Edit Distance:

```
class Solution {
public:
    int editDistance(int i,int j,string word1,string
word2,vector<vector<int>>&dp){
        if(i<0){
            return j+1;
        }
        if(j<0){
            return i+1;
        }
        if(dp[i][j]!=-1){
            return dp[i][j];
        }

        //match and not match
        if(word1[i]==word2[j]){
            return dp[i][j]= 0+editDistance(i-1,j-1,word1,word2,dp);
        }
        else{
            int insertions=1+editDistance(i,j-1,word1,word2,dp);
            int deletions=1+editDistance(i-1,j,word1,word2,dp);
            int replaces=1+editDistance(i-1,j-1,word1,word2,dp);
            return dp[i][j]=min(insertions,min(deletions,replaces));
        }
    }
    int minDistance(string word1, string word2) {
        int i=word1.size()-1;
        int j=word2.size()-1;
        vector<vector<int>>dp(word1.size(),vector<int>(word2.size(),-1));
        return editDistance(i,j,word1,word2,dp);
    }
};
```

Buy and Sell stock-II

```
class Solution {
public:
    int maximumProfit(int i,int
buy,vector<int>&prices,vector<vector<int>>&dp){
        //base case
        if(i==prices.size()){
            return 0;
        }
```

```
        if(dp[i][buy]!=-1){
            return dp[i][buy];
        }
        //buy and sell
        int profit=0;
        if(buy==1){
            int buy=-prices[i]+maximumProfit(i+1,0,prices,dp);
            int notbuy=0+maximumProfit(i+1,1,prices,dp);
            profit=max(buy,notbuy);
        }
        else{
            int sell=prices[i]+maximumProfit(i+1,1,prices,dp);
            int notsell=0+maximumProfit(i+1,0,prices,dp);
            profit=max(sell,notsell);
        }
        return dp[i][buy]=profit;
    }
    int maxProfit(vector<int>& prices) {
        int i=0;
        int buy=1;
        int n=prices.size();
        vector<vector<int>>dp(n,vector<int>(2,-1));
        return maximumProfit(i,buy,prices,dp);
    }
};
```

Buy and Sell Stock-III

```
class Solution {
public:
    int maximumProfit(int i,int buy,int
k,vector<int>&prices,vector<vector<vector<int>>>&dp){
        //base case
        if(i==prices.size()||k==0){
            return 0;
        }
        if(dp[i][buy][k]!=-1){
            return dp[i][buy][k];
        }
        //buy and sell
        int profit=0;
        if(buy==1){
            int buy=-prices[i]+maximumProfit(i+1,0,k,prices,dp);
            int notbuy=0+maximumProfit(i+1,1,k,prices,dp);
            profit=max(buy,notbuy);
        }
```

```
        else{
            int sell=prices[i]+maximumProfit(i+1,1,k-1,prices,dp);
            int notsell=0+maximumProfit(i+1,0,k,prices,dp);
            profit=max(sell,notsell);
        }
        return dp[i][buy][k]=profit;
    }
    int maxProfit(vector<int>& prices) {
        int i=0;
        int buy=1;
        int k=2;
        int n=prices.size();

vector<vector<vector<int>>>dp(n,vector<vector<int>>(2,vector<int>(3,-1)));
        return maximumProfit(i,buy,k,prices,dp);
    }
};
```

Buy and Sell Stock-IV

```
class Solution {
public:
     int maximumProfit(int i,int buy,int
k,vector<int>&prices,vector<vector<vector<int>>>&dp){
        //base case
        if(i==prices.size()||k==0){
            return 0;
        }
        if(dp[i][buy][k]!=-1){
            return dp[i][buy][k];
        }
        //buy and sell
        int profit=0;
        if(buy==1){
            int buy=-prices[i]+maximumProfit(i+1,0,k,prices,dp);
            int notbuy=0+maximumProfit(i+1,1,k,prices,dp);
            profit=max(buy,notbuy);
        }
        else{
            int sell=prices[i]+maximumProfit(i+1,1,k-1,prices,dp);
            int notsell=0+maximumProfit(i+1,0,k,prices,dp);
            profit=max(sell,notsell);
        }
        return dp[i][buy][k]=profit;
    }
    int maxProfit(int k, vector<int>& prices) {
```

```
         int i=0;
        int buy=1;
        int n=prices.size();

vector<vector<vector<int>>>dp(n,vector<vector<int>>(2,vector<int>(k+1,-1)));
        return maximumProfit(i,buy,k,prices,dp);
    }
};
```

Buy and Sell Stock with Cooldown:

```
class Solution {
public:
   int maximumProfit(int i,int
buy,vector<int>&prices,vector<vector<int>>&dp){
        //base case
        if(i>=prices.size()){
            return 0;
        }
        if(dp[i][buy]!=-1){
            return dp[i][buy];
        }
        //buy and sell
        int profit=0;
        if(buy==1){
            int buy=-prices[i]+maximumProfit(i+1,0,prices,dp);
            int notbuy=0+maximumProfit(i+1,1,prices,dp);
            profit=max(buy,notbuy);
        }
        else{
            int sell=prices[i]+maximumProfit(i+2,1,prices,dp);
            int notsell=0+maximumProfit(i+1,0,prices,dp);
            profit=max(sell,notsell);
        }
        return dp[i][buy]=profit;
    }
    int maxProfit(vector<int>& prices) {
        int i=0;
        int buy=1;
        int n=prices.size();
        vector<vector<int>>dp(n,vector<int>(2,-1));
        return maximumProfit(i,buy,prices,dp);
    }
};
```

Buy and Sell Stock with Transaction Fee:

```cpp
class Solution {
public:
    int maximumProfit(int i,int
buy,vector<int>&prices,vector<vector<int>>&dp,int fee){
        //base case
        if(i==prices.size()){
            return 0;
        }
        if(dp[i][buy]!=-1){
            return dp[i][buy];
        }
        //buy and sell
        int profit=0;
        if(buy==1){
            int buy=-prices[i]+maximumProfit(i+1,0,prices,dp,fee);
            int notbuy=0+maximumProfit(i+1,1,prices,dp,fee);
            profit=max(buy,notbuy);
        }
        else{
            int sell=prices[i]-fee+maximumProfit(i+1,1,prices,dp,fee);
            int notsell=0+maximumProfit(i+1,0,prices,dp,fee);
            profit=max(sell,notsell);
        }
        return dp[i][buy]=profit;
    }
    int maxProfit(vector<int>& prices,int fee) {
        int i=0;
        int buy=1;
        int n=prices.size();
        vector<vector<int>>dp(n,vector<int>(2,-1));
        return maximumProfit(i,buy,prices,dp,fee);
    }
};
```