1 transaction

buy → sell

Stock

buy

sell

[6 7 1 9 4 1 6 18 4]

buy sell

1 buy

0

profit = sell - buy

18 - 7 = 11

18 - 1 = 17 ∈

i

sell

0

1

2

3

...

buy/sell

Line 2

sell

sell

	6	7	19	4	1	6	18	4
buy ☆	6	6	6	4	1	1	1	1
profit	0	1	13	0	0	5	17	3

17

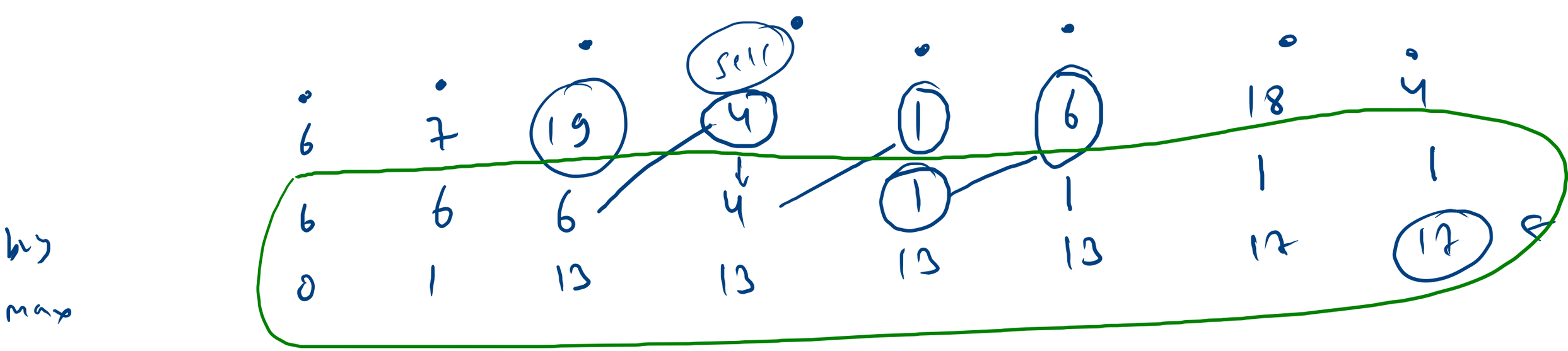
buy = arr[0], ∞

0	1	13	13	13	13

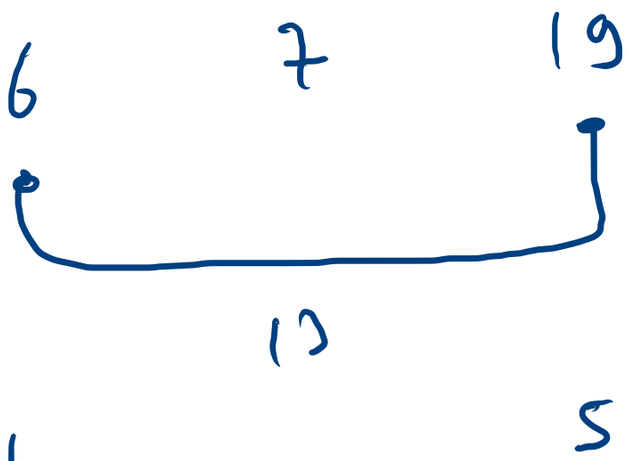
max = 0

val in arr

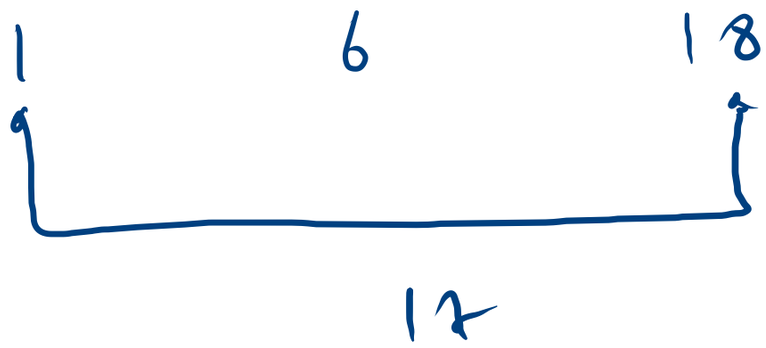
buy = min(val, buy)




```
    int buy = cost[0];
    int max = 0;

    for(int val : cost){ 1  4
        buy = Math.min(buy, val);
        max = Math.max(max, val-buy);
    }                    17, 4-1
                              3
    System.out.println(max);

}
```



4



4

6

5

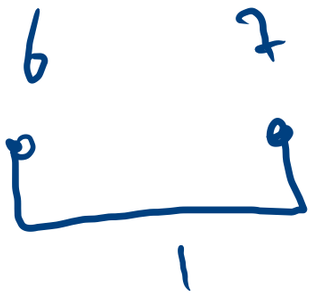
6

6

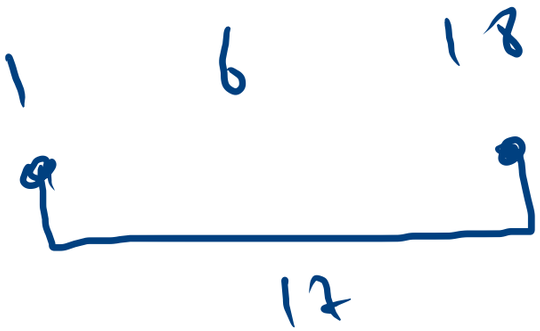
5

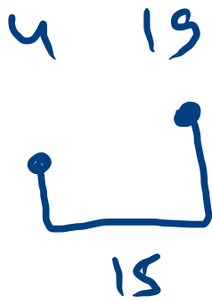
5

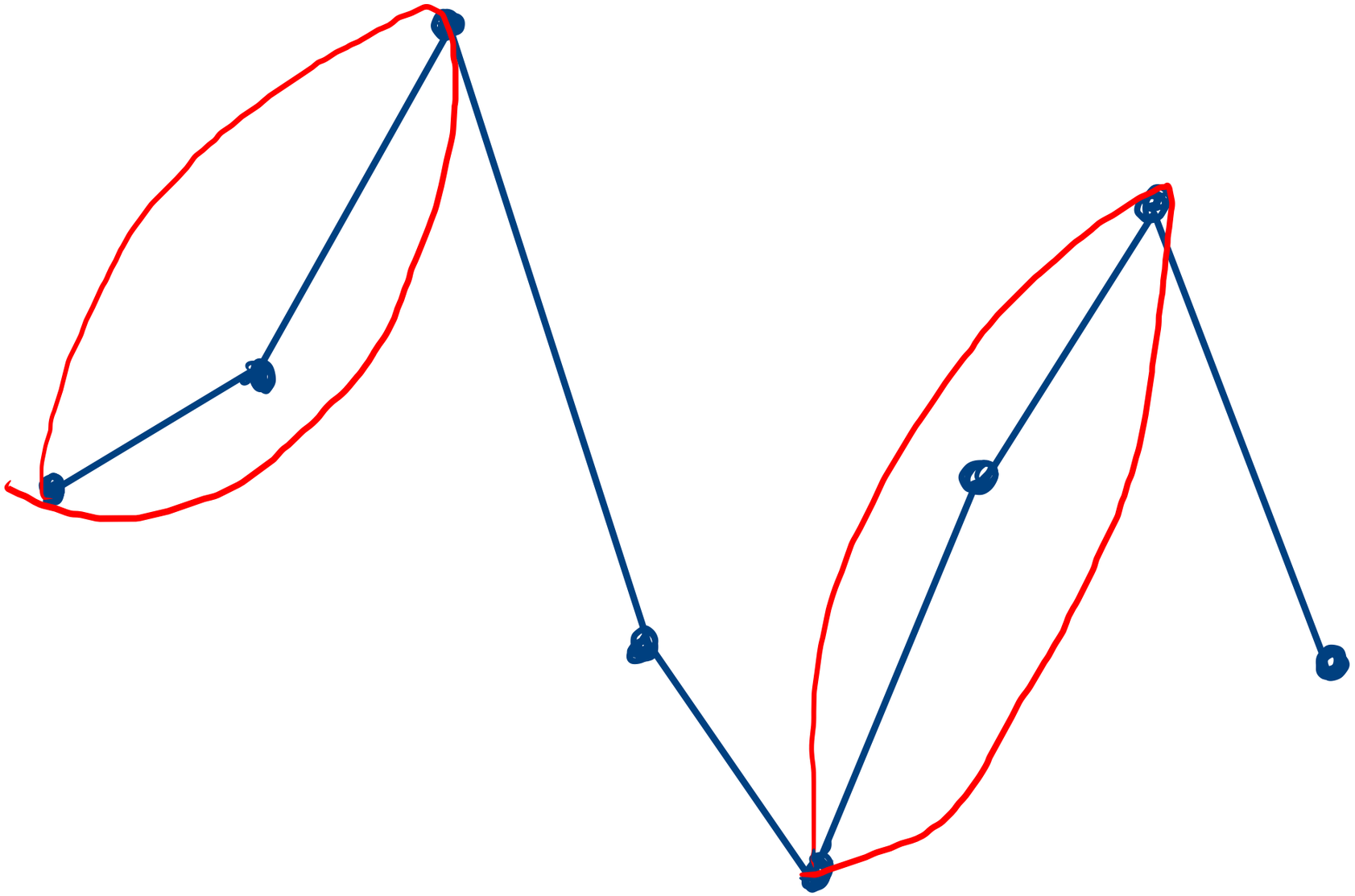
30

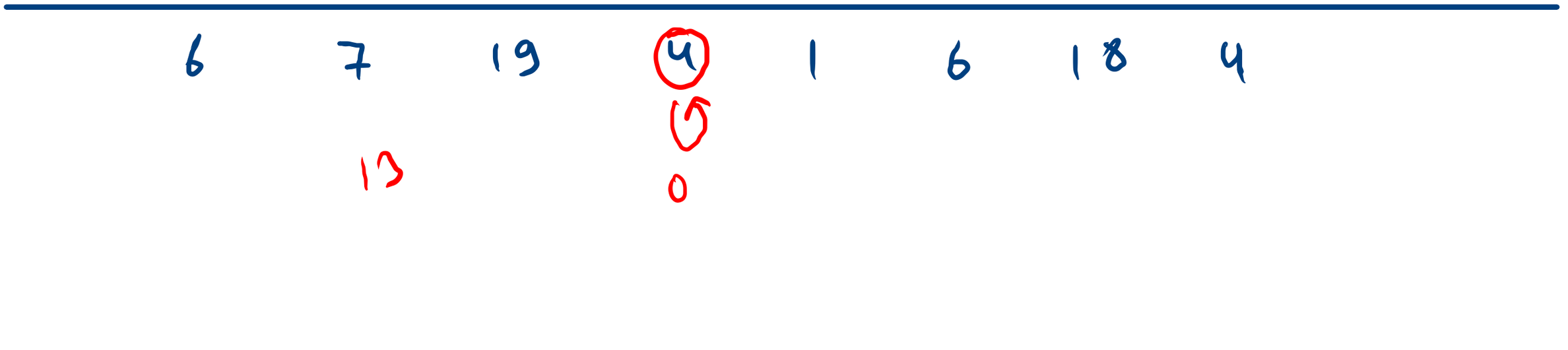


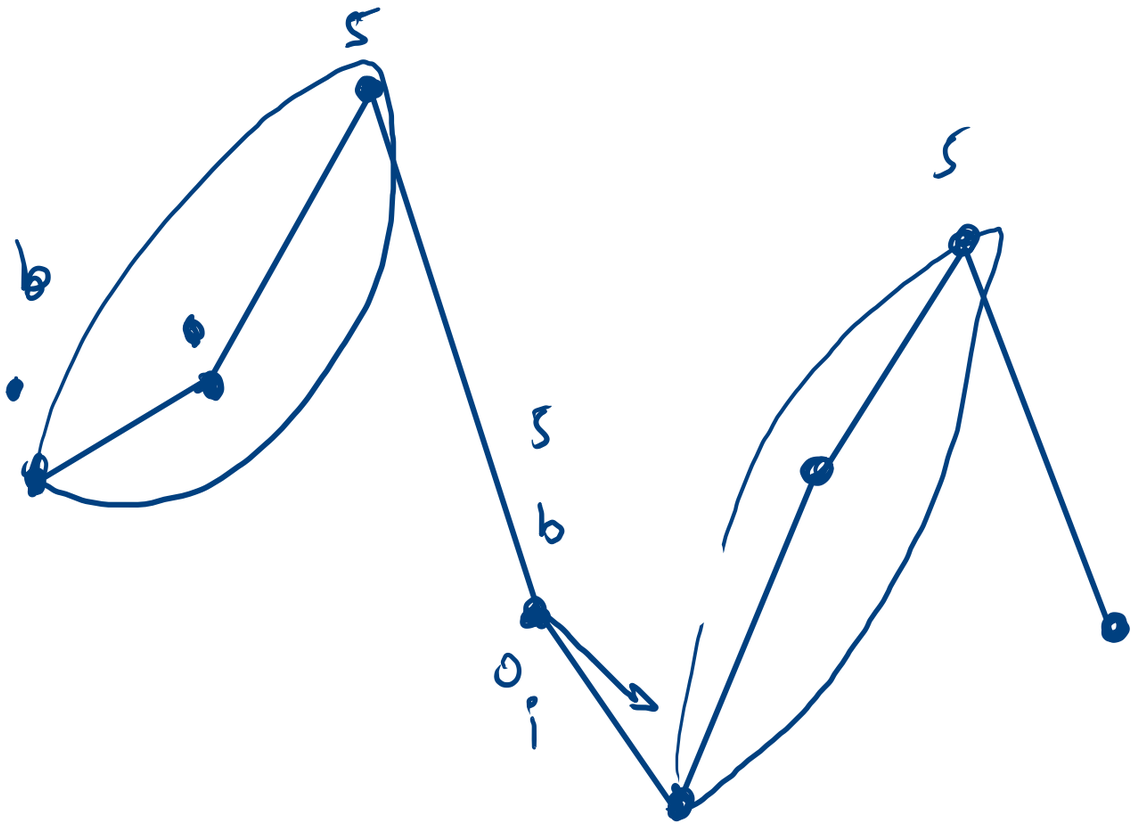
4

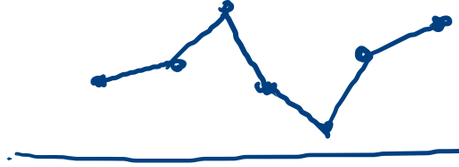


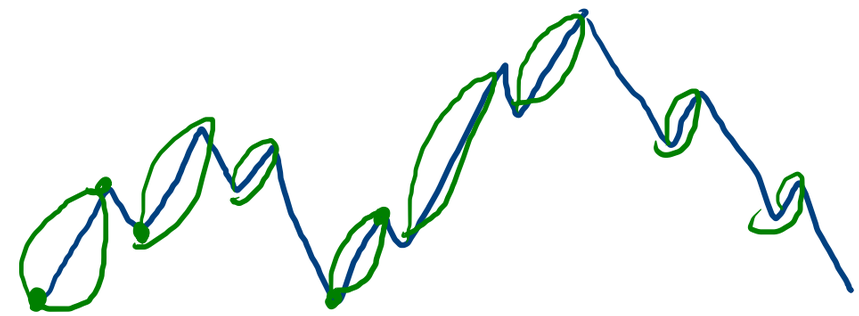





6
7
19
4
1
6
18
4
5
13
0







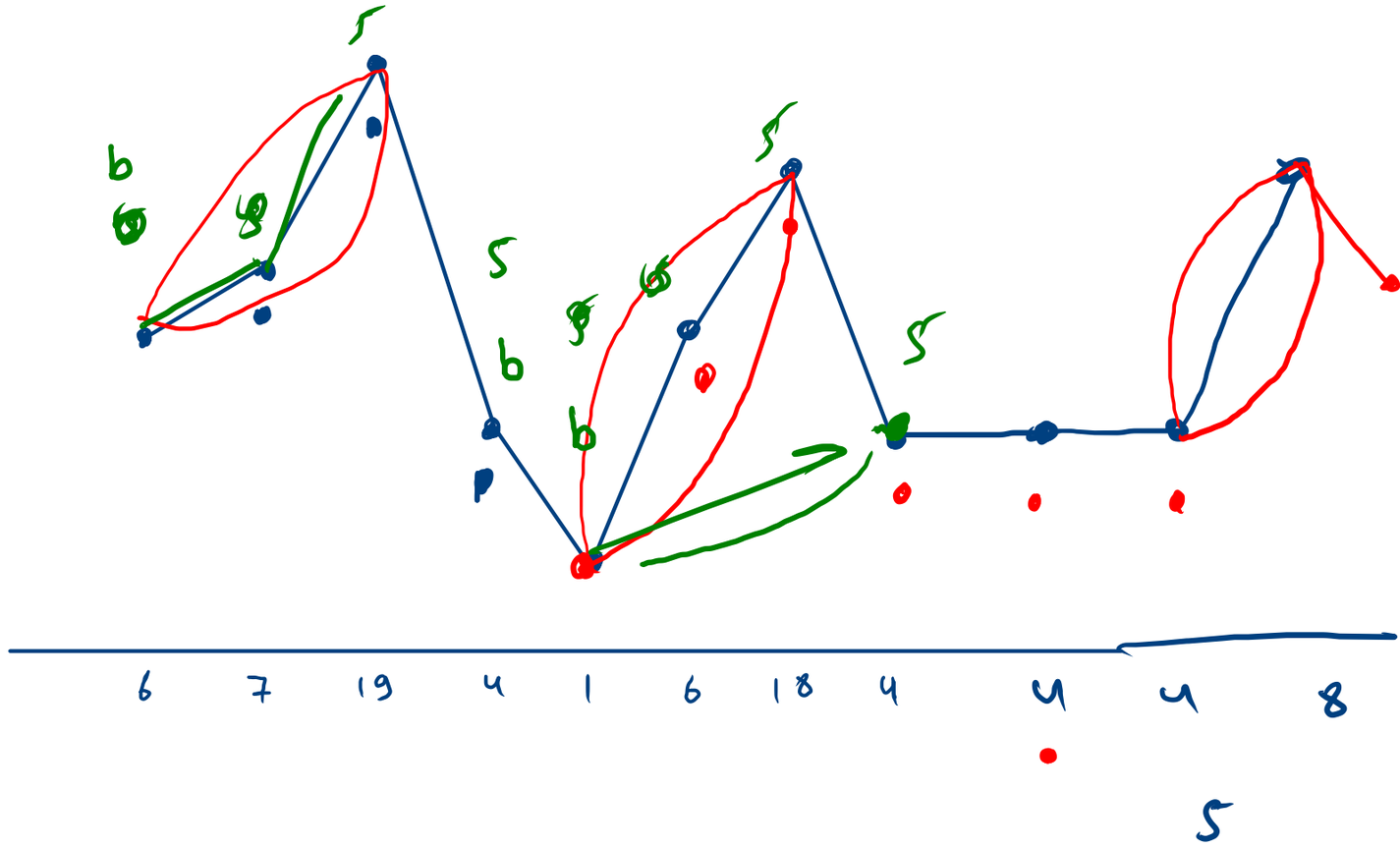
6 7 19 4 1 6 18 4

★ increm. s move

★ decr trans comple

→ b, s, move

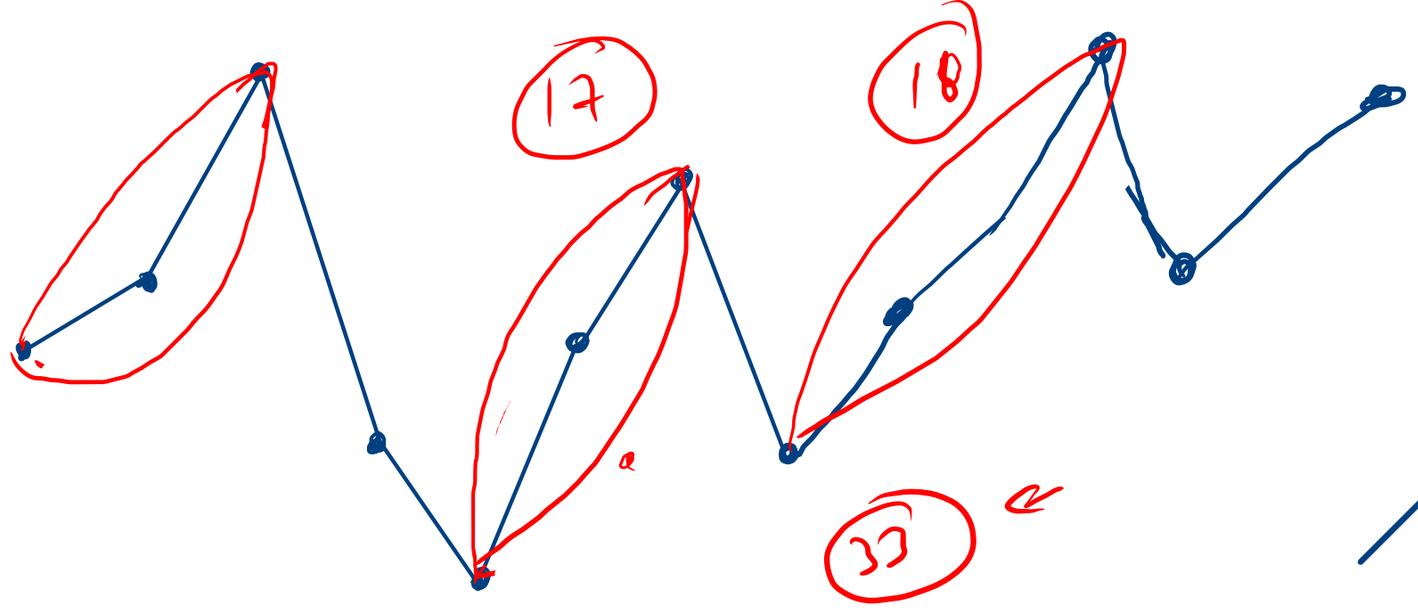
```java
int buy = cost[0];
int sell = cost[0];
int profit = 0;

for(int i=1;i<n;i++){
    if(cost[i-1] < cost[i]){// increasing
        sell = cost[i];
    }else{
        profit += sell-buy;
        sell = buy = cost[i];
    }
}
System.out.println(profit);
```

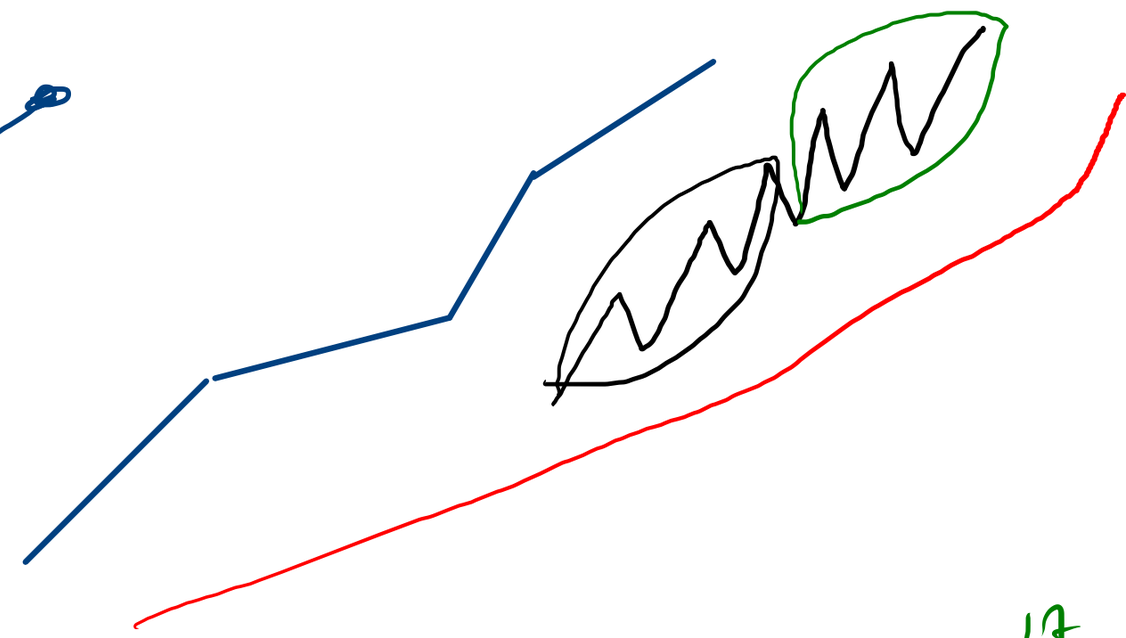


profit = 0 + 13 + 0 + 17 + 0 + 0

4-4 = 0

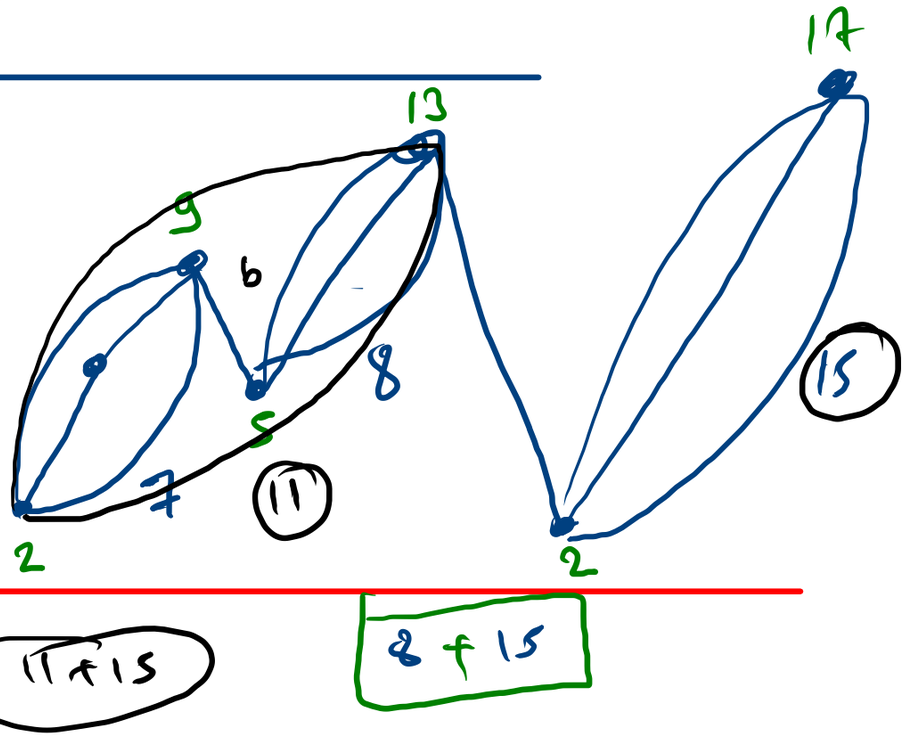
5

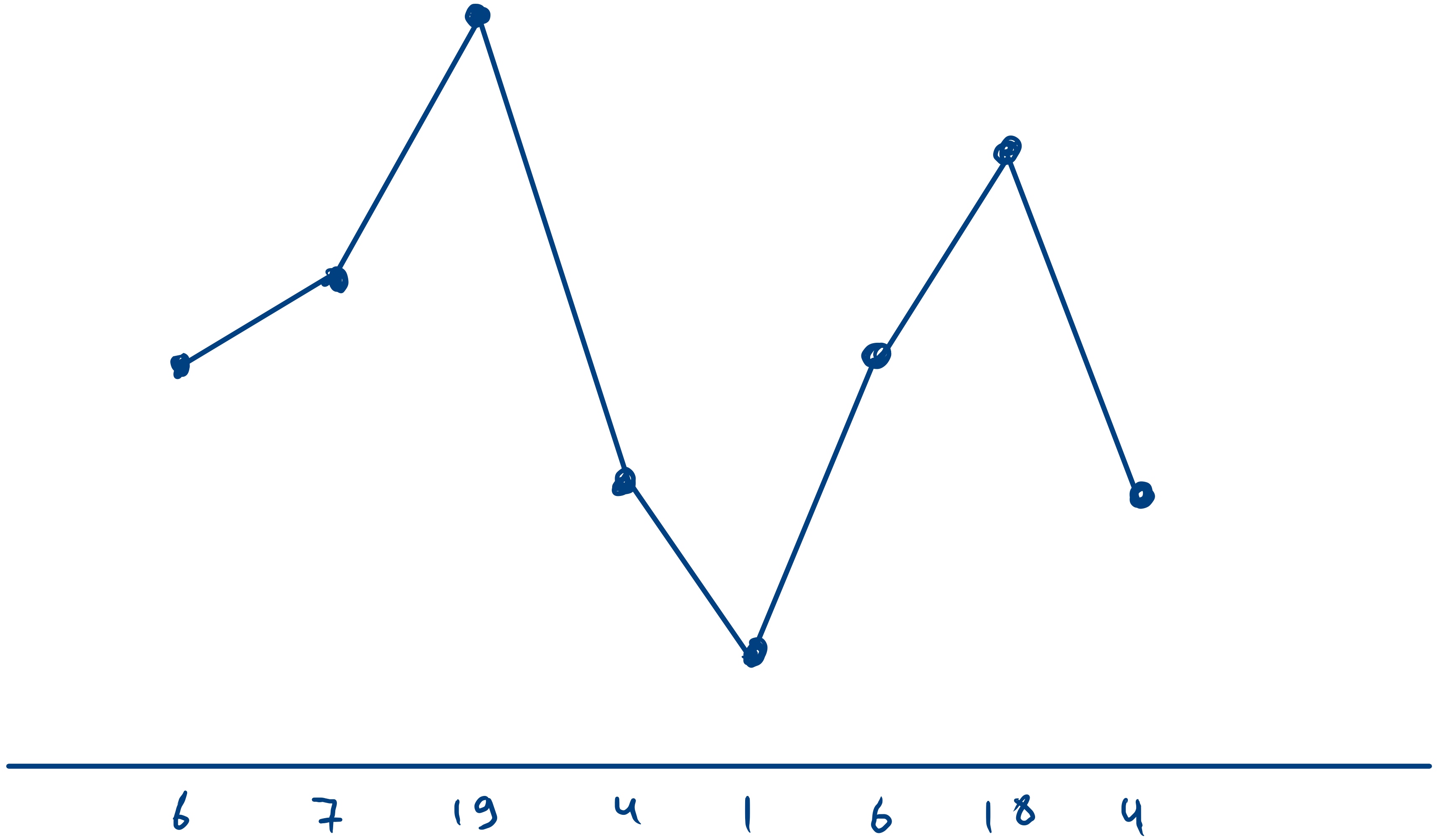




13	17

2 trans allow





min 6

profit 0

→ p1 = 0

stock b1

1 stock

buy → p1 - b1

-b1

s1

p1 - b1 + s1

sell → p1 - b1 + s1

0 stock

Let p2 = s1 - b1

one transaction

→ p2

stock b2

1 stock

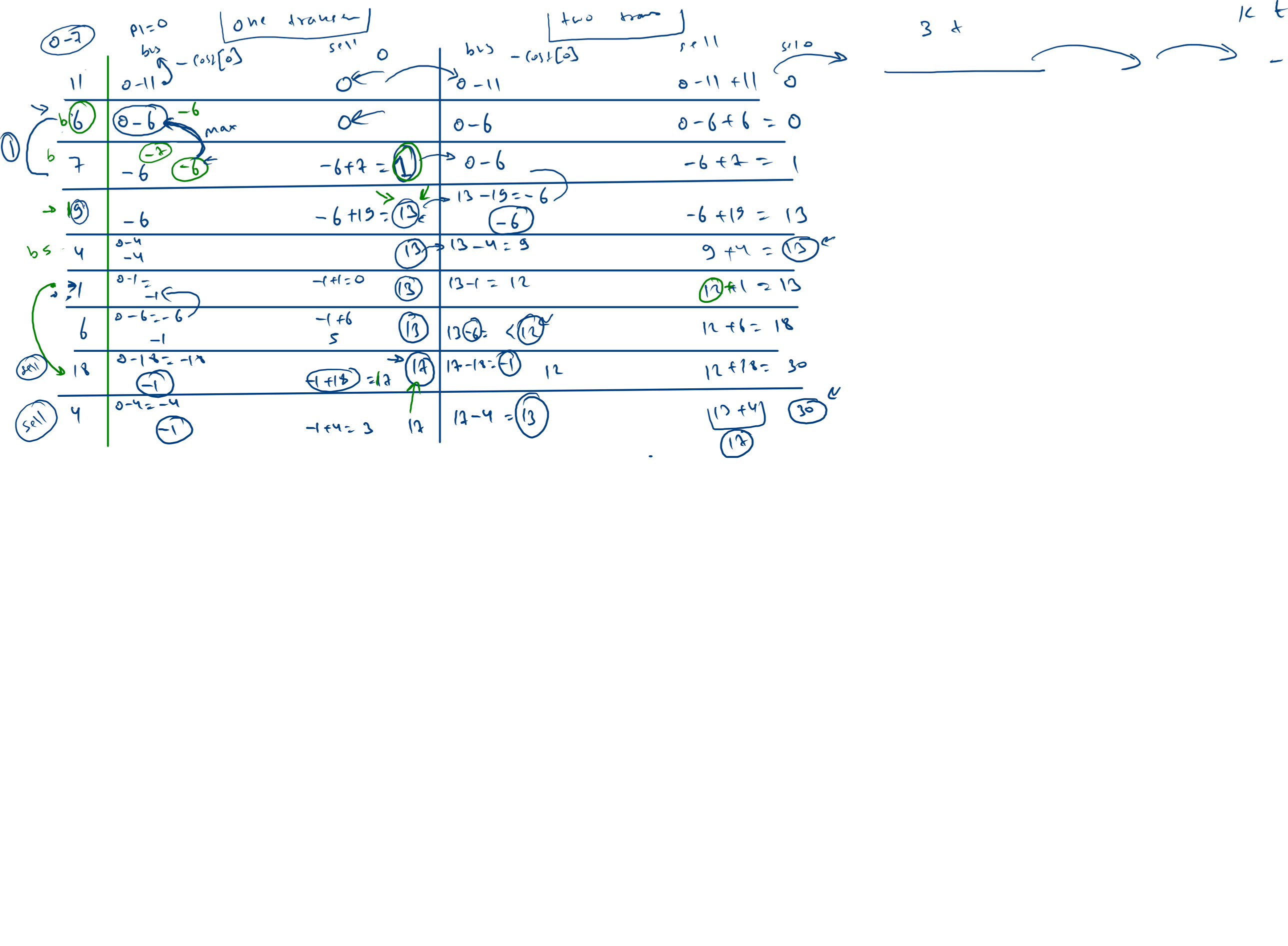
buy → p2 - b2

stock → s2

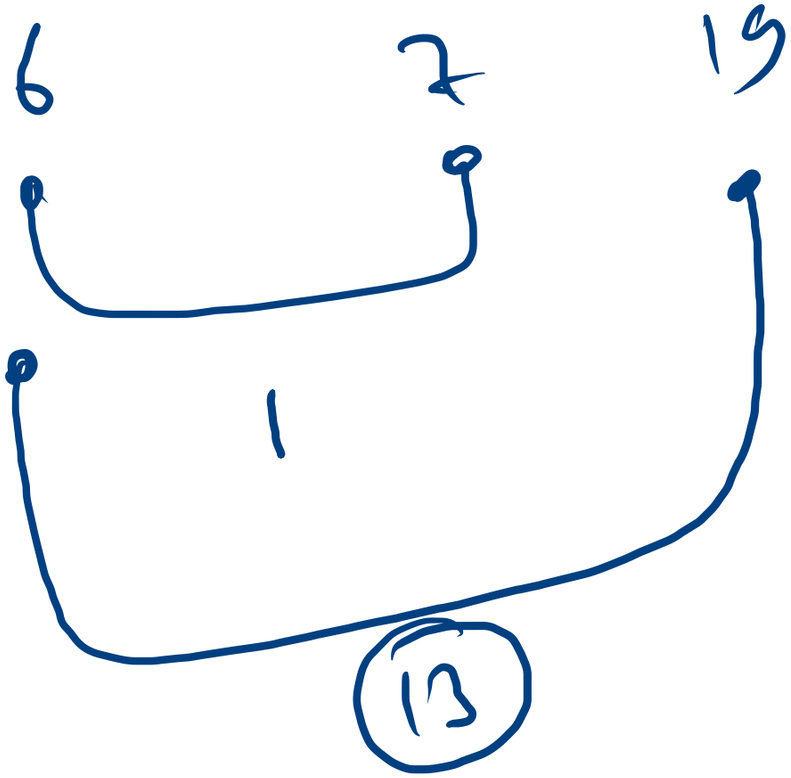
sell

p2 - b2 + s2

two trans.

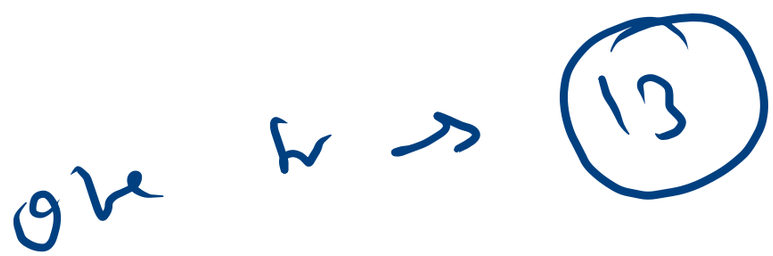

one transaction
two transactions
buy
sell
max
3 t
k t

11



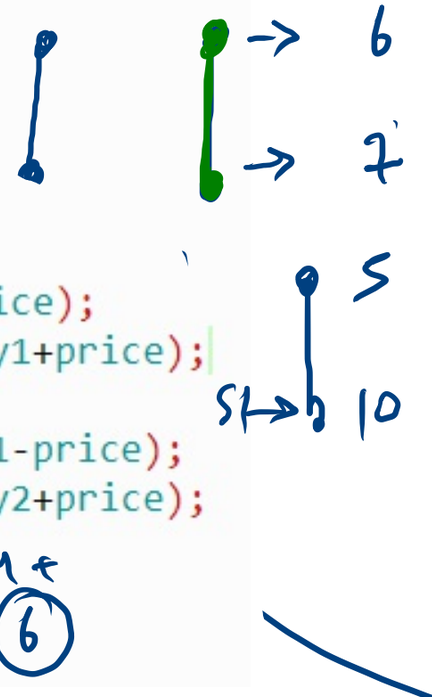
4

18



2 trans

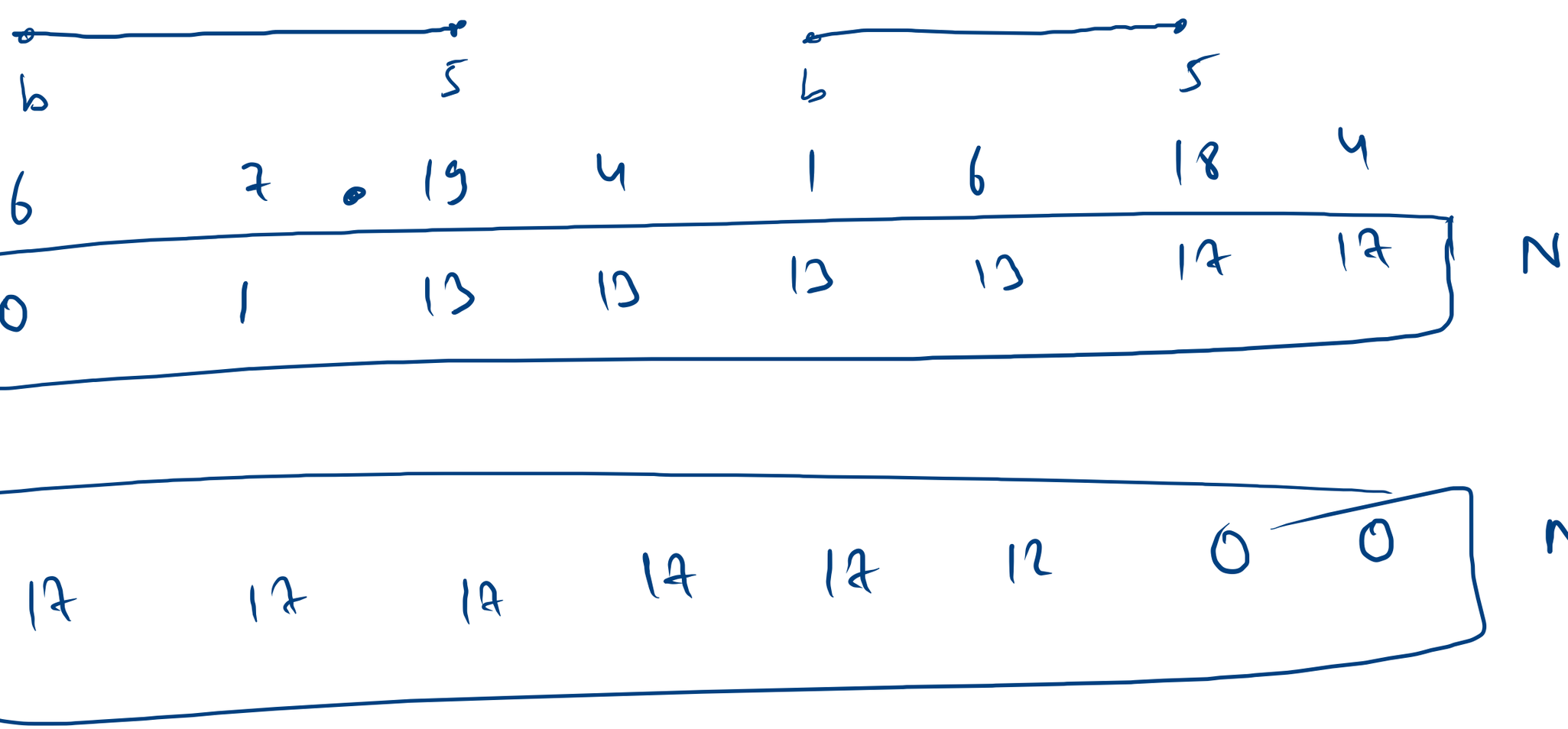
```
int buy1 = -cost[0];
int buy2 = -cost[0];
int sell1=0, sell2=0;

for(int i=1;i<n;i++){
    int price = cost[i];

    buy1 = Math.max(buy1, 0-price);
    sell1 = Math.max(sell1, buy1+price);

    buy2 = Math.max(buy2, sell1-price);
    sell2 = Math.max(sell2, buy2+price);
}

System.out.println(sell2);
```



buy1	sell1	buy2	sell2
-11	0	-11	0
-6	0	-6	0
-6	1	-6	1
-5	1	-4	1
-5	5	-4	6

Max sum non adjacent

If 10 − 5 = 5



time $2N + N$ $O(N)$

space $2N$ $O(N)$

25-16-3 = 6

10 15 17 20 16 18 22 20 22 20 23 25

20-10-3

22-16-3

22-20-3

-1

25-20-3

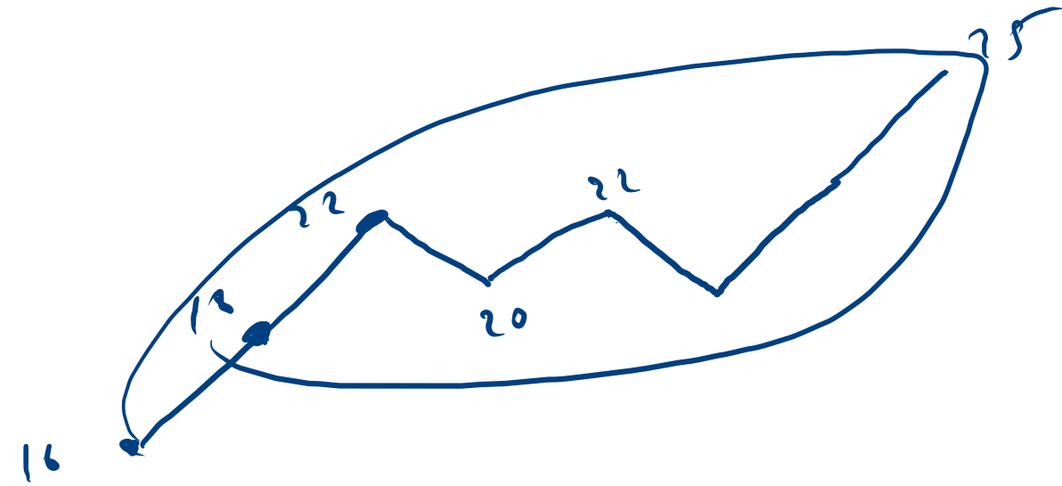
7

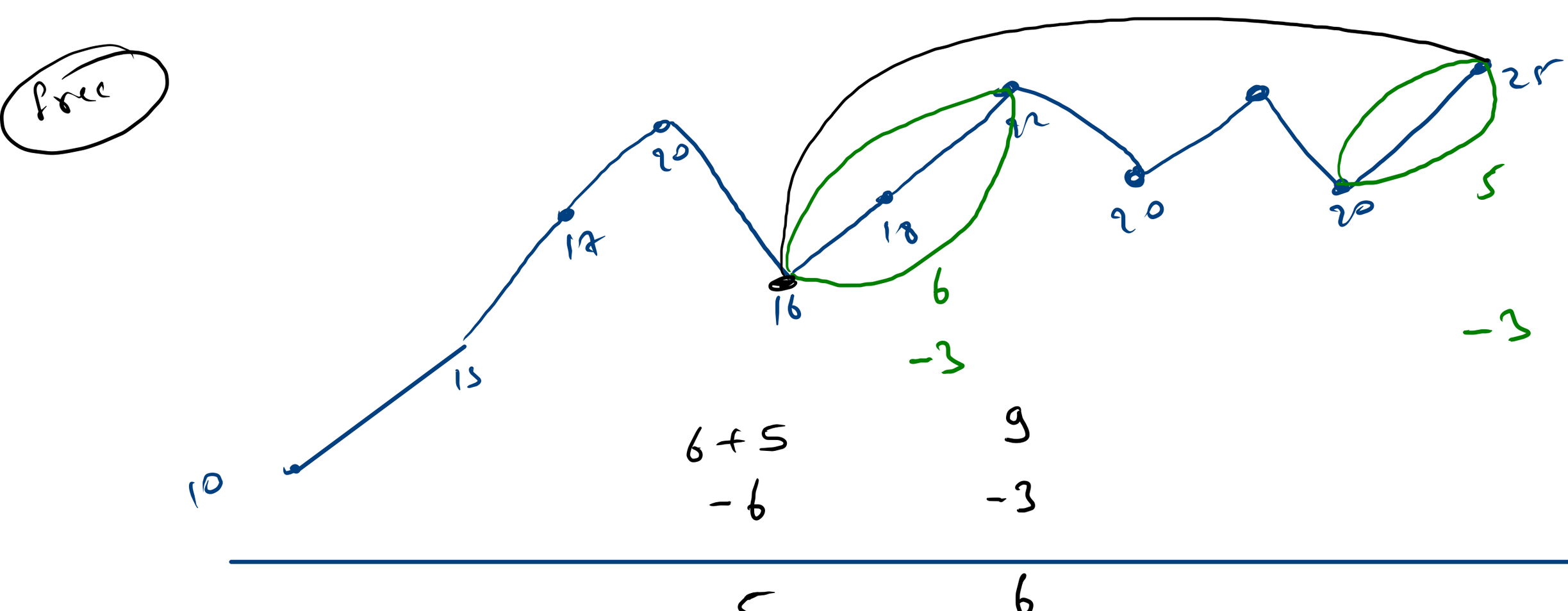
3

2

el -> 3

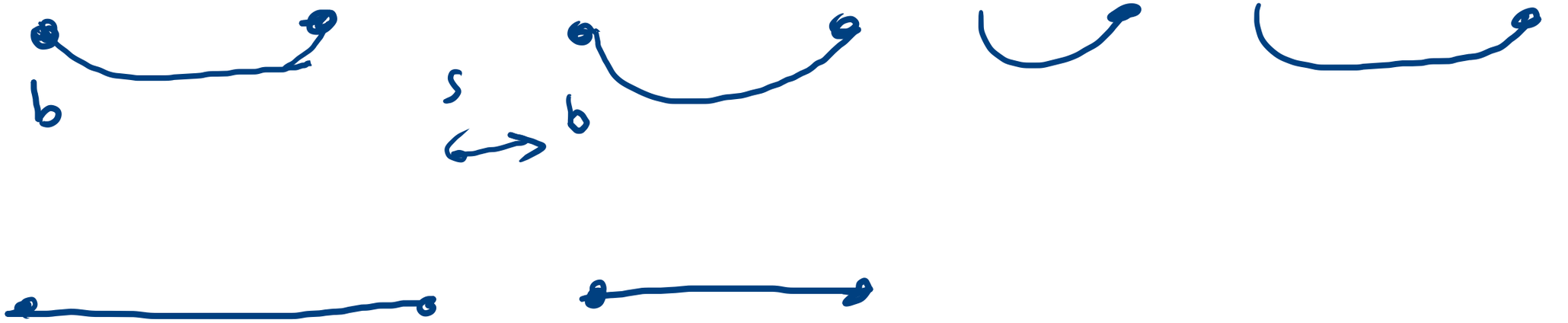
12



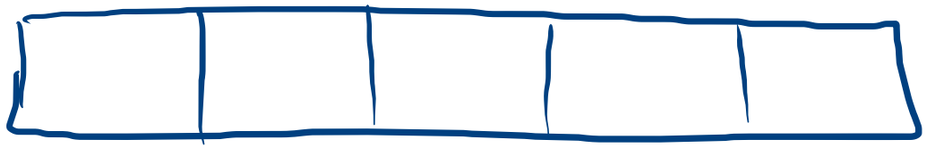


10 15 17 20 16 ~~18 22 20~~ 22 20 23 25

10 15 17 20 16 18 22 20 22 20 23 25



ar



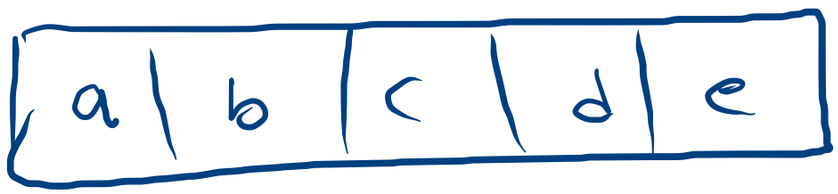
fixed size

stack

nL

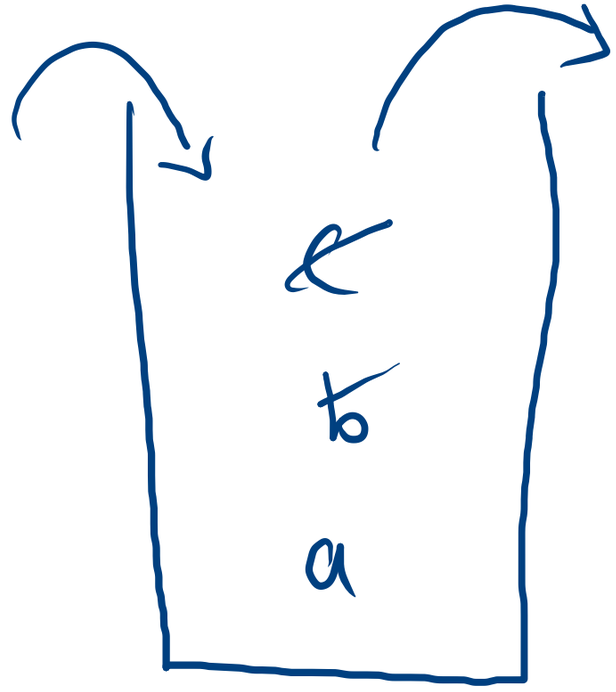
dynamic

stack



discipline

LIFO



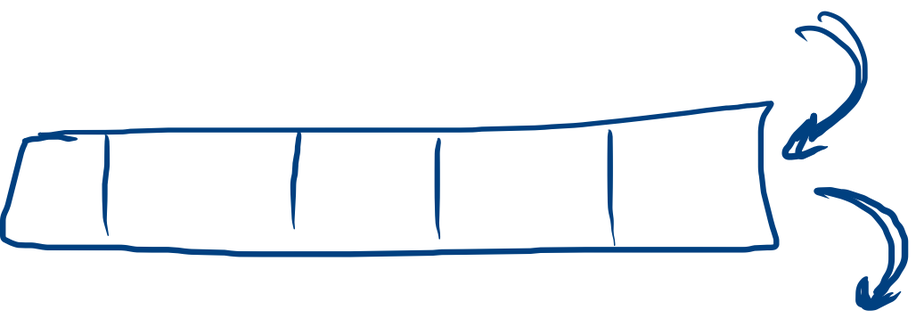
DL

add

remove(i)

get

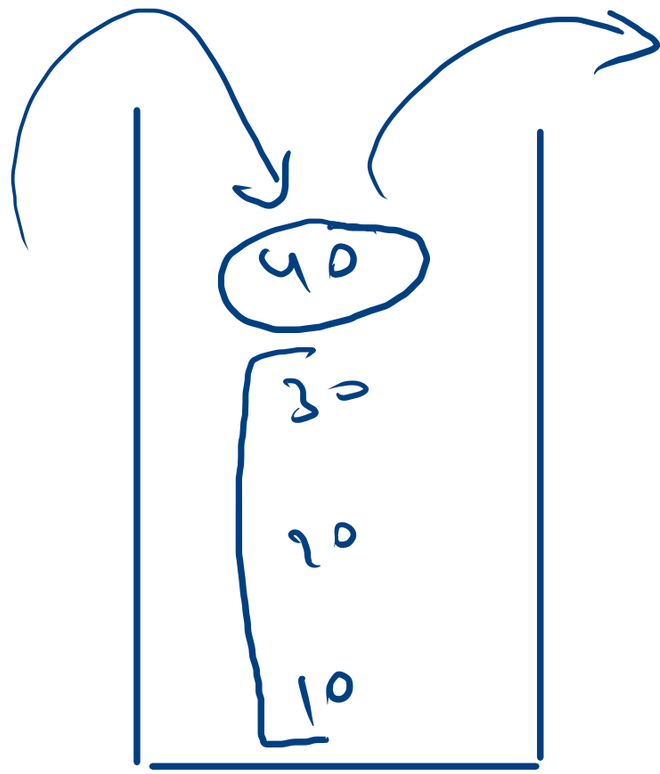
size

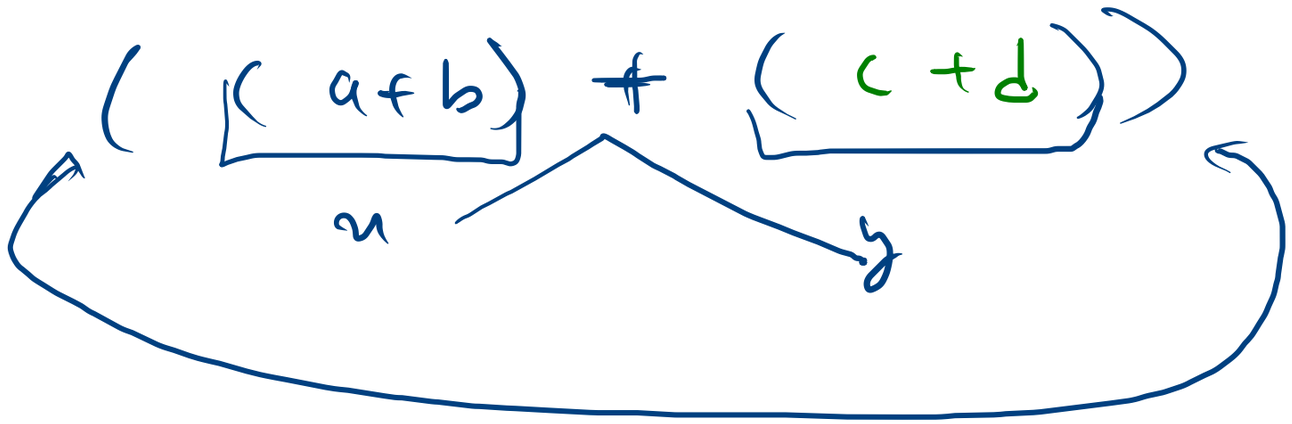


get(i)

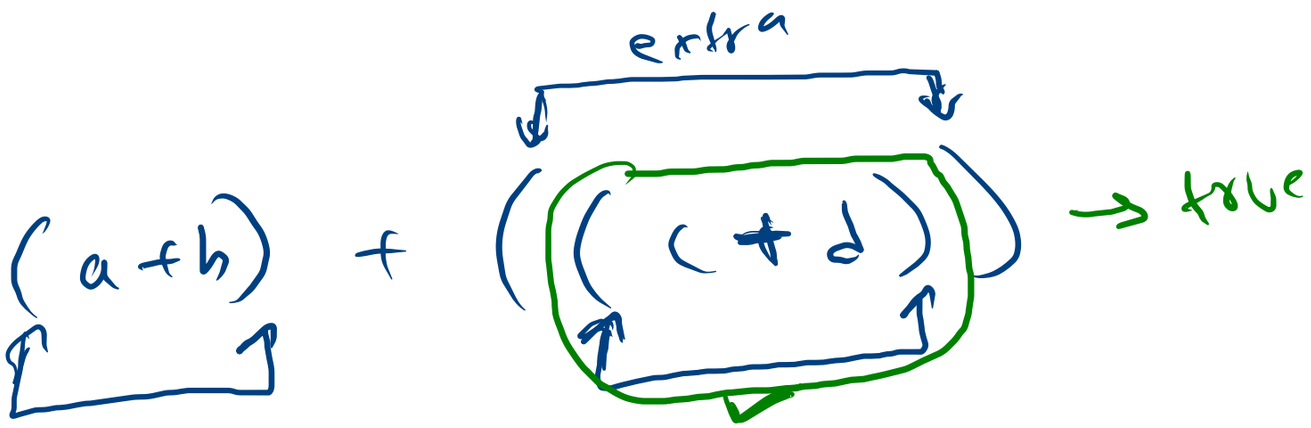
stack

push	O(1)
pop()	O(1)
peek	O(1)
size	O(1)

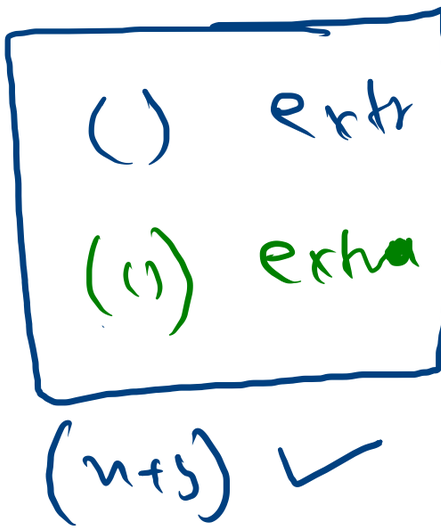


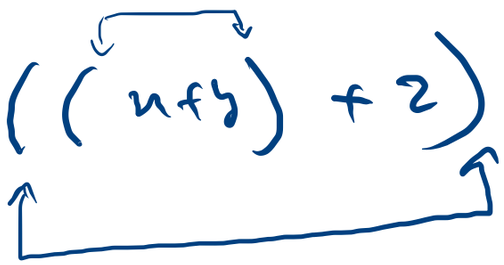


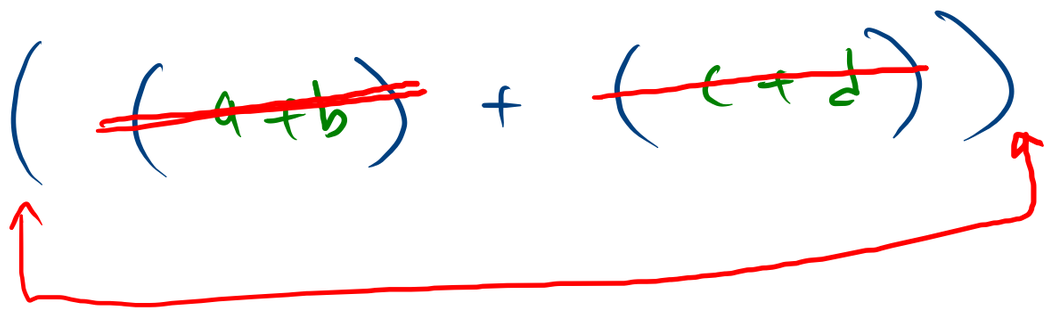
→ false

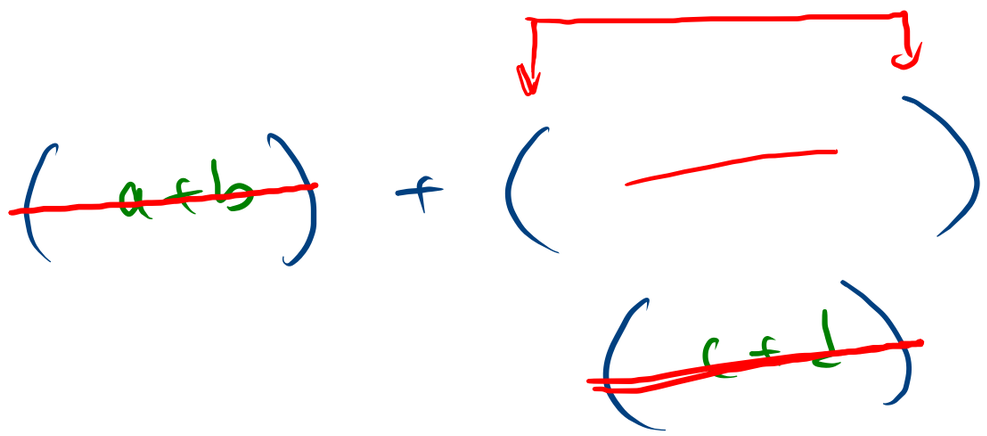


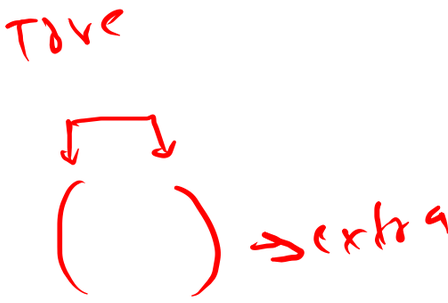
→ true









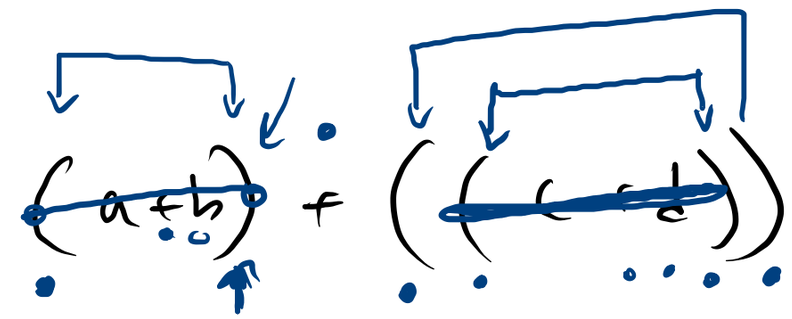


e.g.

((a + b) + (c + d)) -> false

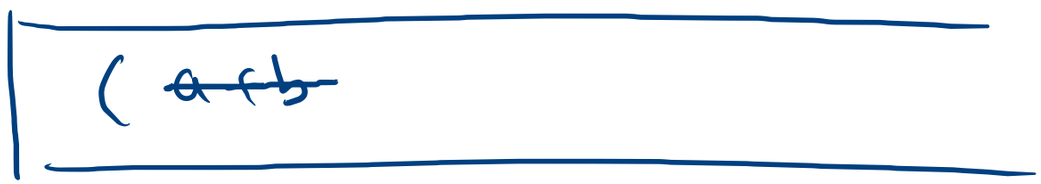
(a + b) + ((c + d)) -> true

str →

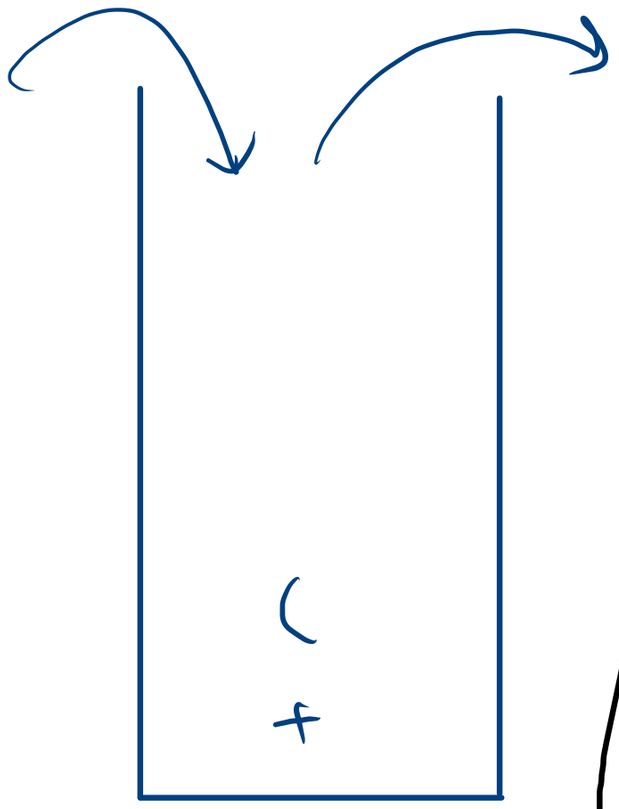


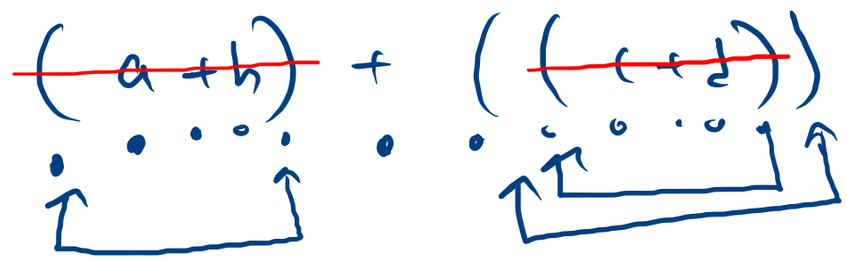
→ true

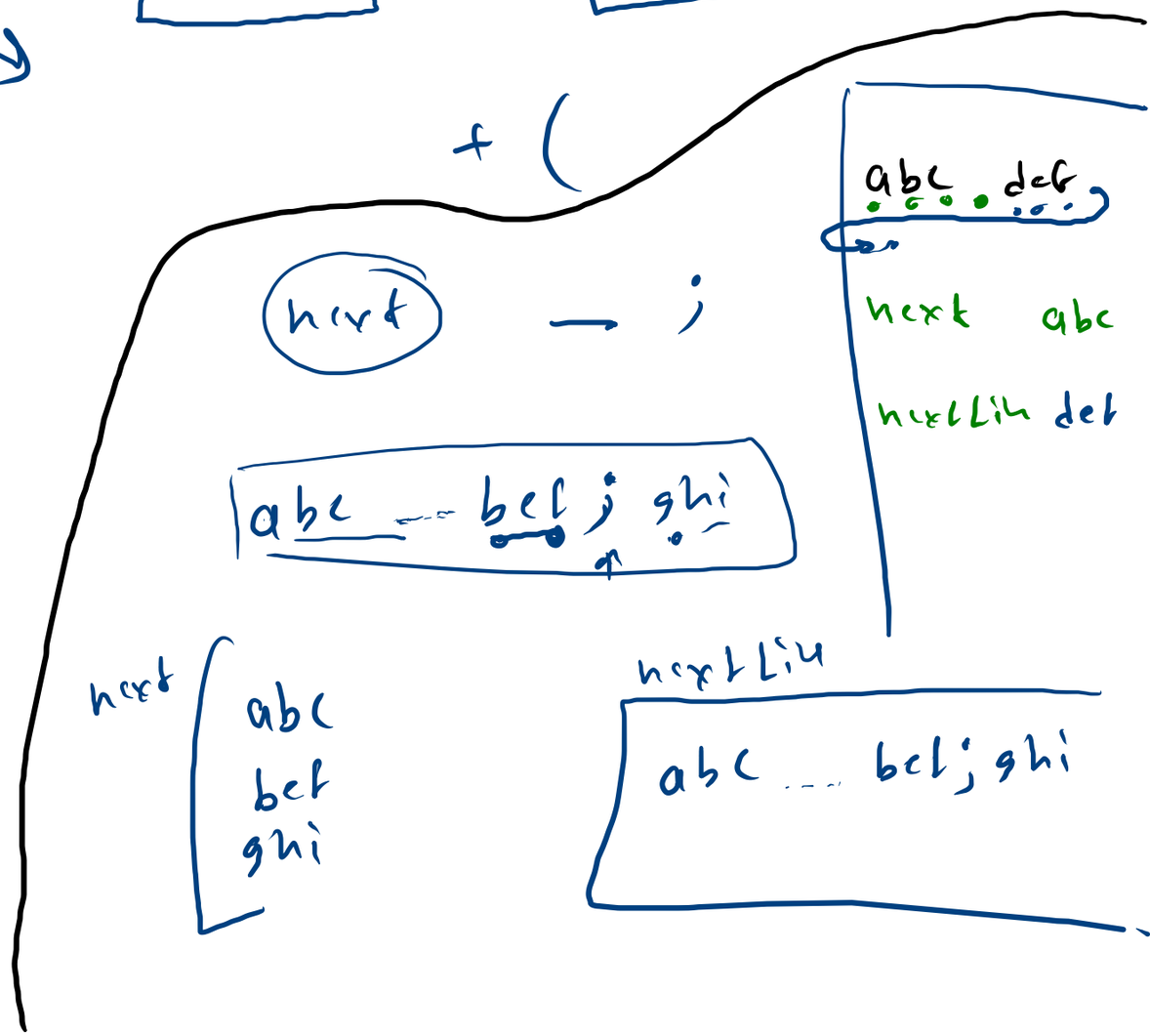
rest
algo
removal



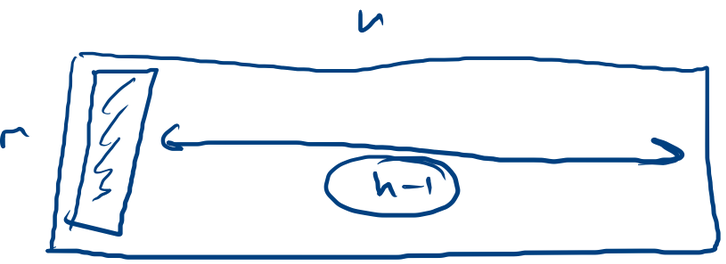
```java
Scanner scn = new Scanner(System.in);

String str = scn.nextLine();

Stack<Character> st = new Stack<>();

boolean hasextra = false;
for(int i=0;i<str.length();i++){
    char ch = str.charAt(i);

    if(ch == ')'){
        if(st.peek() == '('){
            hasextra = true;
            break;
        }else{
            while(st.peek() != '('){
                st.pop();
            }
            st.pop();
        }
    }else{
        st.push(ch);
    }
}

System.out.println(hasextra);
```



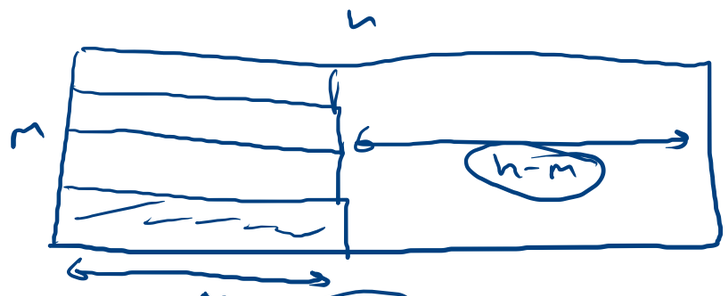




$f(n-1)$



$f(n-m)$



m=3