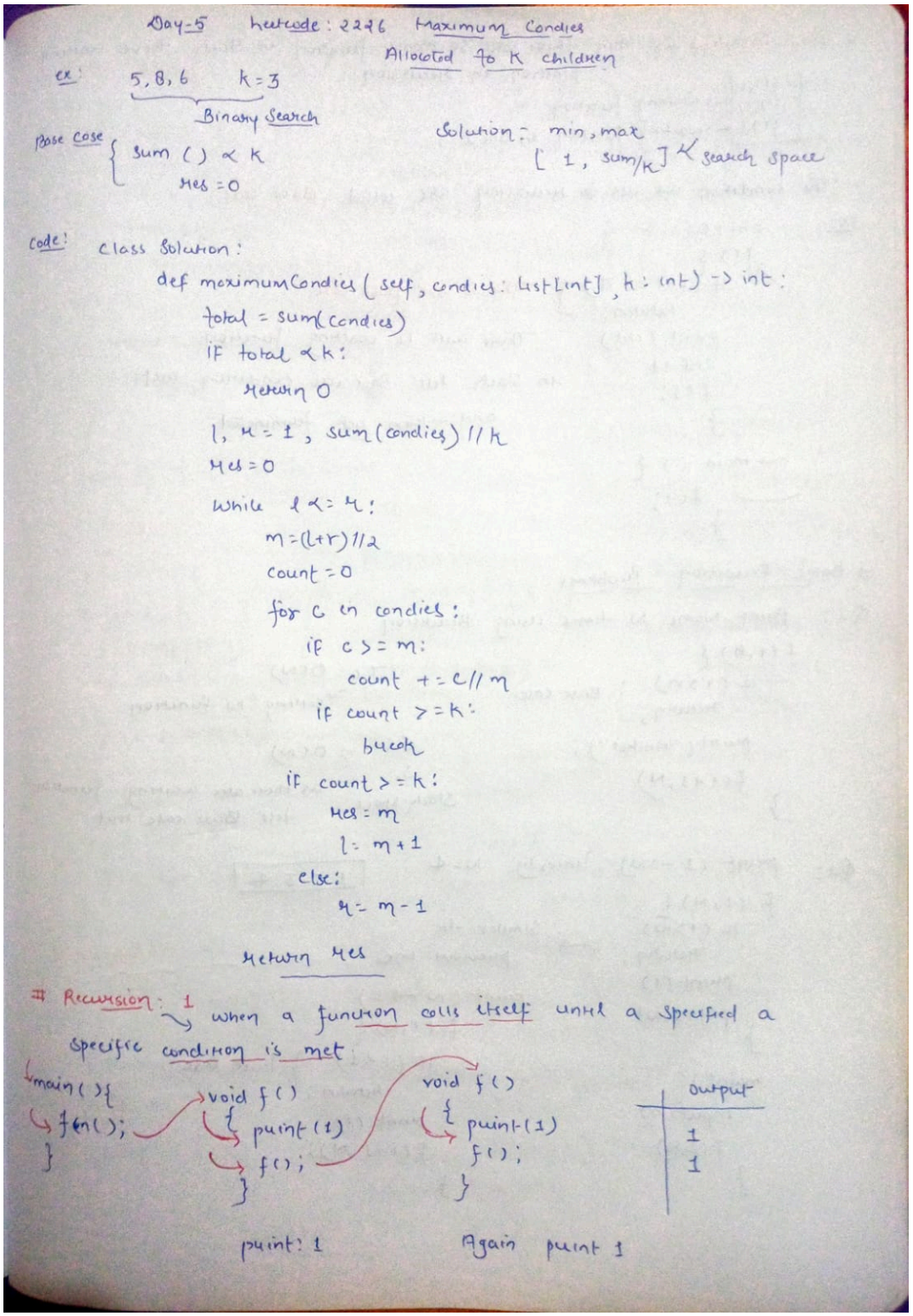
Day_5: DSA



Day-5 leetcode: 2226 Maximum Candies Allocated to K children

ex: 5, 8, 6 k=3

Binary Search

Base case { sum() < k, res = 0

Solution: min, max [1, sum/k] ← search space

code:

```
class Solution:
    def maximumCandies(self, candies: List[int], k: int) -> int:
        total = sum(candies)
        if total < k:
            return 0
        l, r = 1, sum(candies) // k
        res = 0
        while l <= r:
            m = (l+r)//2
            count = 0
            for c in candies:
                if c >= m:
                    count += c // m
                if count >= k:
                    break
            if count >= k:
                res = m
                l = m + 1
            else:
                r = m - 1
        return res
```

Recursion: 1 → when a function calls itself until a specified a specific condition is met

```
main(){
  fen();
}
```

```
void f()
{
  print(1)
  f();
}
```

```
void f()
{
  print(1)
  f();
}
```

output
1
1

print: 1 Again print 1

• Stack Overflow :→ when there are so many function in stack these waiting function in Recursion.

f(),l2 → waiting function
f(),l2 → waiting function
f(),l2 → waiting function in line 2

• The condition we use to Recursion are called Base Case

Ex:

```
cnt=0
f() {
  if (cnt == 4) { return; }   This is our Base Case
  print (cnt)                 There will be waiting function
  cnt++                       in stack till Base case condition met
  f();                        and then get terminated.
}

main () {
  f();
}
```

⇒ Basic Recursion Problems.

Q1: Print Name N times using Recursion

```
f(i,N) {
  if (i>N) { return; }   Base case
  print("Anikat");
  f(i+1,N)
}
```

T.C = O(N) → calling N function

S.C ⇒ O(N) → Stack space, as there are waiting functions till Base case met.

Q2: print (1 → N) linearly N=4

1 2 3 4

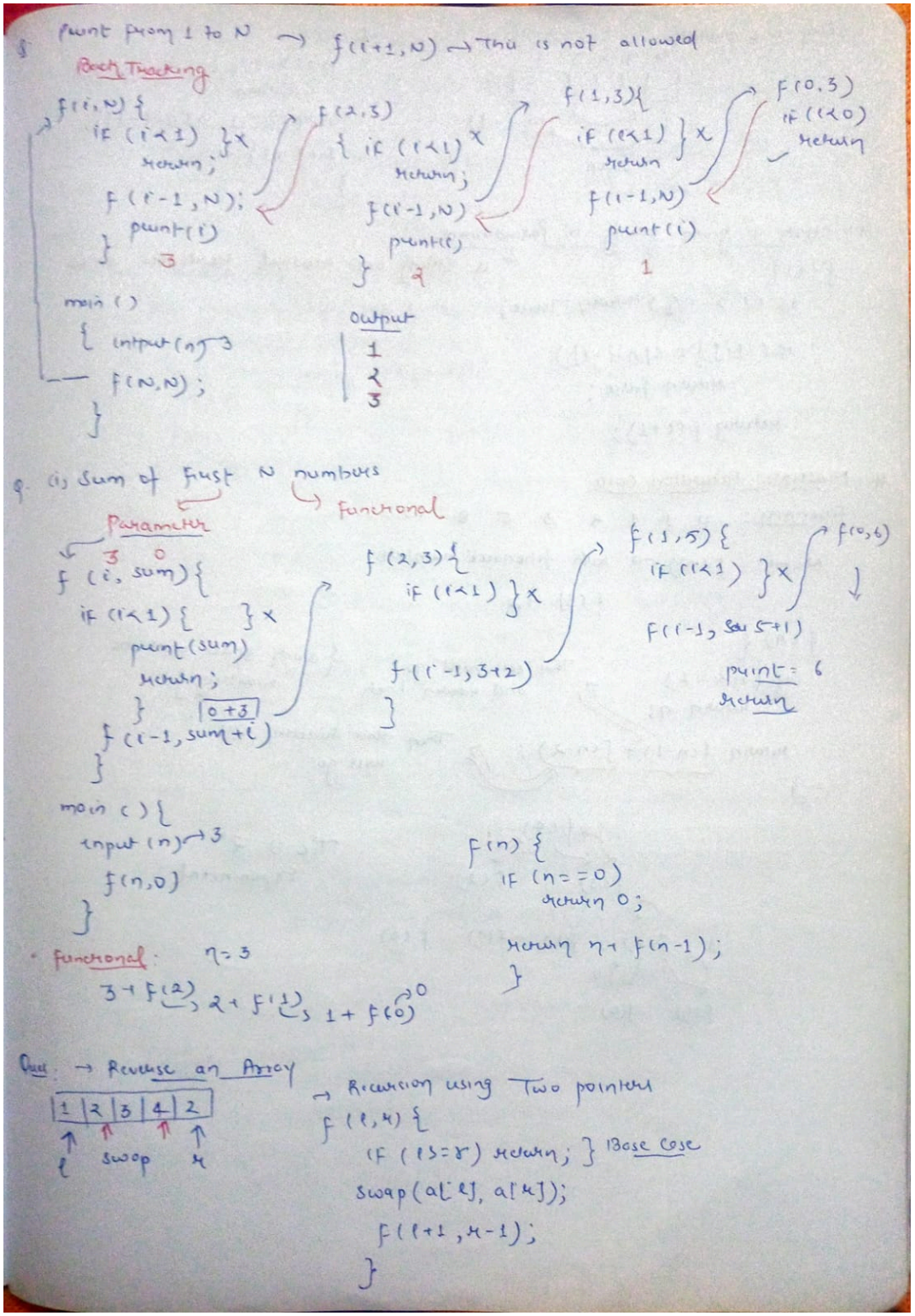
```
f (i,N) {
  IF (i>N)
    return;
  print(i)
  f(i+1,N)
}

main(){
  input(n)
  f(1,N)
}
```

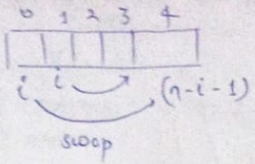
Similar to previous one

• print (N → 1)

```
f (i,N) {
  IF (i<1) { return; }   Base case
  print (i)
  f(i-1,N);
}
```


print from 1 to N → f(i+1,N) → This is not allowed
Back Tracking
f(i,N){
if (i<1) { }x
return;
f(i-1,N);
print(i)
}
3
f(2,3)
{ if (i<1) x
return;
f(i-1,N)
print(i)
} 2
f(1,3){
if (i<1) } x
return
f(i-1,N)
print(i)
1
f(0,3)
if (i<0)
return
main()
{ input(n) 3
f(N,N);
}
Output
1
2
3
q. (i) Sum of first N numbers
Parameter
Functional
3 0
f(i, sum){
if (i<1) { } x
print(sum)
return;
} 0+3
f(i-1, sum+i)
}
f(2,3){
if (i<1) } x
f(i-1, 3+2)
}
f(1,5){
if (i<1) } x
f(i-1, sum 5+1)
print = 6
return
f(0,6)
main(){
input(n) → 3
f(n,0)
}
Functional: n=3
3+f(2), 2+f(1), 1+f(0) → 0
f(n){
if (n==0)
return 0;
return n+f(n-1);
}
Ques → Reverse an Array
1 2 3 4 5
l swap r
→ Recursion using Two pointers
f(l,r){
if (l>=r) return; } Base Case
swap(a[l], a[r]);
f(l+1, r-1);
}

• Using one pointer




```
f(i){
  if (i>=n/2)
    return
  swap(a[i], a[n-i-1]);
  f(i+1);
}
```

Q. Check if given string is Palindrome:
↳ a string on reversal needs the same

```
f(i){
  if (i >= n/2) return True;
  if (s[i] != s[n-i-1])
      return false;
  return f(i+1);
```

Multiple Recursion Calls

Fibonacci: 0 1 1 2 3 5 8 13

N:→ f(N) → Nth fibonacci number

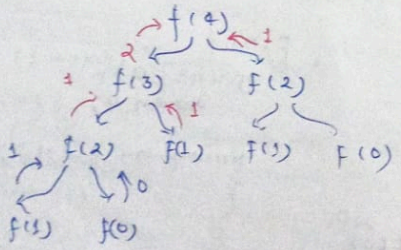
f(3) → 2

```
f(n){
  if (n<=1)
    return n;
  return f(n-1) + f(n-2);
}
```

This call will go and return back

{ stack space will be waiting }

Then this function call will go



T.C → 2^n
Exponential