != -1)
    return ans;
}
else if ( a[mid] > x )
{
    int ans = binarysearch(left, mid-1);
        if (ans != -1)
        { return ans;
        }
}

return -1;
}
```

// some work from recursion

mid, right | l, R

Recursion → Redundant work

```
int bs (vector<int> a, int l, int r, int x)
{
    if( l > r )
    { return -1;        // Base Case
    }
    else{
        int mid = (l+r) / 2;
        if ( a[mid] == x )      8 == 7 ×
            return mid;         =①
        else if ( a[mid] < x )          8 < 7 ×
        { int ans = bs( a, mid+1, right, x );
            if (ans != -1)
            return ans;                 (-1)
        }
        else
        { int ans = bs( a, l, mid-1, x );
          if (ans != -1)
```

0+a = 4

5

(l, r)

(mid+1) - right

f()
{ f()
}

f()
{
return
}

left

mid=3

ans=1

mid=3

int ans = DSL

if (ans != -1)

return ans;

ans=1

return -1;

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

(0, n-1)

mid = 2/2 = 1

=> 2

bs(a, 0, 5, 2)

bs(a, 0, 2, 2)

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

=> 7

n

bs(a, 0, 9, 7)

=> 11

mid = (7+6)/2 = 5

bs(a, 5, 9, 7)

n/2

bs(a, 5, 6, 7)

n/4

bs(a, 6, 6, 7)

n → n/2 → n/4 ---- → 1

Steps

$\frac{n}{2^k} = 1$ => $k = \log_2 n$

T.C = O(log n)

=> No of steps * TT in each

S.C = O(log2n)

Print all subsequence Using Recursion

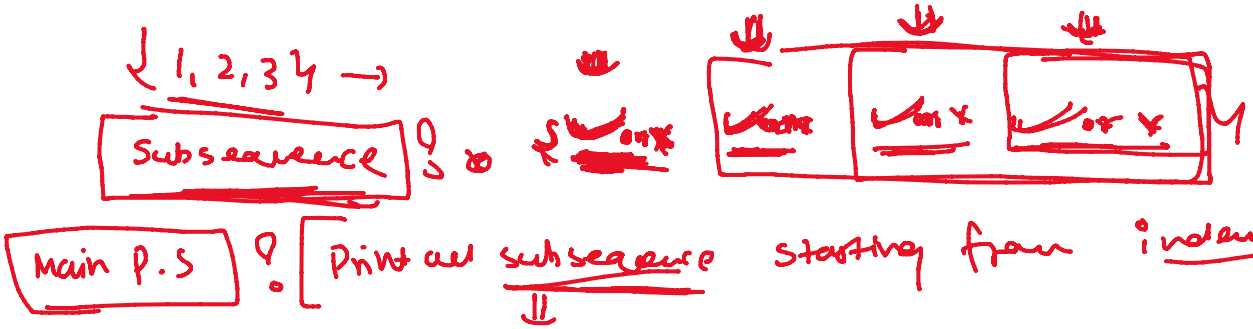
Subarray

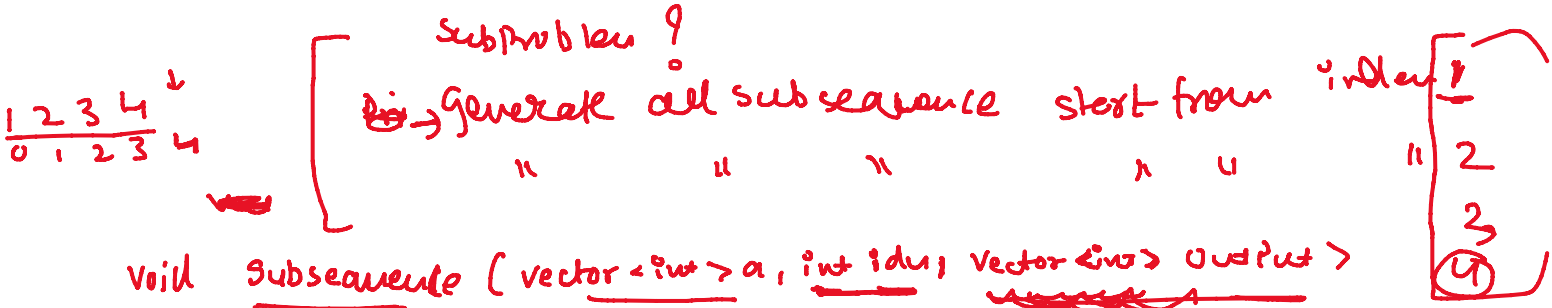
1, 2, 3, 4

(2, 4)

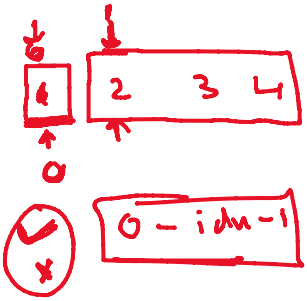
Bit Masking

{1, 2, 3, 4} => {2, 4} + {_ _ _ 4}



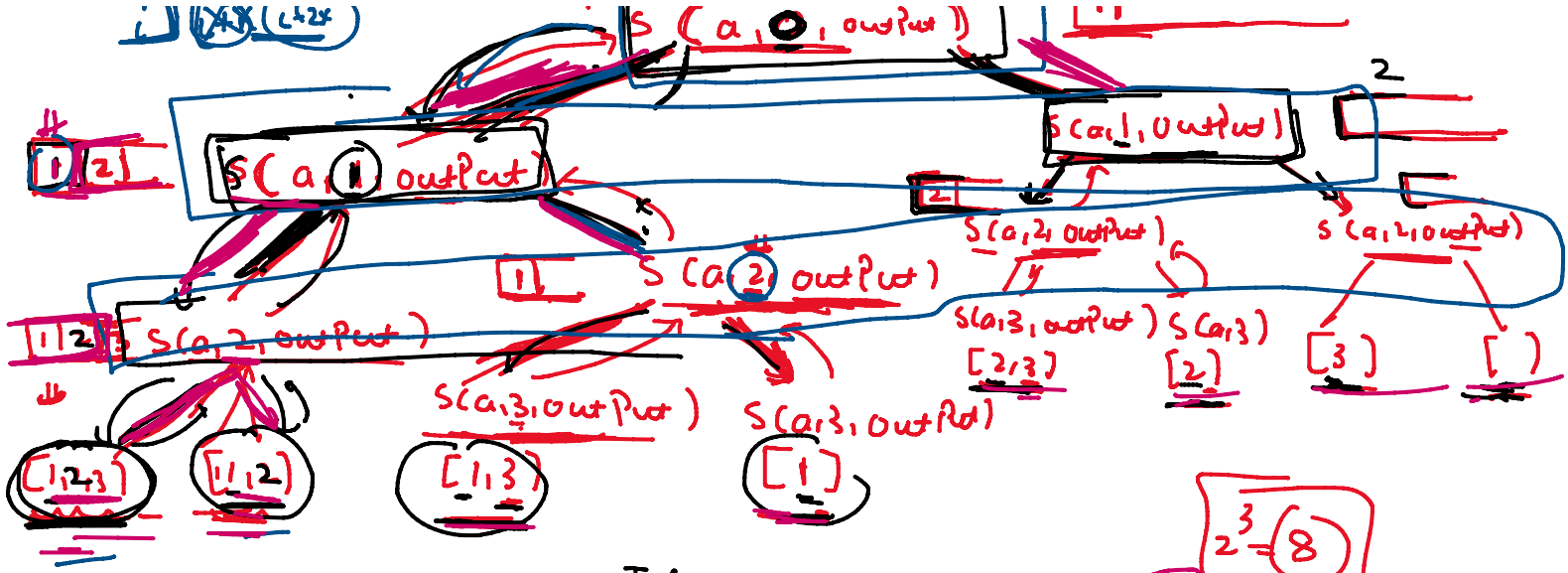



```
Void Subsequence ( vector<int> a, int idx, vector<int> output )
{
    int n = a.size();
    if (idx == n)
    { // Base Case
      for(auto i: output)
          cout << i << " ";
      return
    }
```




```
//2 cases
//Case-1 include element (idx)
output.push_back(a[idx]);
Subsequence(a, idx+1, output);
//case 2 → doesn't include element idx
output.pop_back();
Subsequence(a, idx+1, output);
}
```

(a, 3, output)



T.C =

$2^0 + 2^1 + 2^2 + 2^3 + \cdots 2^n$

$2^3 = 8$

Total states

$= 2^n + O(1) \approx O(2^n)$