

Dynamic Programming

Edit Distance

"horse" "ros"

horse ros

Y_S
Y_{OS} → 4 steps

horse

rose

rose

red

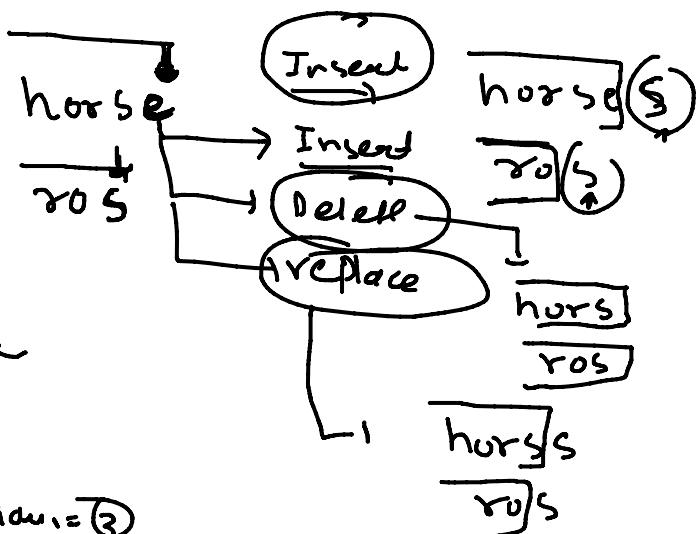
3 instead

abc

Mann Probler

A hand-drawn diagram of three cards stacked vertically. The top card has the letters "abc" written on it. The middle card has two short horizontal lines with arrows pointing to them, indicating they are blank. The bottom card also has two short horizontal lines with arrows pointing to them, indicating they are blank.

- Insert a character
- Delete a "
- Replace "



```
int edit ( String s1, String s2, int idu., int idur )
```

if (idu. == 0)
return idu;

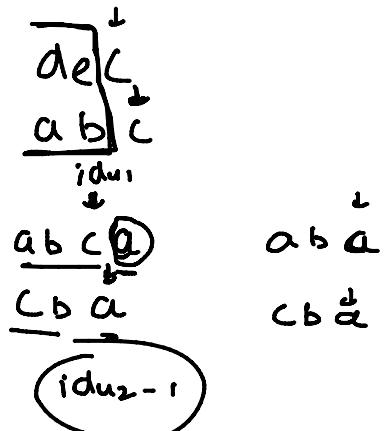
W

i6 ($\sin z = 0$)

L yctea idu.

$\rightarrow \text{if } (\text{dp}[\text{idu}_1][\text{idu}_2] == -1) \text{ return dp}[\text{idu}_1][\text{idu}_2]$

$\text{ih} \mid \text{str}_1(\text{idu}_1 - 1) = \text{str}_2(\text{idu}_2 - 1)$



```

if (str1[id1, -1] == str2[id2, -1])
    op[id1][id2] = 
2 return min(edit(str1, str2, id1, -1, id2, -1));
4
else
    int op1 = edit(str1, str2, id1, -1, id2, -1) + 1; //insert
    int op2 = edit(str1, str2, id1, -1, id2 - 1) + 1; //replace;
    int op3 = edit(str1, str2, id1 - 1, id2, -1) + 1 //Delete
    op[id1][id2] = min(op1, min(op2, op3));

```

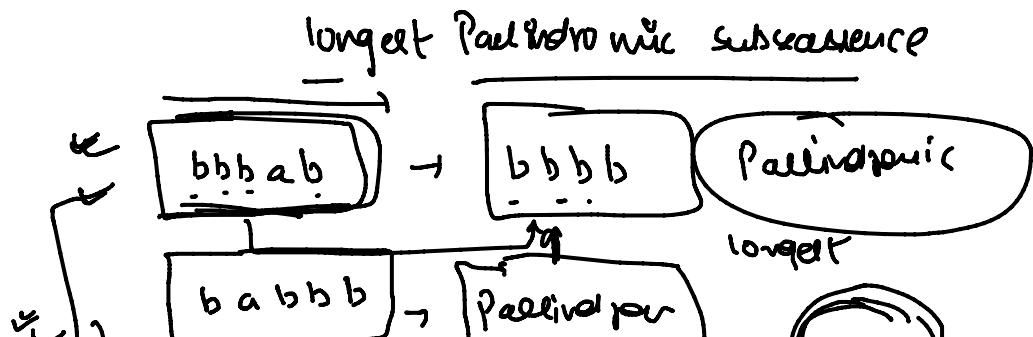
ab ↴
bc ↴
ca ↴

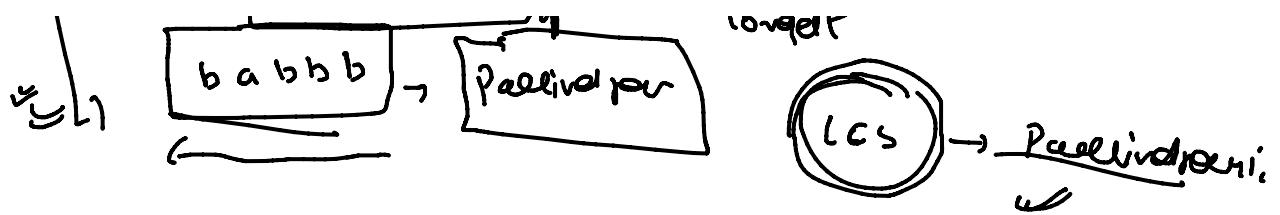
	0	1	2	3	4
0	0	1	2	3	4
1	1				
2	2				
3	3				
4	4				

Final String

Backtracking using
DP Table

C



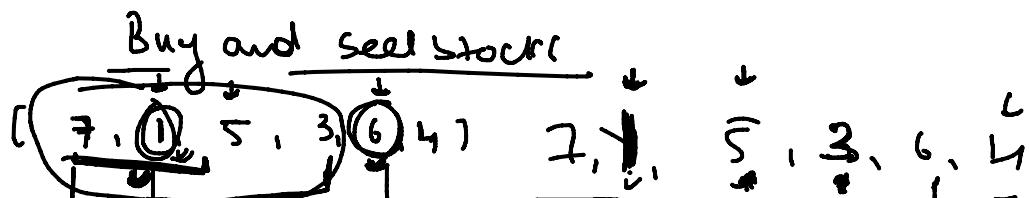


String str;

String str2 = str;

reverse(str2.begin(), str2.end());

return lcs(str1, str2, str1.size(), str2.size());



int buy = 7;
int sell = 0;

sell = 5
buy = 0

For (i=1; i<n; i++)

 sell = max (sell, prices[i] - buy);

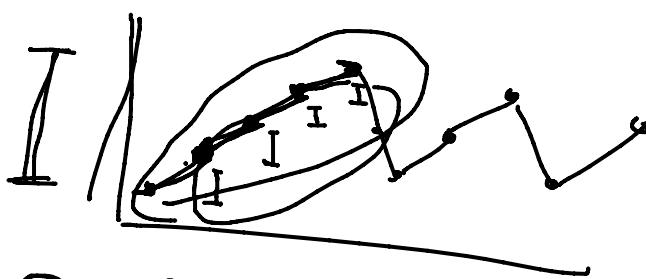
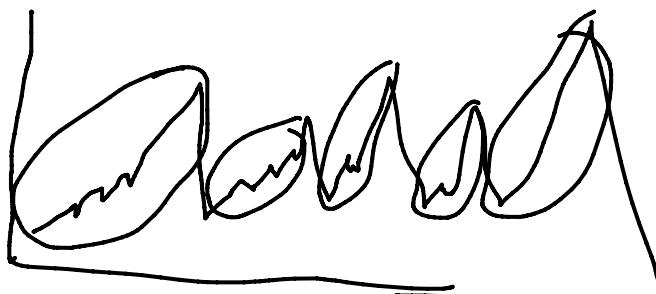
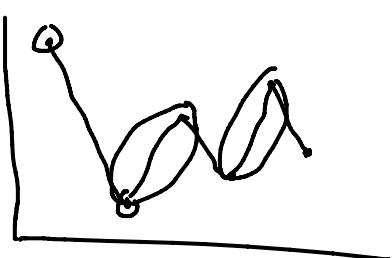
 buy = min (buy, price[i]);

return sell - 5;

Burned cool class - a

Buy and sell stock - 2

[7, 1, 5, 3, 6, 4]



$i \rightarrow i+1$

int profit = 0;

for (i=1; i<n; i++)

{ if (Price[i] > Price[i-1])

 { profit += (Price[i] - Price[i-1]); }



}

[7, 1, 5, 3, 6, 4]

dp[i][1]

[1, 3, 2, 8, 4, 9] fees = 2

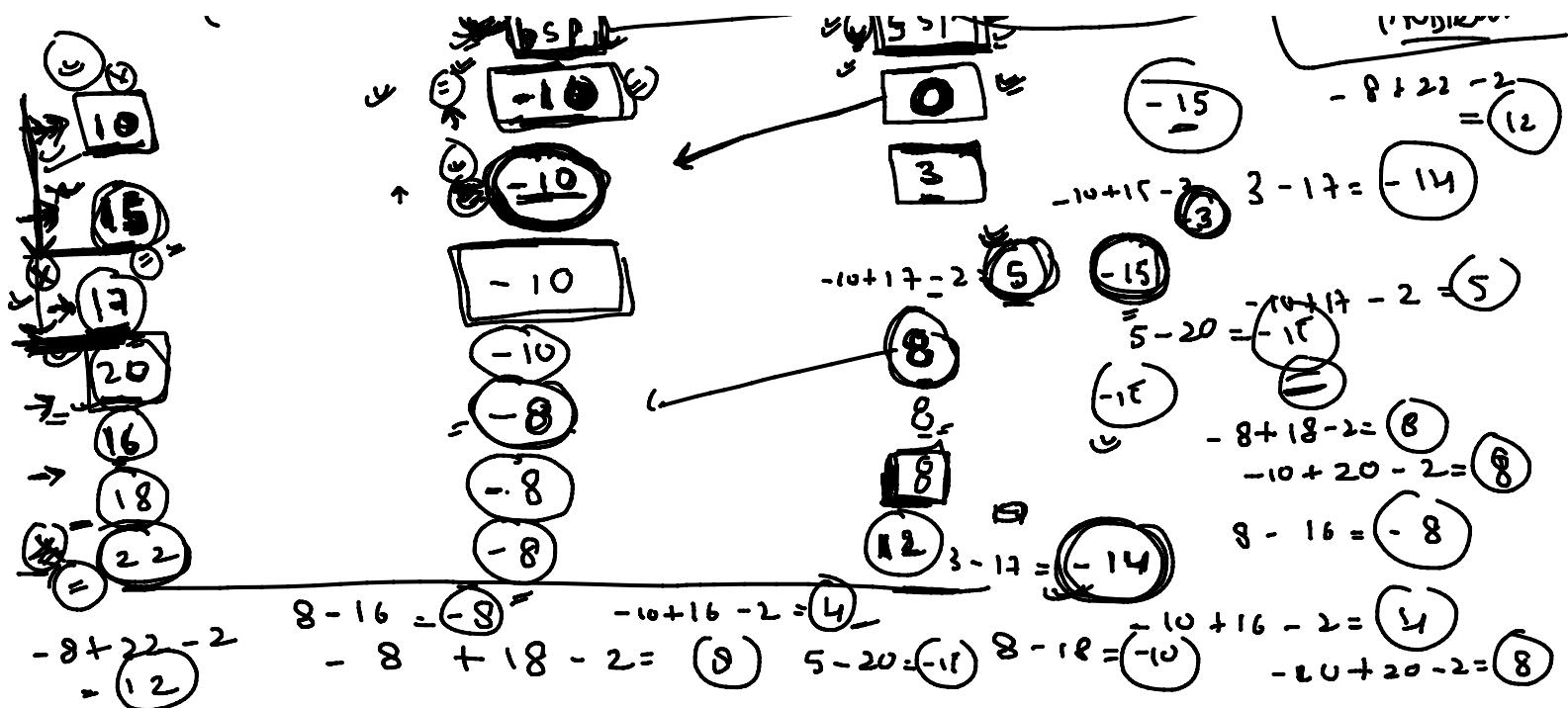
Profit



SP
-10%

buy or sell
S P A T H E
-15%

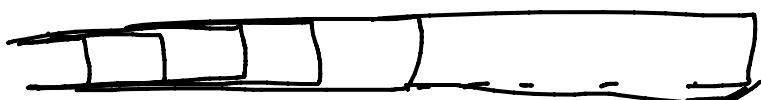
include transaction problem
- 8 + 22 - 2



$c_{IP[i]}$ =

$c_{IP[i][2]}$

$c_{IP[i][0]}$ = Maximum Profit
with holding
one stock



```

int bSP = -arr[0];
int ssP = 0;
int hSP = 0;

for (i=1; i<n; i++) {
    int nbsp = 0
    int nsp = 0,
        if (ssP - arr[i]) > bSP)
            nbsp = ssP - arr[i];
    }
}

```

Q10

$nbsp = bsp;$

if (bsp + aw(i) - c > ssp)

$nssp = bsp + aw(i) - c;$

 2

else

$nsp = ssp;$

$bsp = nbsp;$

$csp = nsp;$

 1

return 