Q Power → BF, optimized, most optimized.
Q Remove duplicates from a string
H.W Q find first and last occurance
Q Binary Search ✓
Q Tiling Problem ★★★ chocolate

✓ Linear Search

find last occurance of 5
[5, 3, 4, 5, 9, (5), 4] ← index.

Imp ⇒ Binary Search — Easy, Medium, Hard

Binary Search

O(log N)

★ 1) Sorted — search → 6

1 2 3 (6) 8 9 → O(n)

H.W ★ LinearSearch (using recursion)

★ BS

s e
1 2 3 6 8 9
0 1 2 3 4 5

while (for-loop)

we need it in all calls — (start, end)

Joh bhi cheej har call ke saath needed hoti hai → Parameter

```
public int BS (arr[], int key, int start, int end){
    if(start > end){
        return -1;
    }
    int mid = s+e/2;            mid = 2
    if( arr[mid] == key){
        return mid;
    }
    if (arr[mid] > key){
        BS(arr, key, start, mid-1);
    }
    else{
        BS(arr, key, mid+1, end);
    }
```

Power x^n

2^{10} ↓ 2×2^9 ↓ 2×2^8 ↓ 2×2^7 ↓ 6 ↓ 2×2^5 ↓ 2×2^4 ↓ 2×2^3 ..... 2^1

$2^{10} = 2\times 2\times 2\times 2\times 2$ ...

$(x \times x^{n-1})$

```
power(int x, int n){        2^0 = 1
    if(n==0){
        return 1;
    }
    return x × power(x, n-1);
}
```

(2)

O(n)

stack: main, 10, 9, 8, 7, 6

5/2 ⇒ 2.

$x^7 \to x^3 \times x^3 \times x$.

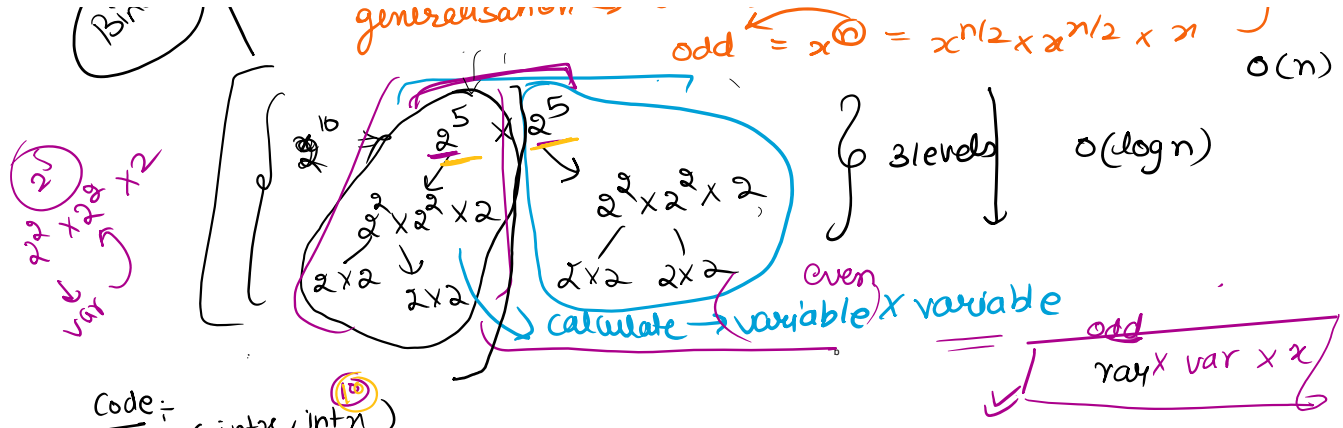
even $x^{12} = x^6 \times x^6$ ↓ $x^3 \times x^3$, $x^3 \times x^3$ ↓ $x^1 \times x^1 \times x$

even: $x^{10} = x^5 \times x^5$ ↓ $x^2 \times x^2 \times x$

Binary

generalisation → even → $x^n = x^{n/2} \times x^{n/2}$
odd = $x^n = x^{n/2} \times x^{n/2} \times x$

O(n)



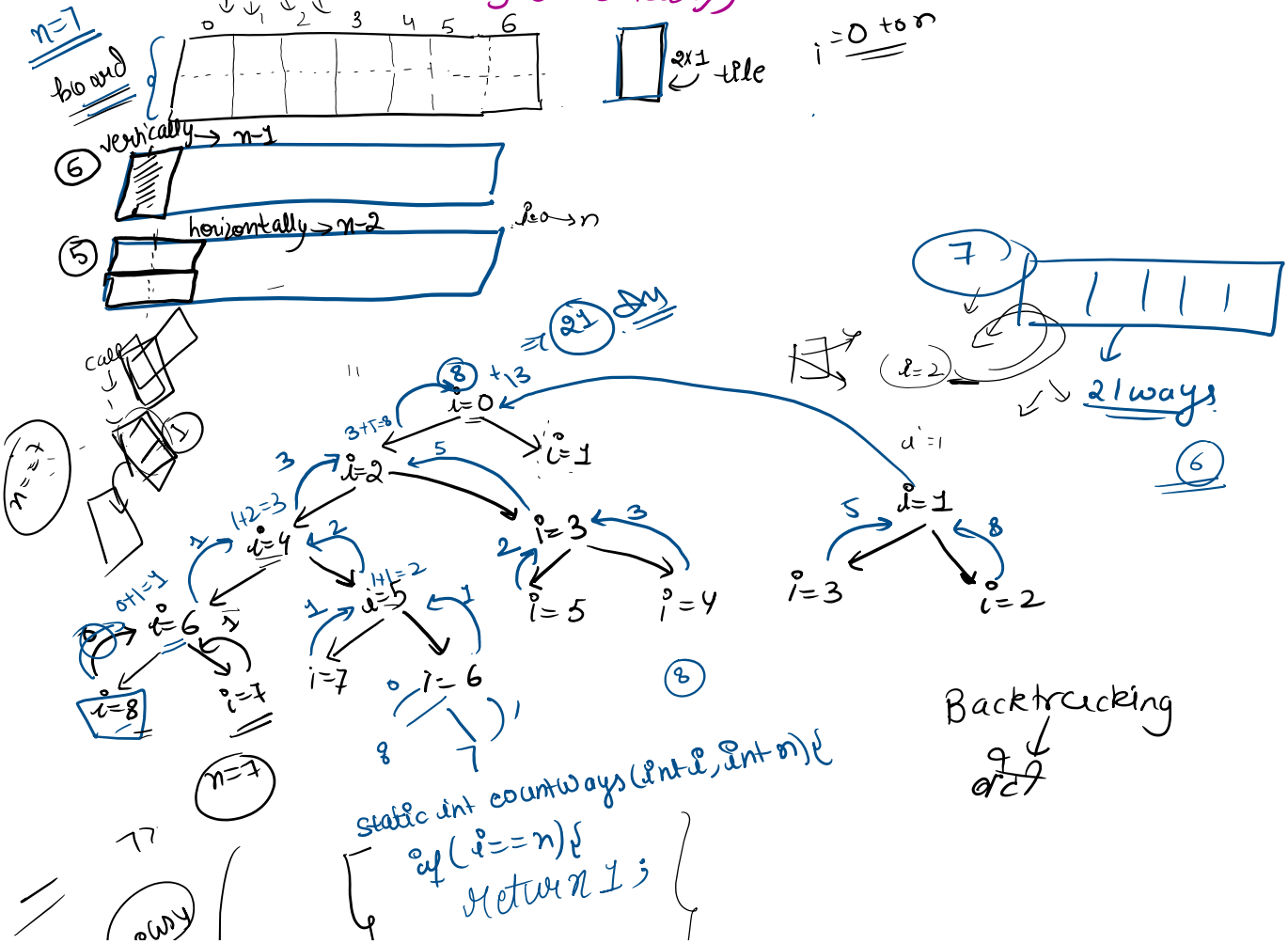
Code :-

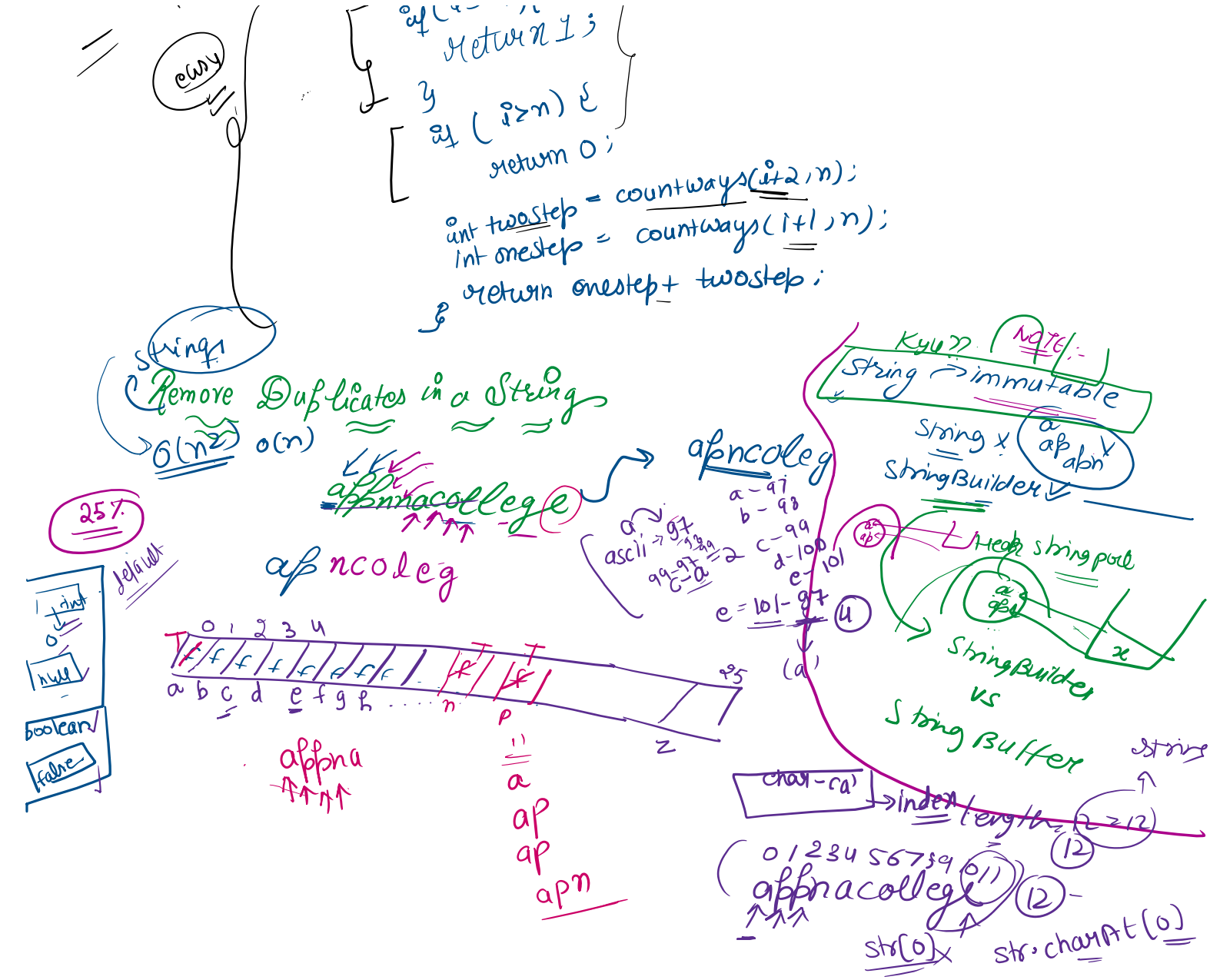
```
op(int x, int n)
if (n==0) {
    return 1;
}
int halfpower = op(x, n/2) * op(x, n/2);
if (n%2 != 0) {
    int halfpower * x;
}
}
```

```
int halfpower = op(x, n/2)
int halfpSq = halfpower * halfpower;
```

Q. Given a "2xn" board and tiles of size "2x1", count the number of ways to tile the given board using the 2x1 tiles. (A tile can either be placed horizontally & vertically)

geeksforgeeks






```
public static String Remove(String str, int idx, StringBuilder s, boolean map[]){
    if(idx == str.length()){
        return s.toString();
    }
    char currchar = str.charAt(idx);
    if(map[currchar - 'a'] == true){
        //duplicate
        Remove(str, idx+1, s, map);
    } else{
        map[currchar - 'a'] = true;
        Remove(str, idx+1, s.append(currchar), map)
    }
}
```

(a)

☆ Reverse a number

12 → 12

☆ findAllIndex of the key

* Reverse a number
* Palindrom
* count no. of zeroes

121 → 121

True / false

* findAllIndex of the key