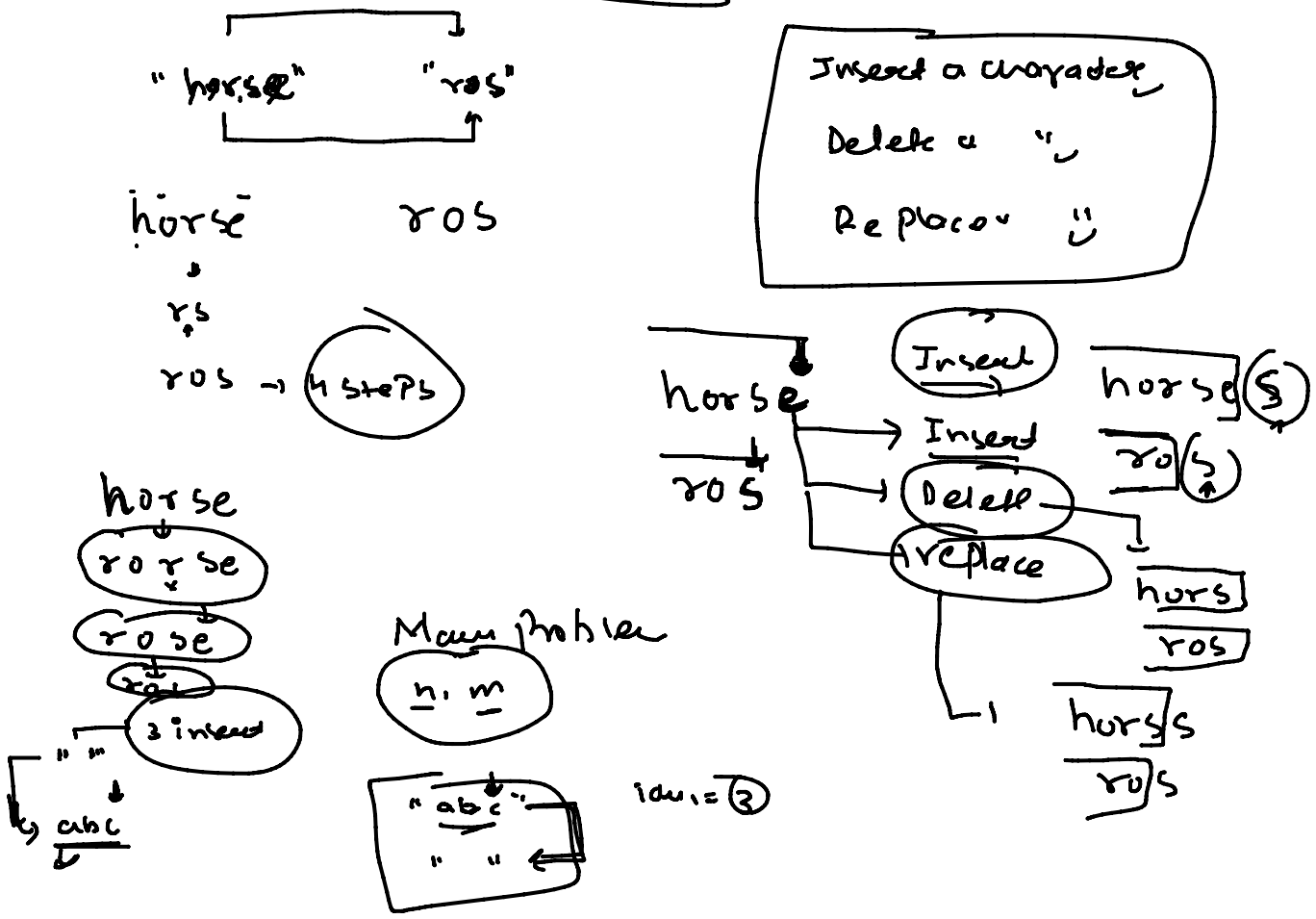
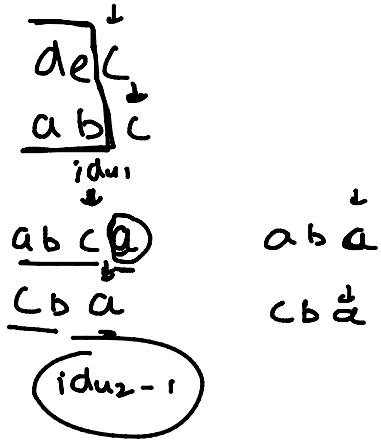
Class - 84

Dynamic Programming

Edit Distance




```
int edit ( String s1, String s2, int idx1, int idx2 )
{   if (idx1 == 0)
    {   return idx2;
    }
    if (idx2 == 0)
    {   return idx1;
    }
    → if (dp[idx1][idx2] != -1) return dp[idx1][idx2]
    if ( str1[idx1 - 1] == str2[idx2 - 1])
```



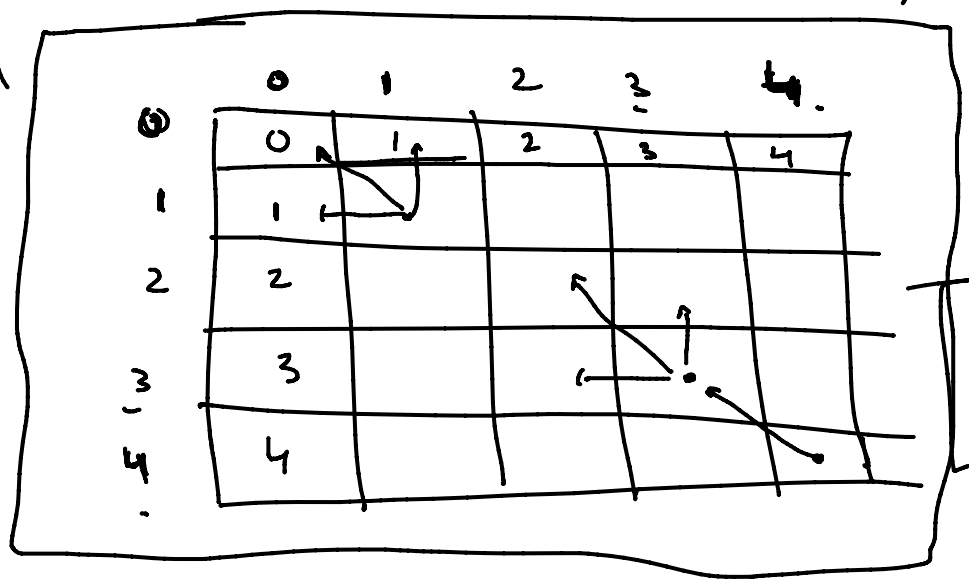
```
if ( str1[idx1 - 1] == str2[idx2 - 1])
    dp[idx1][idx2] =
{   return edit ( str1, str2, idx1 - 1, idx2 - 1);
}
else
{   int op1 = edit ( str1, str2, idx1 - 1, idx2 - 1) + 1;
    //Replace;
    int op2 = edit ( str1, str2, idx1 - 1, idx2 ) + 1;
    //Delete
    int op3 = edit ( str1, str2, idx1, idx2 - 1) + 1
    dp[idx1][idx2] =
    return min(op1, min(op2, op3));
```

i , j-1

i-1 , j-1

i-1 , j

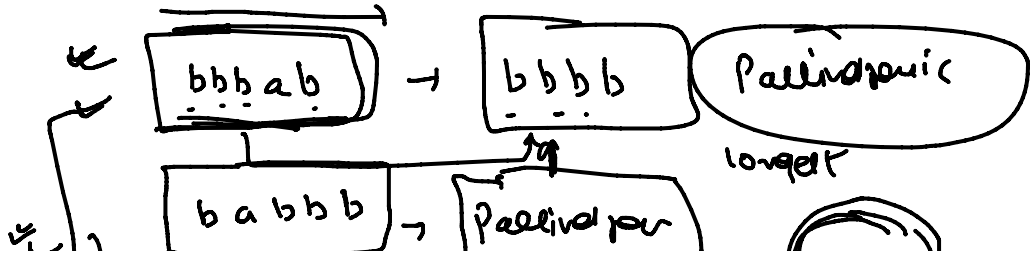
ab
bca

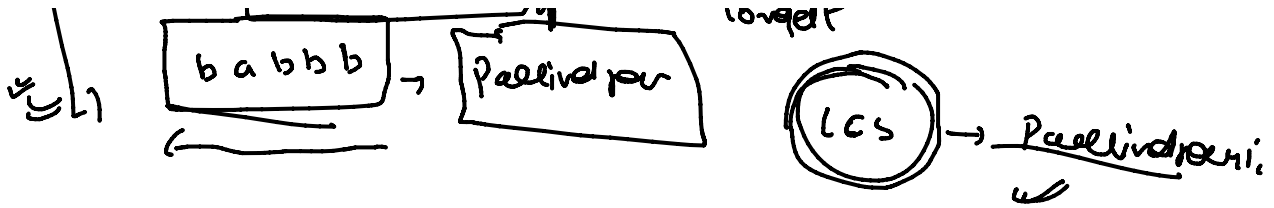


Final String

Backtracking using DP Table

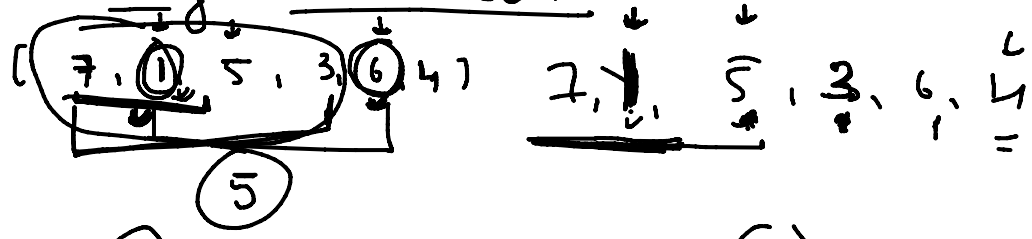
longest Palindromic subsequence






```
String str1;
String str2 = str1;
reverse( str2.begin(), str2.end());
return lcs( str1, str2, str1.size(), str2.size());
```

Buy and sell stocks



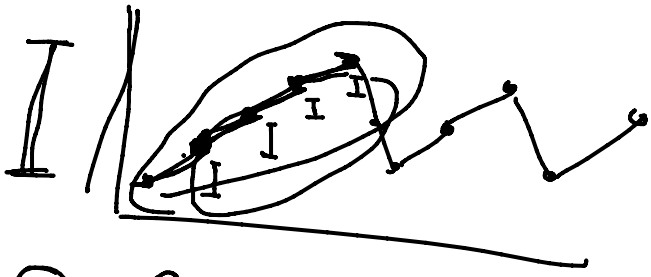

```
int buy = 7;
int sell = 0;
For (i=1; i<n; i++)
{
    sell = max( sell, Prices[i] - buy);
    buy = min( buy, Price[i]);
}
return sell;
```

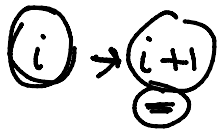
sell = 5
buy = 1

Buy and sell

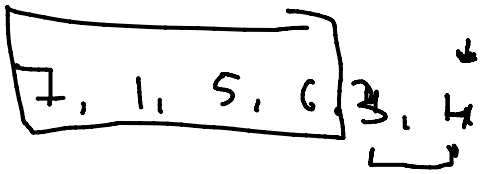
Buy and sell stock - 2

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



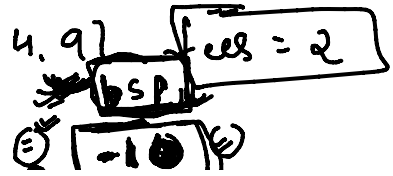


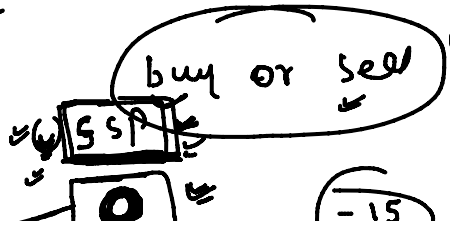
```
int profit = 0;
for (i = 1; i < n; i++)
    if (price[i] > price[i-1])
        profit += (price[i] - price[i-1]);
```



dp[i]

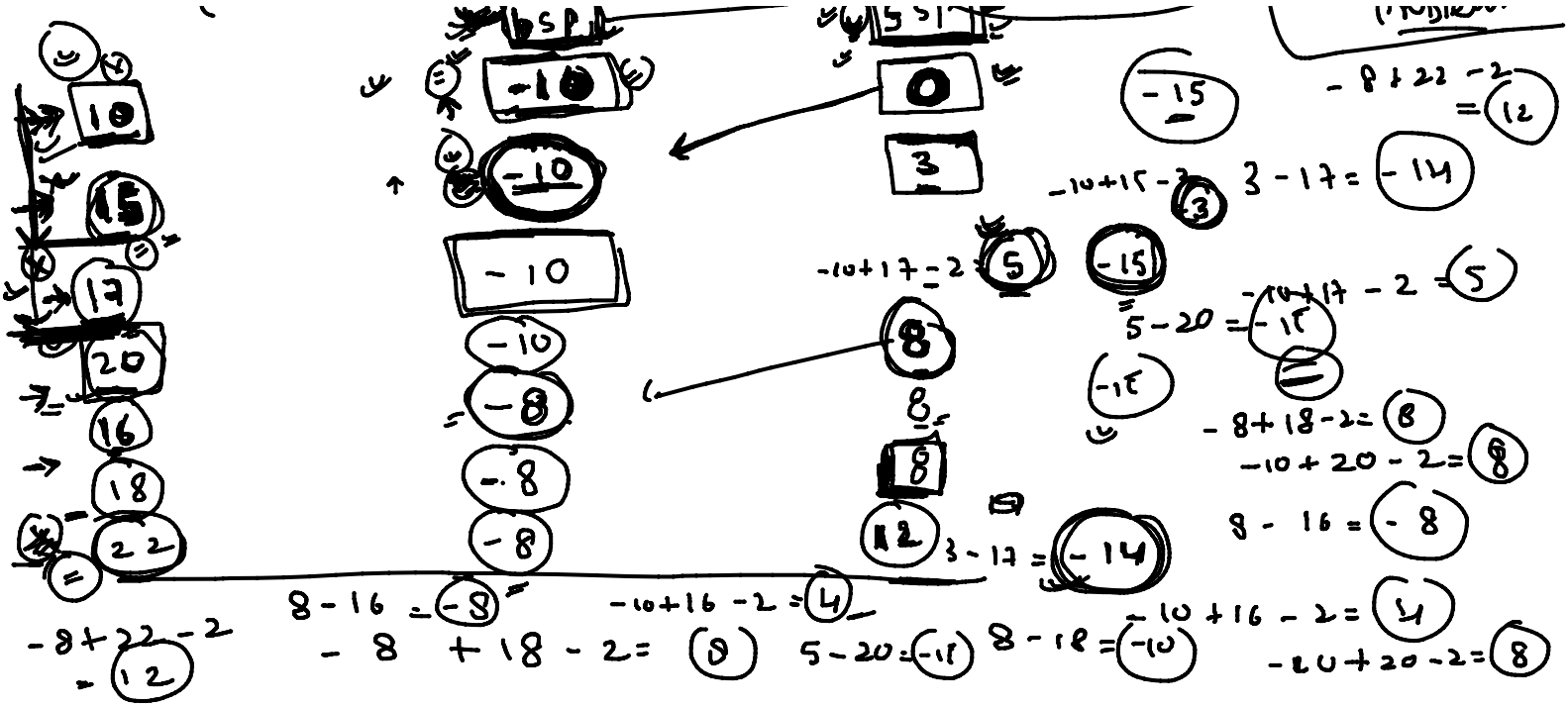
[1, 3, 2, 8, 4, 9]





include | exclude problem

-8 + 22 - 2



dp[i] = dp[i][2]

dp[i][0] = Maximum profit with holding one stock

```
int bsp = -arr[0];
int ssp = 0;

for(i=1; i<n; i++)
{
    int nbsp = 0
    int nssp = 0;
    if(ssp - arr[i] > bsp)
    {
        nbsp = ssp - arr[i];
    }
    else
```

```
        nbsp = bsp;

    if ( bsp + arr[i] - k > ssp )
        {   nssp = bsp + arr[i] - k;
        }
    else
        nssp = ssp;

    bsp = nbsp;
    csp = nssp;
}
return ssp
```