Design and Analysis of Algorithms
Assignment 3

By Himanshu Sardana

Question 1

Find the length of the Longest Common Subsequence in input: `S1` = "AGGTAB", `S2` = "GXTXAYB"

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
        string s1 = "AGGTAB";
        string s2 = "GXTXAYB";

        int n = s1.size();
        int m = s2.size();

        vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

        for (int i = 1; i <= n; i++) {
                for (int j = 1; j <= m; j++) {
                        if (s1[i - 1] == s2[j - 1]) {
                                dp[i][j] = 1 + dp[i - 1][j - 1];
                        } else {
                                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
                        }
                }
        }

        cout << dp[n][m] << endl;
}
```

Question 2

There are 3 matrices of dimensions 2×1, 1×3, and 3×4, using Matrix Chain Multiplication find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
        vector<int> arr = {2, 1, 3, 4};
        int n = arr.size();
```

```
        vector<vector<int>> dp(n, vector<int>(n, 0));

        for (int len = 2; len < n; len++) {
                for (int i = 1; i < n - len + 1; i++) {
                        int j = i + len - 1;
                        dp[i][j] = INT_MAX;
                        for (int k = i; k < j; k++) {
                                dp[i][j] = min(dp[i][j], dp[i][k] + dp[k +
 ↪  1][j] + arr[i - 1] * arr[k] * arr[j]);
                        }
                }
        }

        cout << "Minimum number of multiplications: " << dp[1][n - 1] <<
 ↪  endl;
}
```

Question 3

Given N = 3 items where each item has some weight and profit associated with it `profit[] = {1, 2, 3}` , `weight[] = {4, 5, 1}` and also given a bag with capacity W=4, [i.e., the bag can hold at most W weight in it]. Put the items into the bag such that the sum of profits associated with them is the maximum possible with the constraint either put an item completely into the bag or cannot put it at all.

```
#include <bits/stdc++.h>

using namespace std;

int main() {
        vector<int> profit = {1, 2, 3};
        vector<int> weight = {4, 5, 1};
        int W = 4;
        int n = profit.size();

        vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));

        for (int i = 1; i <= n; i++) {
                for (int w = 1; w <= W; w++) {
                        if (weight[i - 1] <= w) {
                                dp[i][w] = max(dp[i - 1][w], profit[i - 1]
 ↪  + dp[i - 1][w - weight[i - 1]]);
                        } else {
                                dp[i][w] = dp[i - 1][w];
                        }
                }
        }

        cout << "Maximum profit: " << dp[n][W] << endl;
}
```

Question 4

The Maximal Square Problem involves finding the largest square that can be formed in a given binary matrix. The binary matrix consists of 0s and 1s, where 1s represent parts of the square and 0s represent empty spaces.

```
#include <bits/stdc++.h>

using namespace std;

int main() {
        vector<vector<int>> matrix = {
                {1, 0, 1, 0, 0},
                {1, 0, 1, 1, 1},
                {1, 1, 1, 1, 1},
                {1, 0, 0, 1, 0}
        };

        int n = matrix.size();
        int m = matrix[0].size();

        vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));
        int max_side = 0;

        for (int i = 1; i <= n; i++) {
                for (int j = 1; j <= m; j++) {
                        if (matrix[i - 1][j - 1] == 1) {
                                dp[i][j] = 1 + min({dp[i - 1][j], dp[i][j -
 ↪  1], dp[i - 1][j - 1]});
                                max_side = max(max_side, dp[i][j]);
                        }
                }
        }

        cout << "Maximum side of the square: " << max_side << endl;
}
```