arr = [7, 1, 5, 3, 6, 2]

0 1 2 3 4 5

day 0 ③ day 1 day 2 day 3 day 4 day 5

Profit = S.P − B.P

sell buy

6 − 1 = 5

arr = [100, 1, 10, 3]

max = 100

min = 3

97

① To get maximum Profit, try to sell at higher value (after buying) and buy at lower value.

So iterate through the array from left to right and use 2 variables, first to store buying price and other to store Profit.

```
int buy = arr[0];
int profit = 0;
for(int i = 1; i < n; i++)
{ if( arr[i] < buy )
      buy = arr[i];
  else if( arr[i] - buy > profit)
      profit = arr[i] - buy;
}
```

- Whenever problems asks you to find maximum value in an array, use `int maxi = INT_MIN` for initialization.

buy = 1 sell = 3

① Sum += n means

Sum = Sum + n

① subarray is a contiguous array elements present in an array. [1, n]

eg:- [20, 30, 40, 50, 15, 25, 85]

(20, 30, 50) (20, 50, 85)

○ HashMap :- It is a data structure which is used to store key-value pair

map["IPL"] = 5;

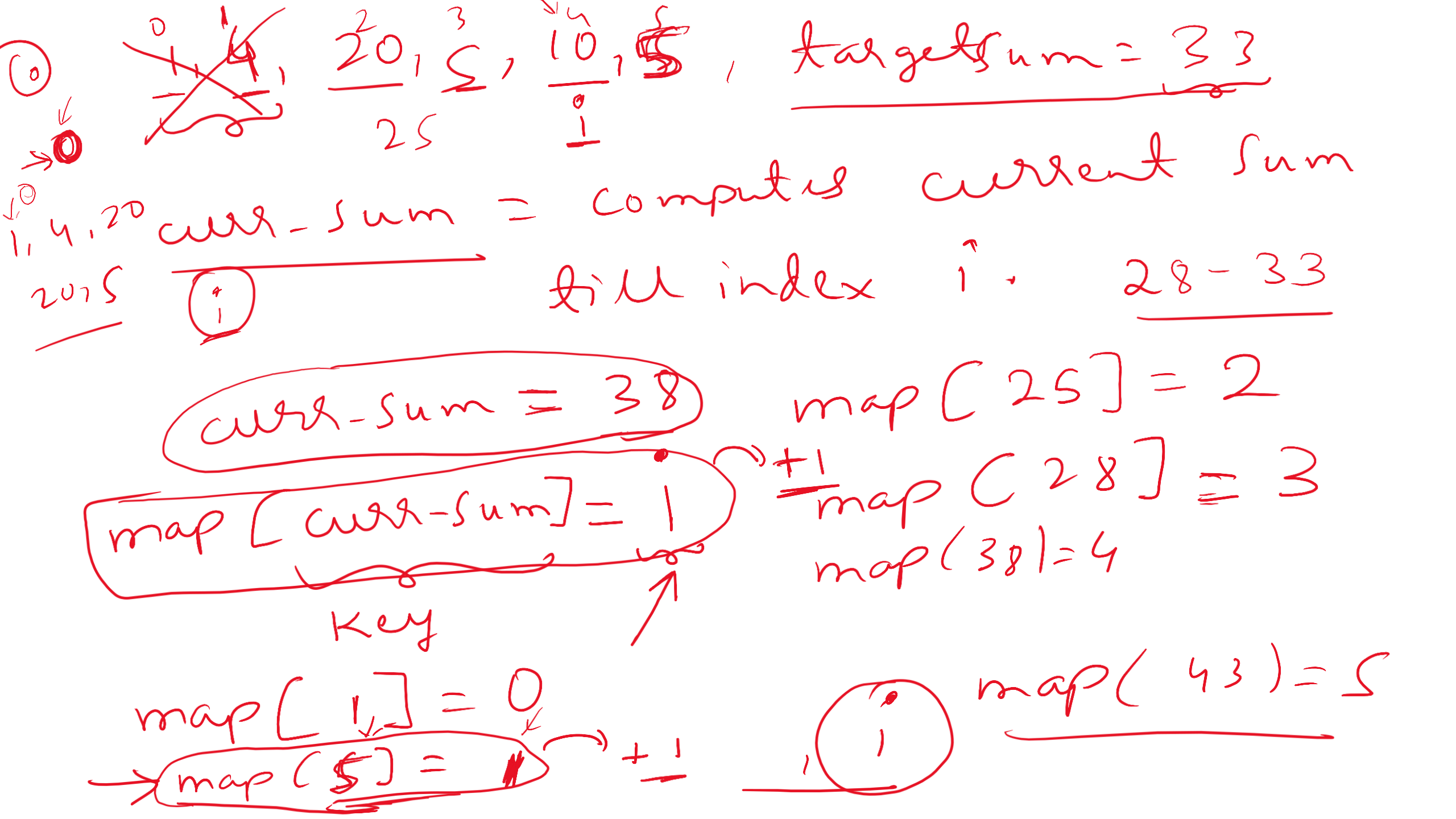
"IPL" → Key

5 → value

map[50] = 10;

50 → Integer Key

10 → value


0 1 2 3 4
1, 4, 20, 5, 10, 5, targetSum = 33
25
1
1, 4, 20
2015
curr-sum = computes current sum till index i. 28-33
curr-sum = 38
map[curr-sum] = 1
key
+1
map[25] = 2
map[28] = 3
map(38) = 4
map[1] = 0
map[5] = 1
+1
map(43) = 5

① 5, 3, 10, 15, 5 ; target sum = 30

0 1 2 3 4

end = 4
start = 2

size of subarray = 3

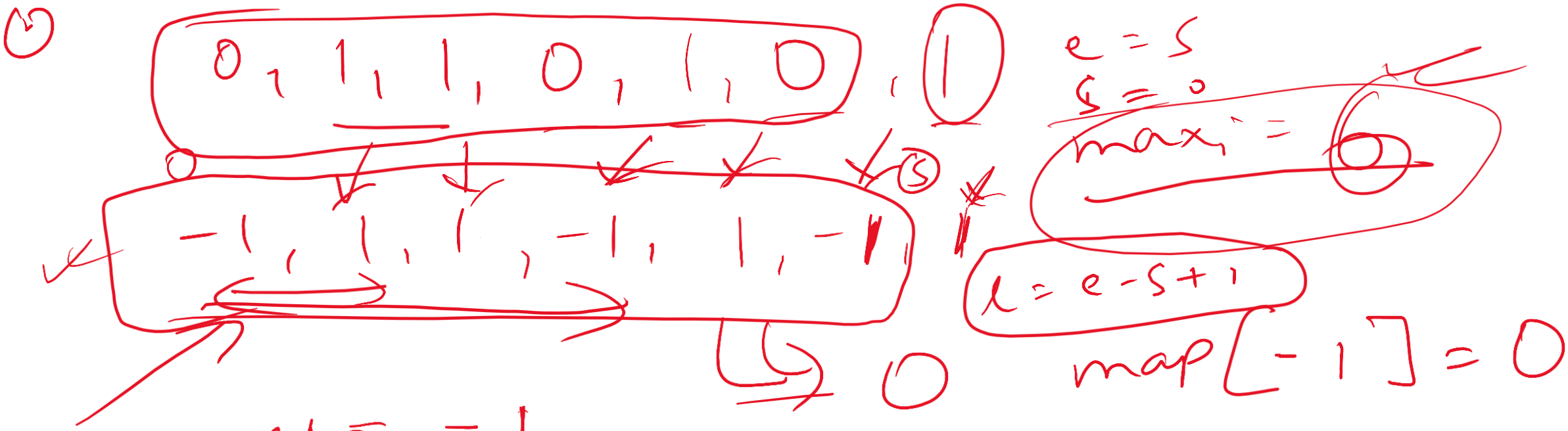
map(curr - t.s] = index

i → last element

start = index + 1

end = i

end - s.start + 1 == 3


0, 1, 1, 0, 1, 0, 1
-1, 1, 1, -1, 1, -1
5
0
e = 5
s = 0
maxi = 6
l = e - s + 1
map[-1] = 0
curr = -1
curr = 0
curr = 1
curr = 0
curr = 1
curr = 1
curr = 0
map[1] = 6

arr = [18, 5, 18, 19, 18, 13, 20, 13, 13]

h = 9