9

11 6 7 19 4 1 6 18 4

Max Profit ?

↳ only 1 transaction

↓

B-S

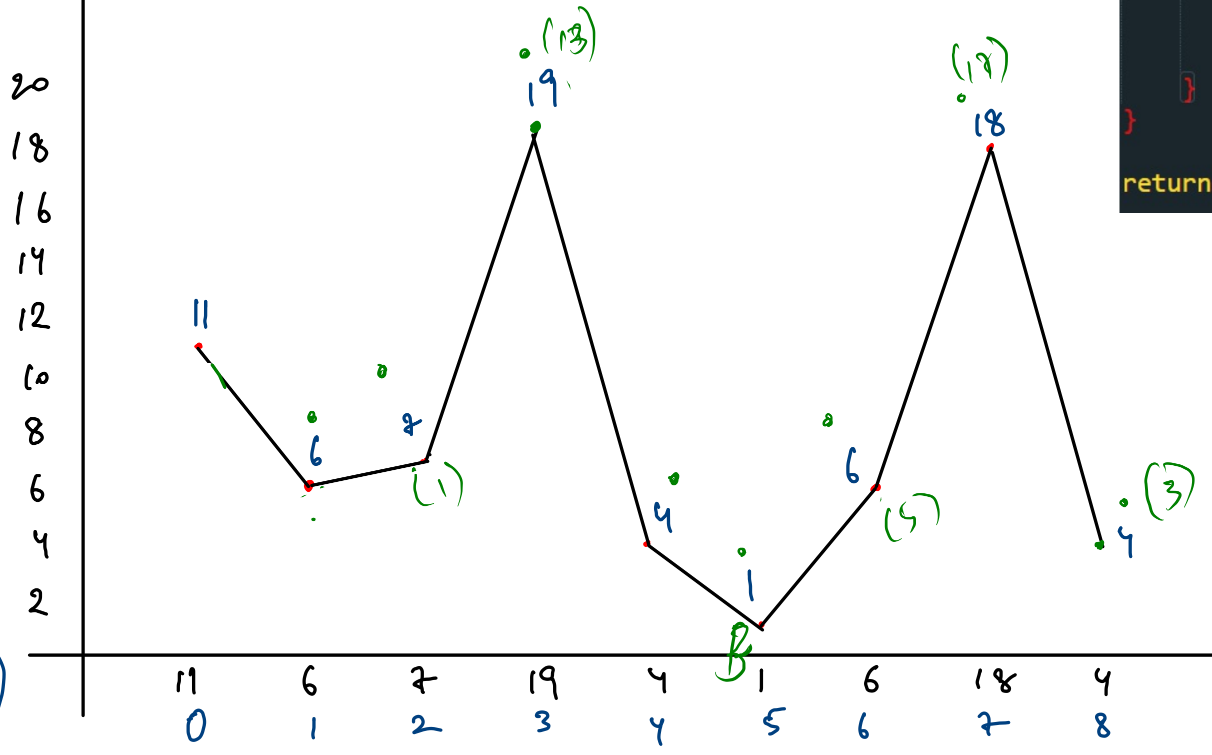
Profit → SP - CP

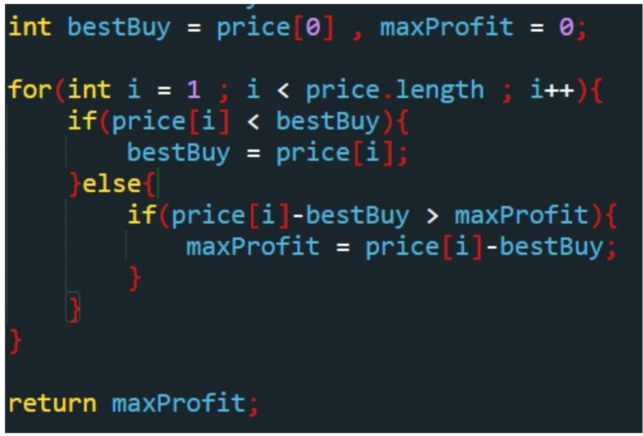
⇓

Maximum

SP >>> CP

Max Profit = ~~1~~ ~~13~~ 17





```
int bestBuy = price[0] , maxProfit = 0;

for(int i = 1 ; i < price.length ; i++){
    if(price[i] < bestBuy){
        bestBuy = price[i];
    }else{
        if(price[i]-bestBuy > maxProfit){
            maxProfit = price[i]-bestBuy;
        }
    }
}

return maxProfit;
```

Profit => 0+0 +9+0 +2 +2+8 => 21

Greedy — Max, Min

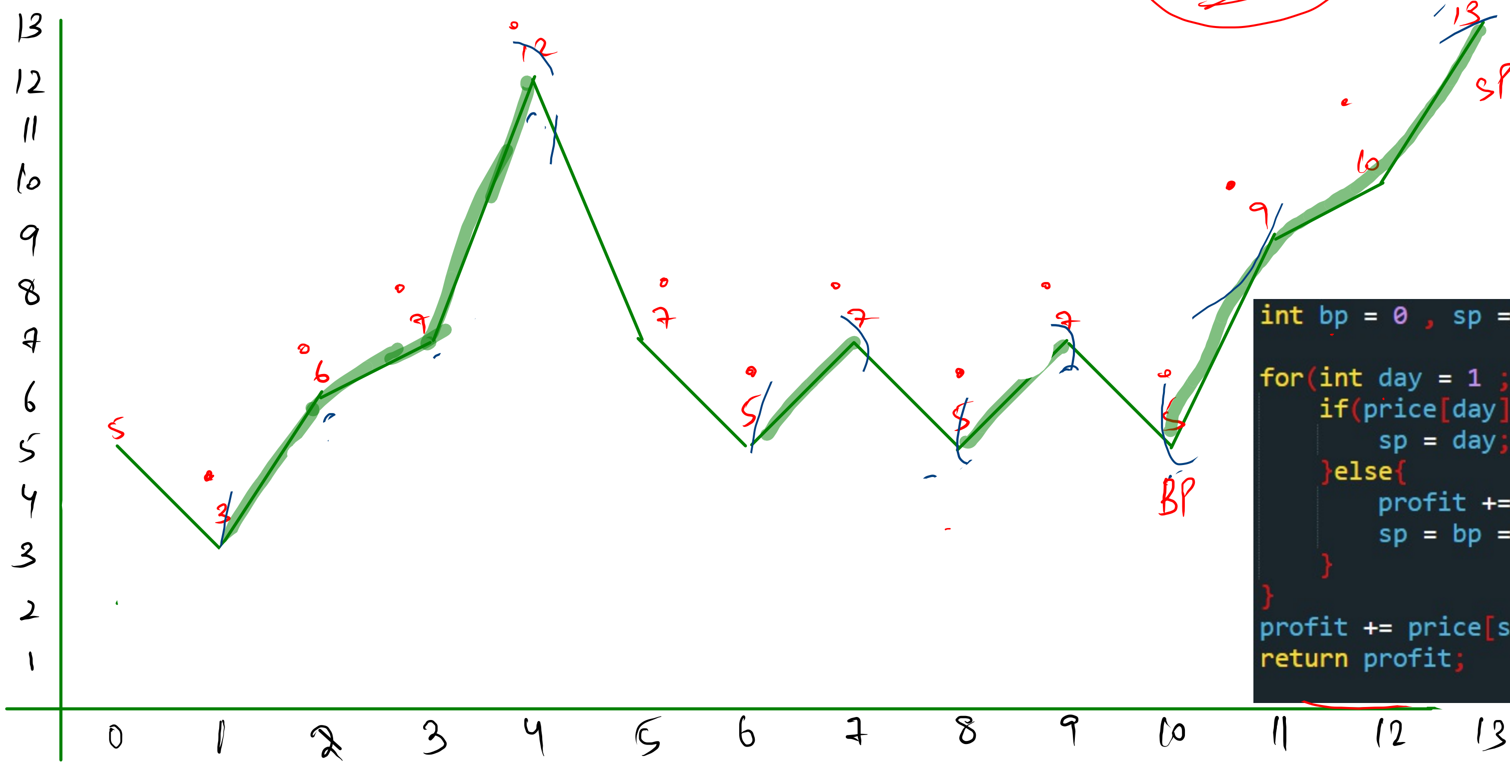
Profit
↳ ∞ transactions

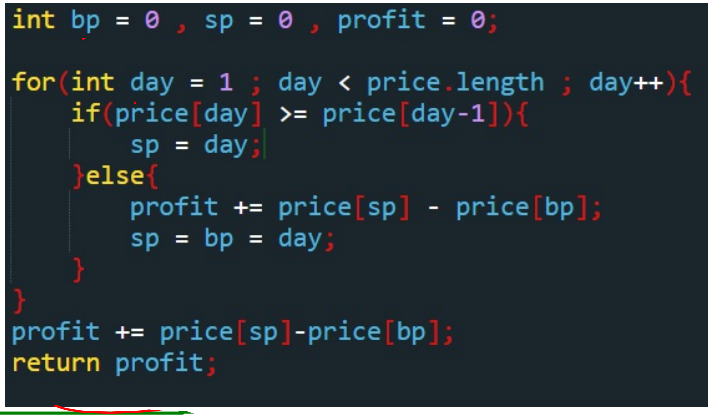
B S B S B S ✓

B B S S ✗

⇓

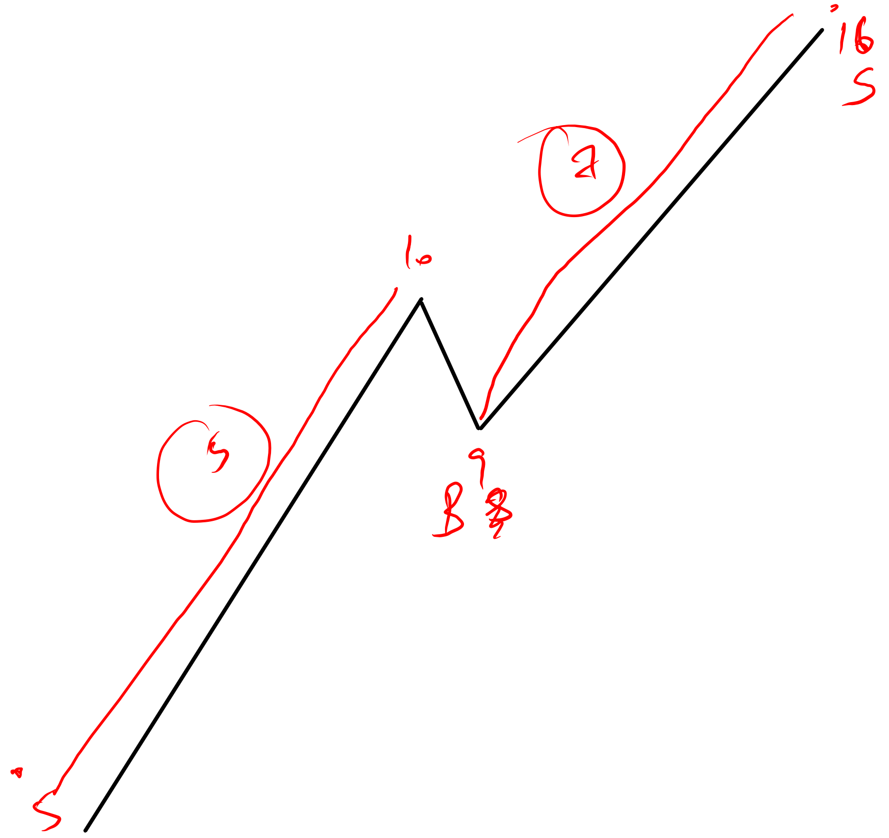
Overlapping





```
int bp = 0 , sp = 0 , profit = 0;

for(int day = 1 ; day < price.length ; day++){
    if(price[day] >= price[day-1]){
        sp = day;
    }else{
        profit += price[sp] - price[bp];
        sp = bp = day;
    }
}
profit += price[sp]-price[bp];
return profit;
```



Maximum Profit 1 trans allowed ⇒ 11

= = ∞ = = ⇒ $\boxed{12}$

DP =

12

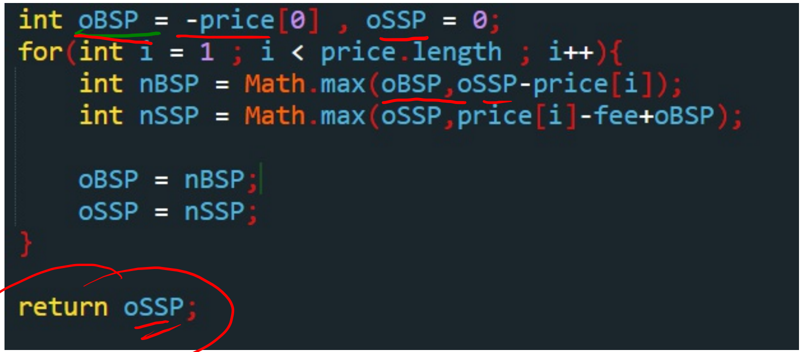
10 15 17 20 16 18 22 20 22 20 23 25

~~3~~

Transaction → ∞

Tn ↳ fee

BS

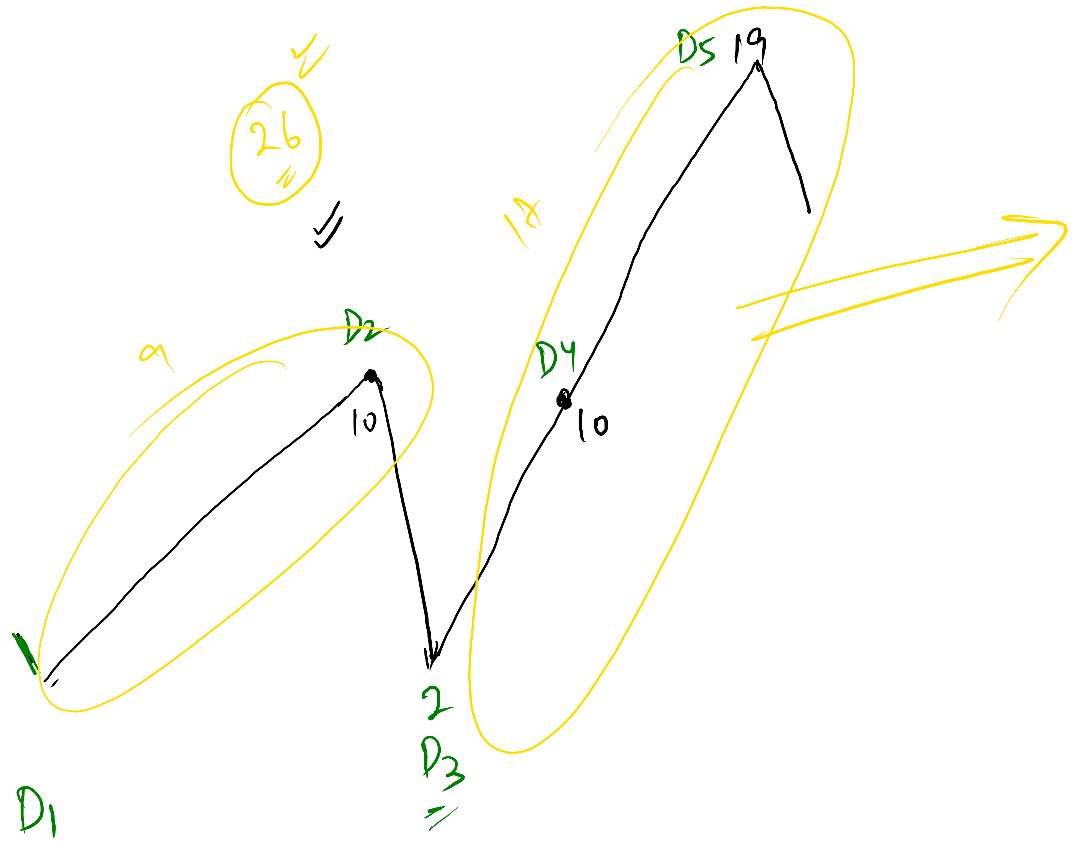


```
int oBSP = -price[0] , oSSP = 0;
for(int i = 1 ; i < price.length ; i++){
    int nBSP = Math.max(oBSP,oSSP-price[i]);
    int nSSP = Math.max(oSSP,price[i]-fee+oBSP);

    oBSP = nBSP;
    oSSP = nSSP;
}

return oSSP;
```

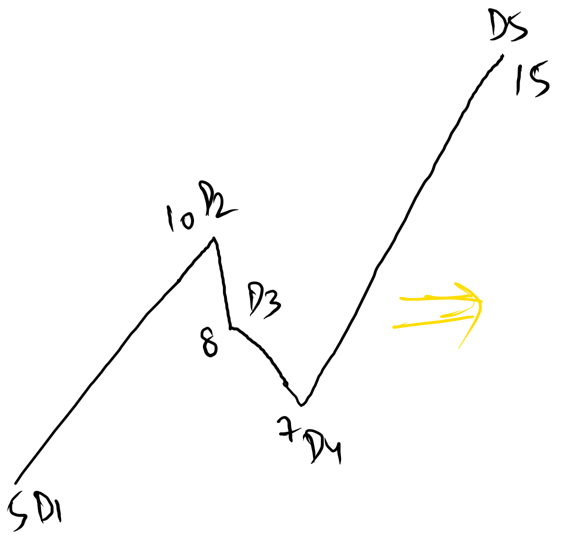
		BUY STATE	SELL STATE
0	10	-10	0
1	15	-10	2
2	17	-10 ~~2-17 => -15~~	4 ~~2~~
3	20		
4	16		
5	18		
6	22		
7	20		
8	22		
9	23		
10	25	-	-



$B_1\ S_2\ C_3\ B_4\ S_5 \Rightarrow \boxed{14}$

$B_3\ S_5 \Rightarrow 17$

$B_1\ S_5 \Rightarrow 18$



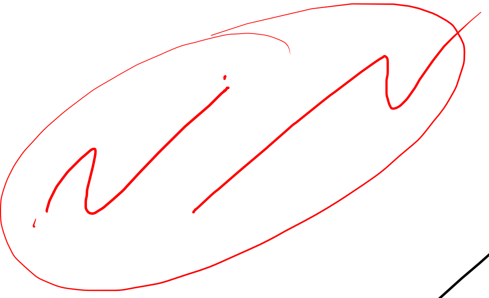
~~10~~

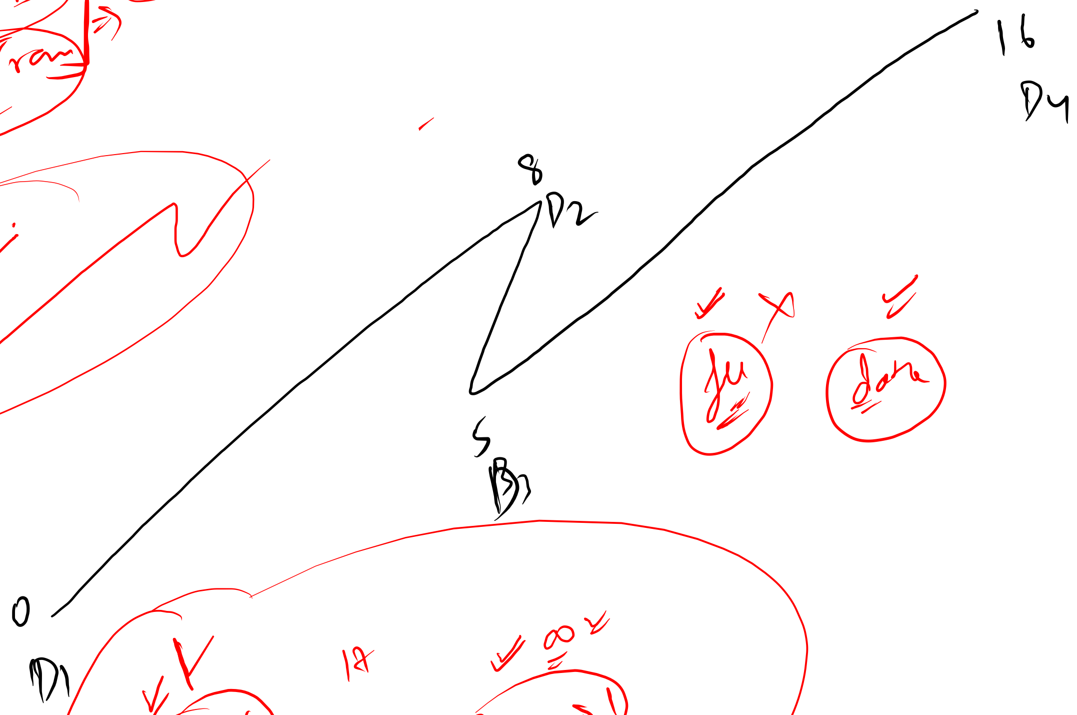
$B_1\ S_5 \Rightarrow 10$

$B_1\ S_2\ C_3\ B_4\ S_5 \Rightarrow 13$

1 Trans

∞ Trans ⇒ DP





fee data

T fee ⇒ 10

∞

T fee ⇒ 1

17

7

8

5

4

MP → 1 Trans		MP - ∞ Trans
fee ⇒ 6,	$B_1 S_4 \rightarrow \boxed{10}$	$B_1 S_2 B_3 S_4 \rightarrow \boxed{7}$
fee ⇒ 3,	$B_1 S_4 \Rightarrow \boxed{13}$	$B_1 S_2 B_3 S_4 \Rightarrow 8-3 + 11-3 \Rightarrow \boxed{13}$
fee ⇒ 2,	$B_1 S_4 \Rightarrow \boxed{14}$	$B_1 S_2 B_3 S_4 \Rightarrow 8-2 + 11-2 \Rightarrow \boxed{15}$

12

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

$$\begin{tikzcd}
\text{Price} + \text{oBSP} \arrow[rd] & & \text{oSSP} \arrow[ld] \\
& \text{max} \arrow[d] & \\
& \text{nSSP} &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{oBSP} \arrow[rd] & & \text{oCSP} - \text{Price}[i] \arrow[ld] \\
& \text{Max} \arrow[d] & \\
& \text{nBSP} &
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{\text{oSSP}} \arrow[d] \\
\text{nCSP}
\end{tikzcd}$$

		BSP	SSP	CSP
0	10	B_0 −10	− 0	− 0
1	15	B_0 −10	5 B_0S_1	− 0
2	17	B_0 −10	B_0S_2 7	5 $B_0S_1C_2$
3	20	B_0 −10	10 B_0S_3	7 $B_0S_2C_3$
4	16	$B_0S_2C_3B_4$ −9	10 B_0S_3	10 B_0S_3
5	18	−8 $B_0S_2C_3B_5$	10 B_0S_3	10 $B_0S_3C_5$
6	22	−8 $B_0S_2C_3B_5$	$B_0S_2C_3B_5S_6$ 14	10 B_0S_3
7	20	−8	14	14
8	22	−8	14	14
9	20	−6	14	14
10	23	$\bigcirc{-6}$	17	17
11	25	−6	$\boxed{19}$	17

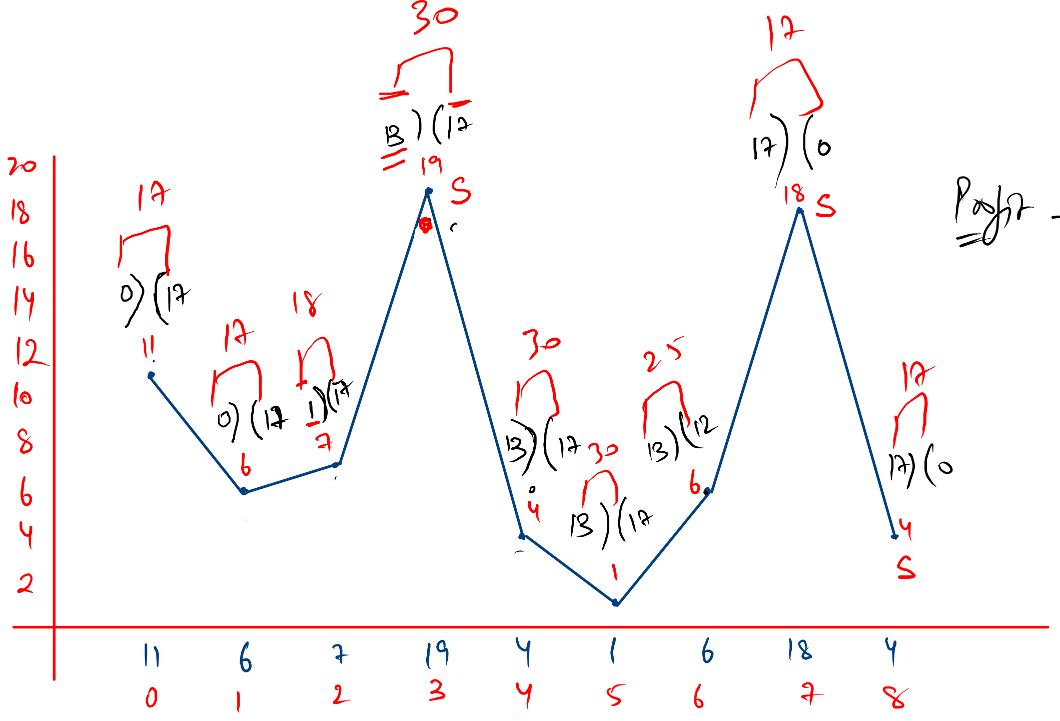
$\xrightarrow{}$ Ans

9
11 6 7 19 4 1 6 18 4

$\boxed{2T}$

Profit → 1 Transaction

$\boxed{CM}$



	0	1	2	3	4	5	6	7	8
sd =	0	0	1	13	0	0	5	17	3
bd =	8	13	12	0	14	17	12	0	0
(C) sd ⇒	0	0	1	13	13	$\boxed{13}$	13	17	17
(C) bd =	17	17	17	17	17	$\boxed{17}$	12	0	0

Profit → (Buy) - sell

$$\frac{(PSD)/MP}{1} \Big) \Big(\frac{(PBD)/MP}{1}$$

Day

Max 2 Trans

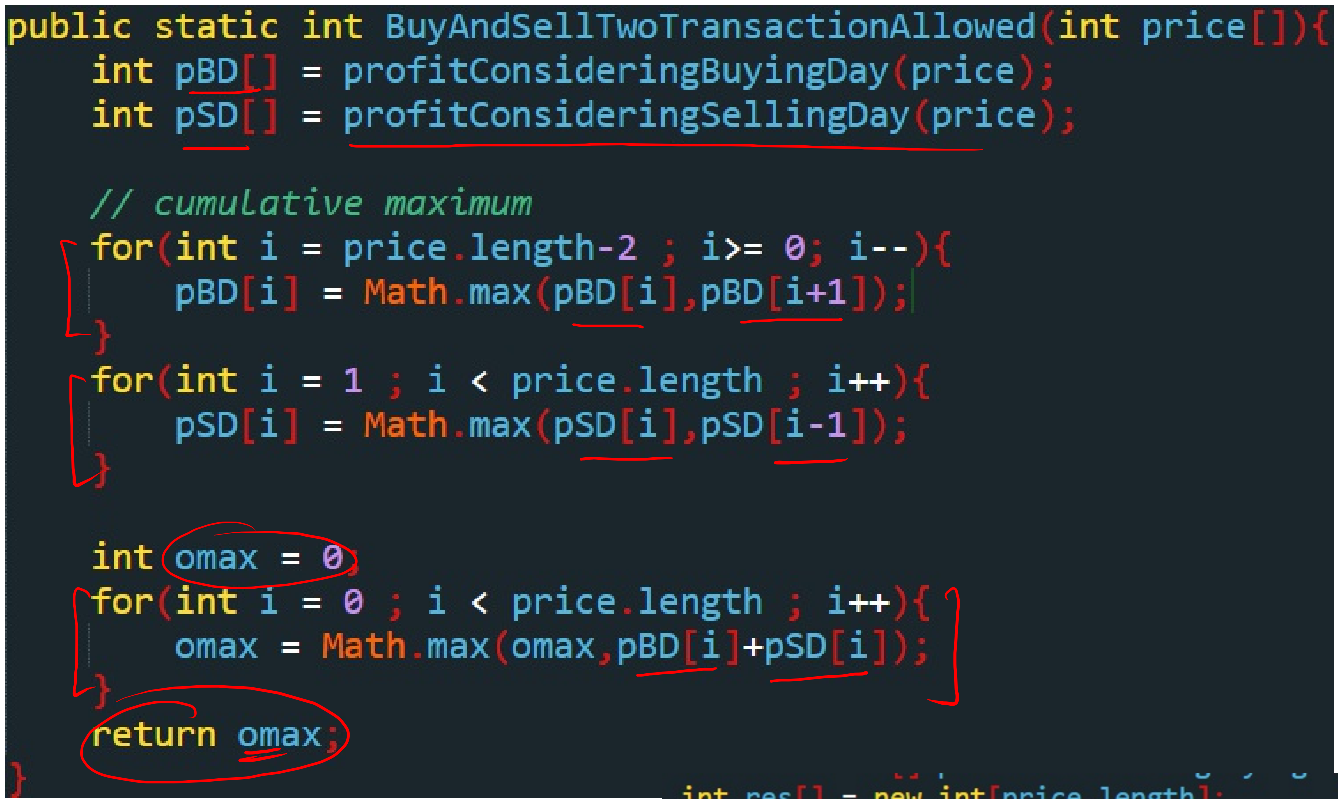
$\boxed{30}$

n ⇒ 9

	0	1	2	3	4	5	6	7	8
price	11	6	7	19	4	1	6	18	4
pBD=	17	17	17	17	17	17	12	0	0
pSD=	0	0	1	13	13	13	13	17	17

$\text{omax} = \cancel{0}\ \cancel{17}\ \cancel{18}\ 30$

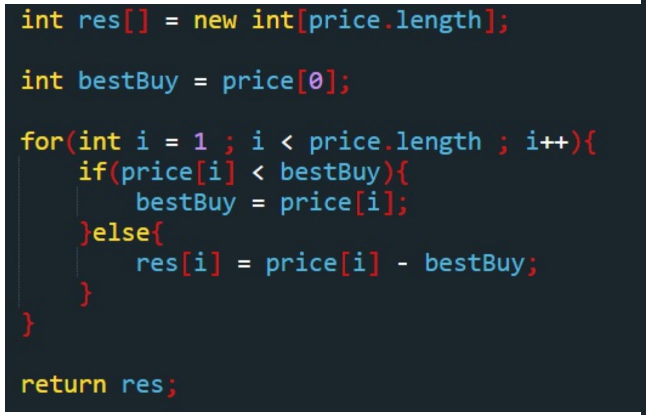
$\boxed{30}$



```java
public static int BuyAndSellTwoTransactionAllowed(int price[]){
    int pBD[] = profitConsideringBuyingDay(price);
    int pSD[] = profitConsideringSellingDay(price);

    // cumulative maximum
    for(int i = price.length-2 ; i>= 0; i--){
        pBD[i] = Math.max(pBD[i],pBD[i+1]);
    }
    for(int i = 1 ; i < price.length ; i++){
        pSD[i] = Math.max(pSD[i],pSD[i-1]);
    }

    int omax = 0;
    for(int i = 0 ; i < price.length ; i++){
        omax = Math.max(omax,pBD[i]+pSD[i]);
    }
    return omax;
}
```

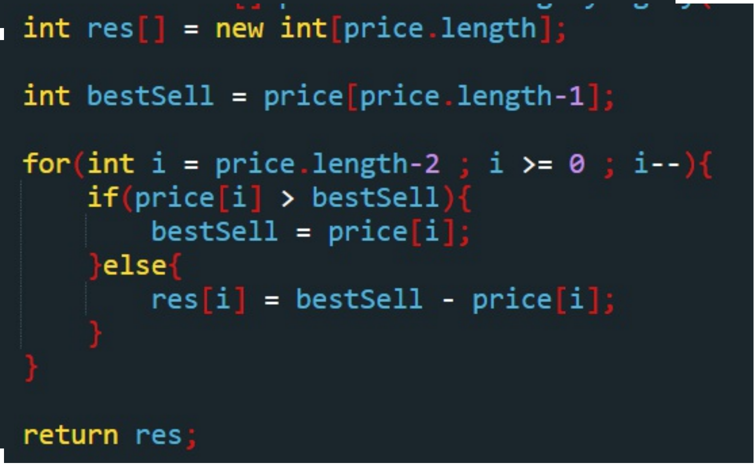


```java
int res[] = new int[price.length];

int bestBuy = price[0];

for(int i = 1 ; i < price.length ; i++){
    if(price[i] < bestBuy){
        bestBuy = price[i];
    }else{
        res[i] = price[i] - bestBuy;
    }
}

return res;
```

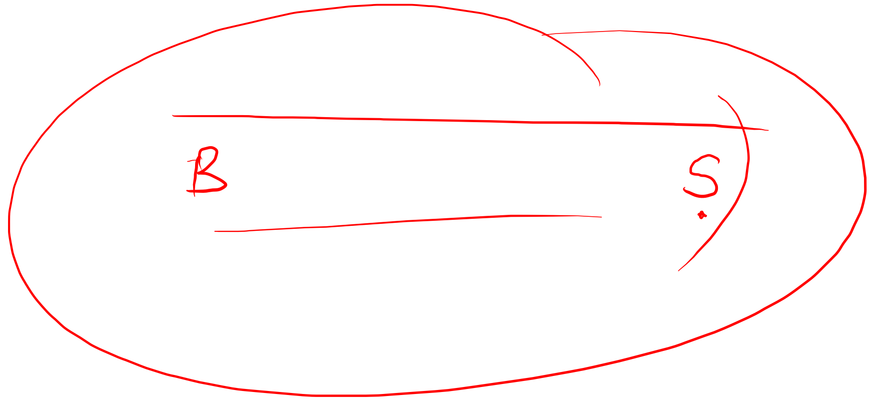
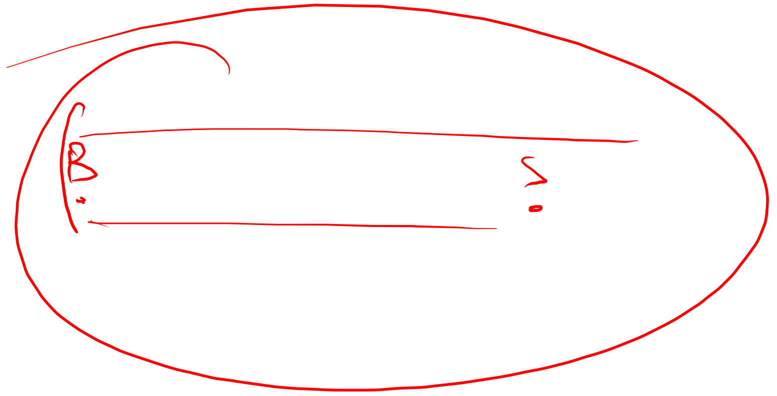


```java
int res[] = new int[price.length];

int bestSell = price[price.length-1];

for(int i = price.length-2 ; i >= 0 ; i--){
    if(price[i] > bestSell){
        bestSell = price[i];
    }else{
        res[i] = bestSell - price[i];
    }
}

return res;
```


B
S

B
S