Buy and Sell Stocks III

Part II → Infinite Transcation

part III → 2 Transcation

↳ here we are putting the bound & this we can relate to the knapsack where we are bounding of our bag capacity

```
f(idx, buy, cap)        → 2 (initially)
{
    if( cap == 0) return 0;
    if( idx == n) return 0;

    if(buy)
    {
    return max( -prices[idx] + f(idx+1, 0, cap);   → money goes into market
                 0 + f(idx+1, 1, cap) );
    }
    else
    {
    return max( prices[idx] + f(idx+1, 1, cap-1);   → money get from market
                 0 + f(idx+1, 0, cap) );
    }
```

Memoization

States

idx	buy	cap
↓	↓	↓
0 to N	2	0/1/2 = 3

(0/1)

dp[N][2][3]

T.C. = O(N*2*3)

S.C. = O(N*2*3) + O(N)

Tabulation

(1) Base cases

(2) Changing parameters

i

buy

cap

(3) copy paste recurrance

(1) Base cases

(i) cap = 0 that means idx & buy can be anything

```
for(idx=0 to n-1)
    for(buy 0 to 1)
        dp[idx][buy][0] = 0
```

(ii) idx=n then buy & cap can be anything

```
for( buy=0 to 1)
    for(cap = 0 → 2)
        dp[n] [buy] [cap] = 0;
```

(2) Changing parameters

i ⟶ n-1 to 0

buy ⟶ 0 to 1

cap ⟶ 0 to 2

(3) copy paste recurrance

(4) return dp[0][1][2];

---

other sol^n dp[N][4] → 2 transaction

f(idx, transaction)

```
{if(idx==n || transaction==4)
    return 0;
```

B S B S

0 1 2 3

transaction no. = even then buy

odd then sell

```
if(trans%2==0) //buy
    max( -prices[idx]+f(idx+1, trans+1)
         0+f(idx+1, trans) )
else //sell
```

```
else //sell
{ max{ prices[idx] + f(idx+1, trans+1);
       f(idx+1 , trans);

dp[N][4]
```