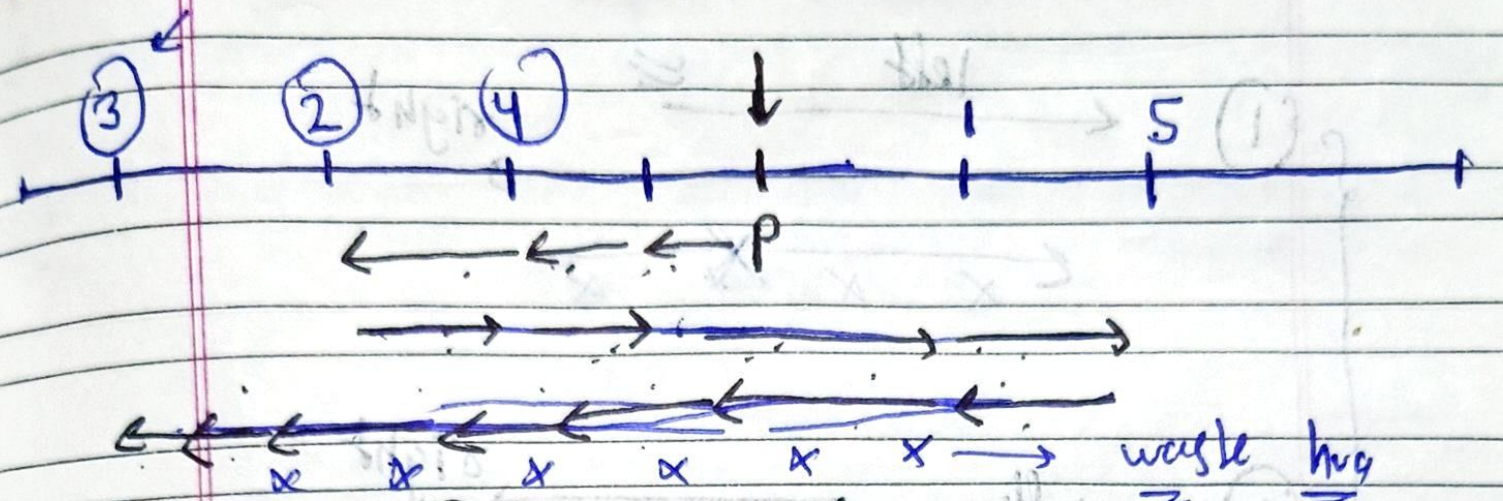
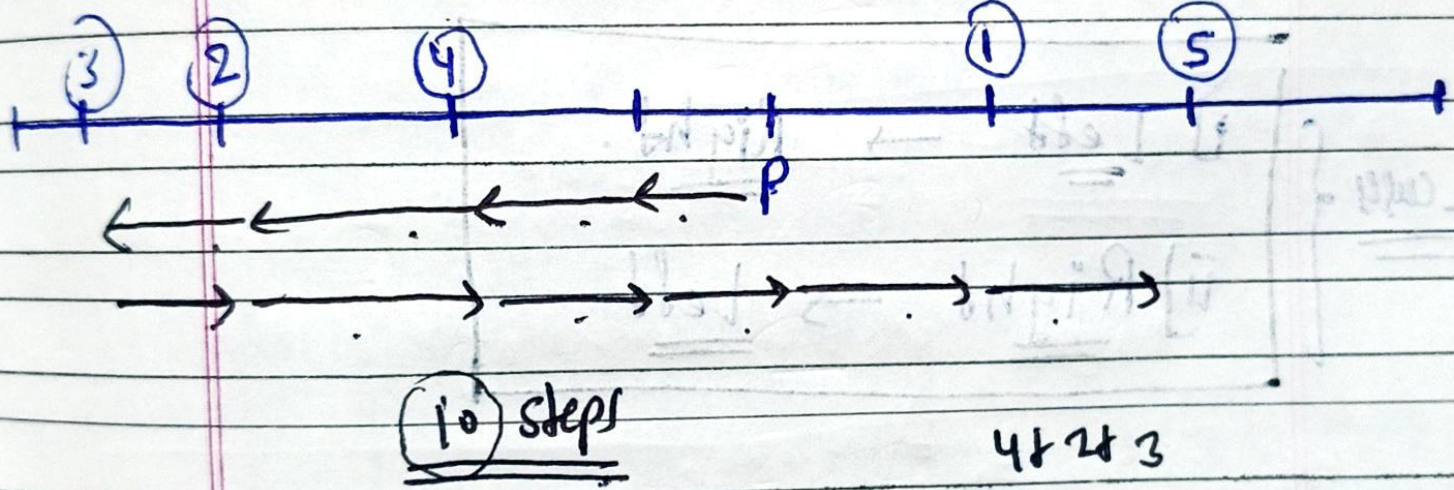
Maximum Fruits Harvested After At most K Steps



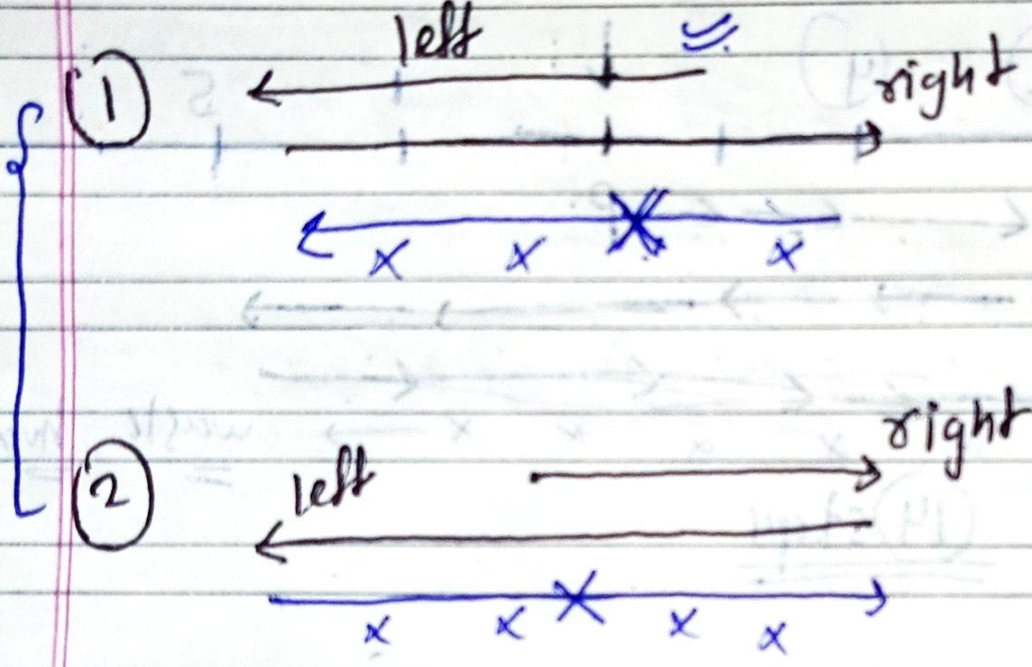
14 steps



10 steps

4+2+3

Possible Cases



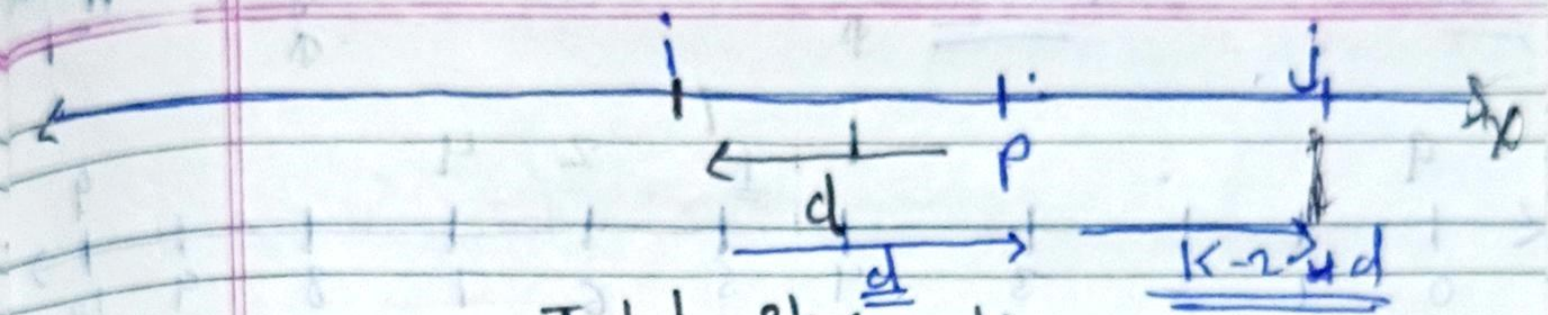
2 cases

i] Left → Right.

ii] Right → Left

Case 1: Left → Right

K



Total steps = K

left steps = d

Remain steps = (K - d)

left position (i) = (P - d)

final Rem = (K - d) - d = K - 2d

right Position = P + (K - 2*d);
(j)

leftSide = d
i = (P - d)
j = (P + K - 2*d)

(3) lower_bound(3)

(8) upper_bound(8) - 1

sorted

{0, 4, 5, 6, 7, 10}

47/3

Ans = [(0,4), (4,1), (5,7), (6,2), (7,4),

[0,4] left

10 > 8

Fruits = [(0,9), (4,1), (5,7), (6,2), (7,4), (10,4)]

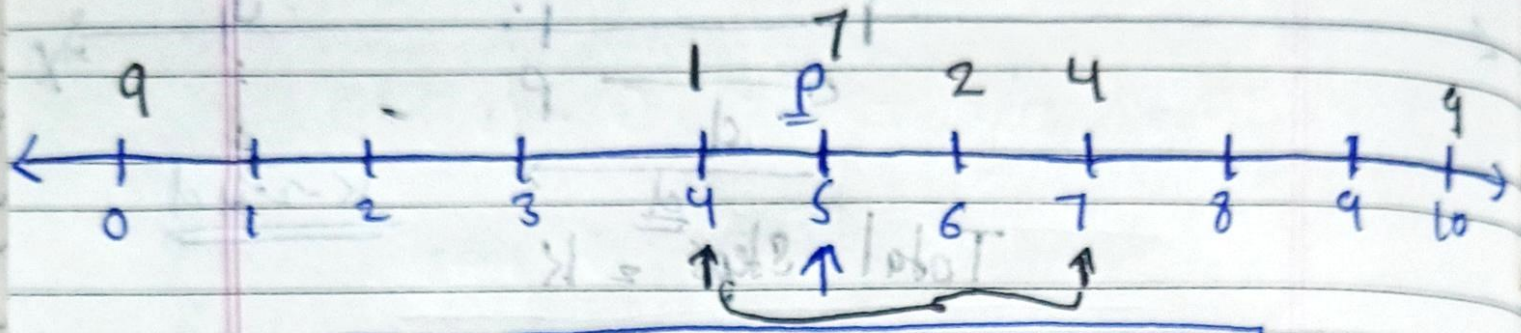
0 1 2 3 4 5

4 > 3

p = 5 (left)

right

upper



prefix sum of fruits: [9, 10, 17, 19, 23, 32]

Result: psum[Right] - psum[left-1]

---

* call 2:- Right → left

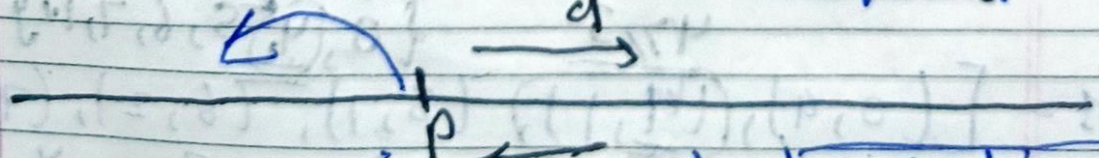
Right side = d

j = p + d

k

k - d - d

k - 2d



i = p - (k - 2d)

j = p + d's

i = p - k - 2d

* Call 2 :- R → L

j = P + d ;
i = P - (k - 2d) ;

upper_bound(j)

lower_bound(i)

Left & Right

Result | PSum[Right] - PSum[Left - 1] ;

⟵ d
d ⟶

Call - 1 (Left)

i = P - d

j = P + (k - 2d)

Call - 2 (Right)

i = P - (k - 2d)

j = P + d

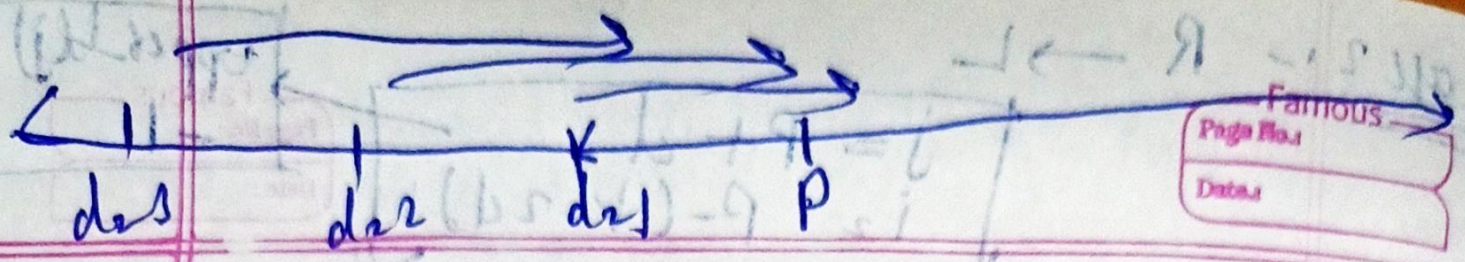
k steps — Left → d.

(k - d) Remain.

k - 2d >= 0

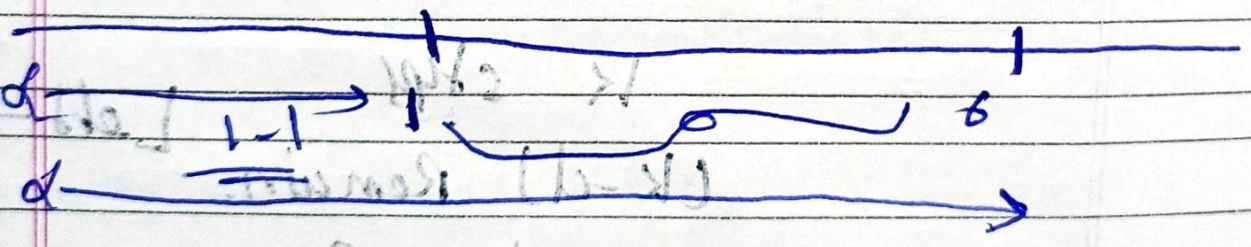
2d <= k

d <= k/2




```
for( d = 0; d <= k/2; d++){
    // case 1 : left
        i = P - d;
        j = P + k - 2d;

    left = lower_bound(i); //Position
    right = upper_bound(j) - 1; //Position
    : Result = PSum[right] - PSum[left-1];
    mxfruits = max(mxfruits, Result);
```




```
// case 2 Right
    i = P - (k - 2d);
    j = P + d;
```

low ct_b
upp —
pt —
Rec —
marks. —
}