알고리즘

# [LeetCode - 123] Best Time to Buy and Sell Stock III

회색바지여우 | 2019. 12. 16. 19:03

Stock 시리즈 3탄이다. 이번에는 두번만 사고 팔아서 최대의 Profit 을 만들면 된다.

시작해보자.

X = [3,3,5,0,0,3,1,4] 이 Input 으로 주어졌다고하면, 눈대중으로 0원에 사서 3원에 팔고 1원에사서 4원에 팔아서
Max Profit=6이라는 답을 얻을수 있다.

X = [1,2,3,4,5] 일때 max profit 이 4인걸 보면 주식을 한번사면 다시팔때까지는 주식을 살 수 없다는 걸 확인 할 수 있다.
(물론 문제에도 써있다)

일단 두번의 사고팔고가 필요하기 때문에 각시점에서의 best profit을 b1,s1,b2,s2 라고 하자.
좀더 이해하기 쉽게 추가하자면
b1은 첫 사고팔고에서 살때의 bestprofit(당연히 쌀수록 좋다)
s1은 첫번째 사고팔고에서 얻을수 있는 수익
b2는 s1에서 두번째 사고팔고에서 살때의 가격을 뺀 값
s2 는 첫번째 사고팔고+두번째 사고팔고이다

접근방법은 크게 3단계 정도이다.
1. index가 증가할때마다 b1,s1,b2,s2 를 새로 갱신할것이냐 말것이냐이다.
  -> 여기서 갱신 한다는 말은 기존의 사고팔고시점을 변경한다는 것이다.
    예를 들어 1,2,1,2 이렇게 있을때 최대 수익은 2이다.
    이 때 index가 증가하면서 1,2,1,2,5 가 된다면 최대수익은 5가 되고 갱신한것이다.
    만약 1,2,1,2,0 이라면 최대수익은 그대로 2이고 갱신 하지 않은 것이 된다.
2. 새로 갱신하는 기준은 무엇인가 이다.
  -> 기준은 기존 b는 최대한 작아지게, s는 최대한 커지게 하는 것이다.

쉽게 보기위해 D[i] 를 i번째 에서 얻을 수 있는 가장 큰 profit 이라고 하면
D[0] = 0
b1=-3 ( 3원짜리 샀으니까 profit 은 -3)
s1=0 (3원에 사고 3원에 팔음 이라고 이해하면쉬움)
b2= -3 (첫번째 사고팔고에서 0원 얻은거에서 + 3원에샀으니 -3)
s2= 0 (3원에 산거 다시 3원에 팔음)
D[1]=0
새로 생긴 값이 3이니 위와 동일

D[2]=2
여기서부터 조금 생각을 해야 한다.
일단 새로 생긴 값은 5이다.
b1=-3 (5원주고 사는거보다 당연 3원이 싸니까, 당연히 기존꺼 유지)
s1=2 (기존의 첫 사고팔고에서 이득이 0원이었는데 5원에 판다고 하면 -3+5=2 이므로 갱신)
b2=-3 (기존 사고팔고+사고 가 -3 이었는데 갱신한다 치면 s1-5=-3 이므로 갱신하든말든 상관없다)
s2=2 ( i=1일때 두번째 사고팔고 까지의 최대수익이 0원이었는데, 갱신하면 2가 되므로 갱신한다.)

D[3]=2
b1=0 ( 기존 b1=-3 이었으므로 새로들어온 0으로 갱신해주는 것이 이득이 최대가 될수있다.)
s1=2 (기존 s1=2, 새로들어온 0으로 갱신해봤자 b1=0이므로 profit 이 0원이 된다. 당연히 기존시점 profit 을그대로 둔다)
b2=2 (기존 b2=-3, 새로들어온 0을 사용해서 두번째 사는시점을 갱신하면 첫번째 사고팔고까지의 이득 2 - 0 이므로 갱신)
s2=2 (기존 s2=2 , 새로들어온 0을 사용해 갱신해도 b2까지의 최대 profit +0 =2 이므로 갱신하든말든 상관없다.)

...

위와같이 나아가면서 index가 늘어나면서 새로운 값이 들어올때마다.
이전 index 까지의 b1,s1,b2,s2 시점에서 profit 과 비교하여 새로운 값을 사용한 사고팔고가 profit 을 더 크게 만들어 준다면 값을 갱신해주면된다.

최종값은 두번째 사고 팔고 까지의 profit 이 담겨있는 s2를 리턴해주면 된다.

```
class Solution {
public:

  int maxProfit(vector<int>& prices) {
    int b1=-2710000;
    int s1=0;
    int b2=-2710000;
    int s2=0;

    for(int i=0; i<(int)prices.size(); i++)
    {
      b1=max(b1, -prices[i]);
      s1=max(s1,prices[i]+b1);
      b2=max(b2,s1-prices[i]);
      s2=max(s2,b2+prices[i]);
    }

    return s2;
  }

  int max(int a, int b)
  {
    if(a>b) return a;
    else return b;
  }
};
```





'알고리즘' 카테고리의 다른 글

[LeetCode - 234] Palindrome Linked List (0) 2020.01.03
[LeetCode - 125] Valid Palindrome (0) 2019.12.31
[LeetCode - 124] Binary Tree Maximum Path Sum (0) 2019.12.30
[LeetCode - 123] Best Time to Buy and Sell Stock III (0) 2019.12.16
[LeetCode-122] Best Time to Buy and Sell Stock II (0) 2019.12.10
[LeetCode-123] Best Time to Buy and Sell Stock (1) 2019.11.27

Tag best time to buy and sell stock삭제#Kadane's algorithm삭제#LeetCode삭제#leetcode 123삭제

'알고리즘' Related Articles more

[LeetCode - 125] Valid Palindrome 2019.12.31
[LeetCode - 124] Binary Tree Maximum Path Sum 2019.12.30
[LeetCode-122] Best Time to Buy and Sell Stock II 2019.12.10
[LeetCode-123] Best Time to Buy and Sell Stock 2019.11.27

## 0 Comments



Prev 1 · · · 7 8 9 10 11 12 13 Next