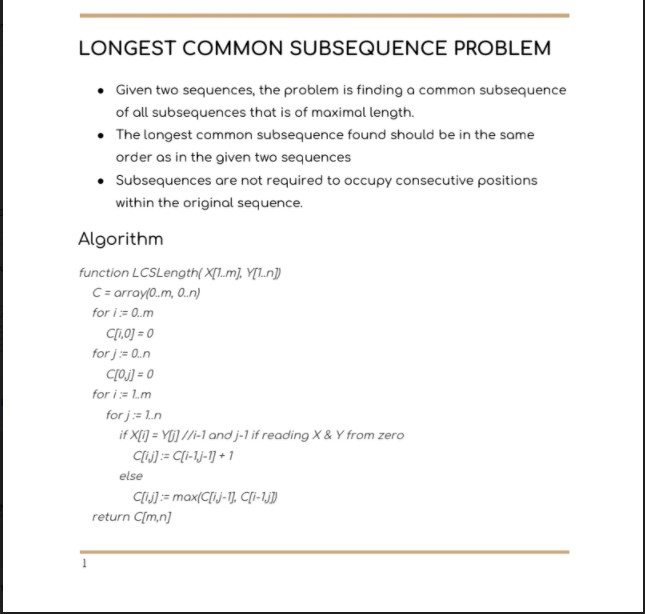
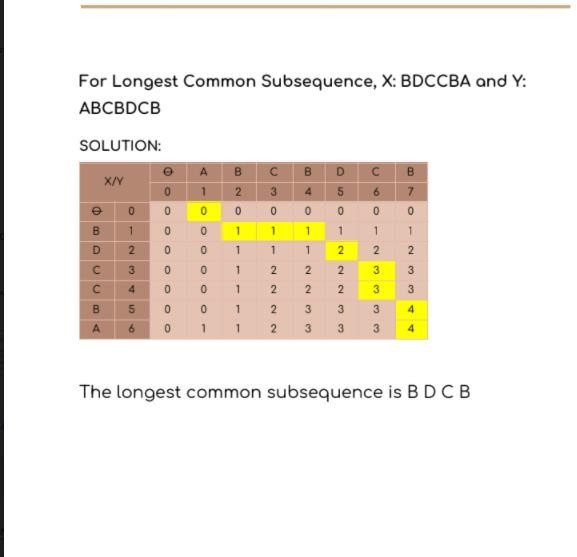
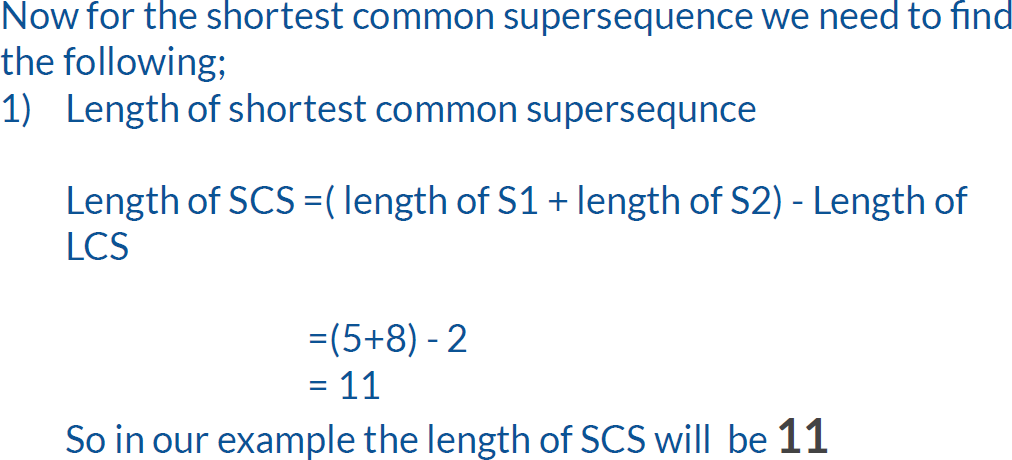
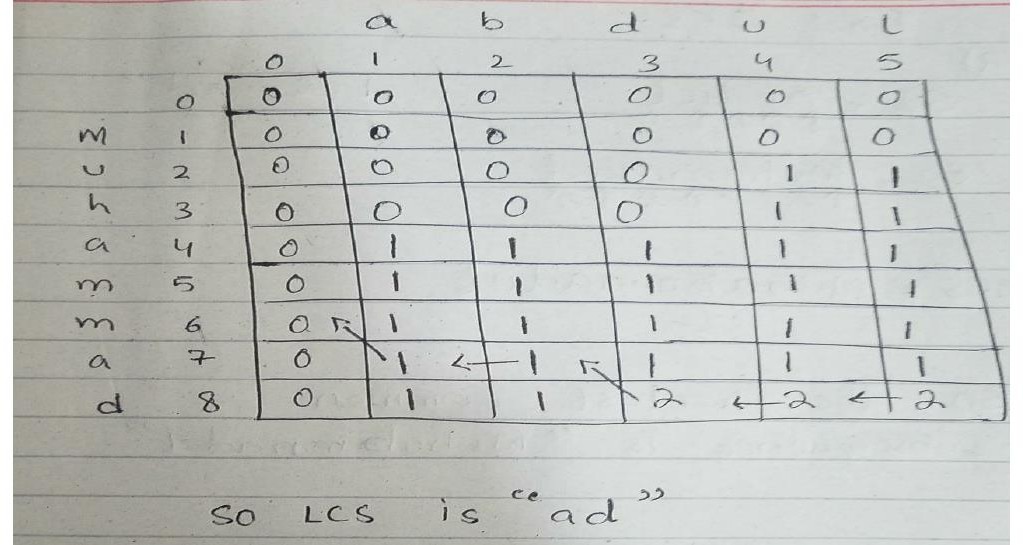
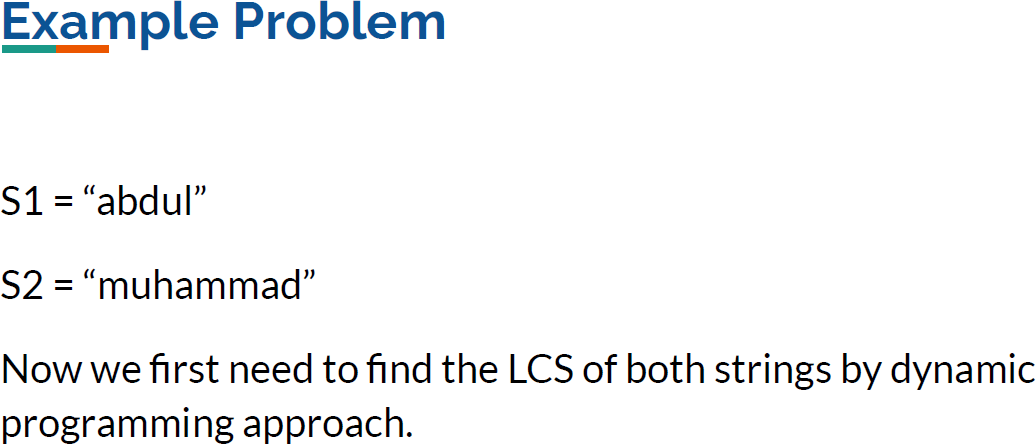
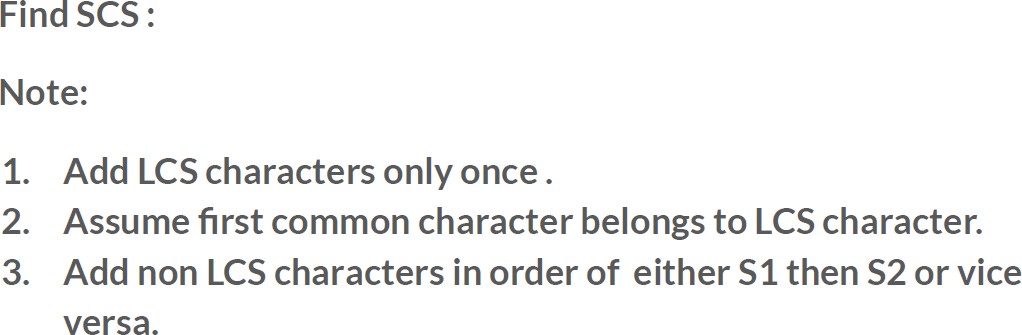
1. Longest Common Subsequence

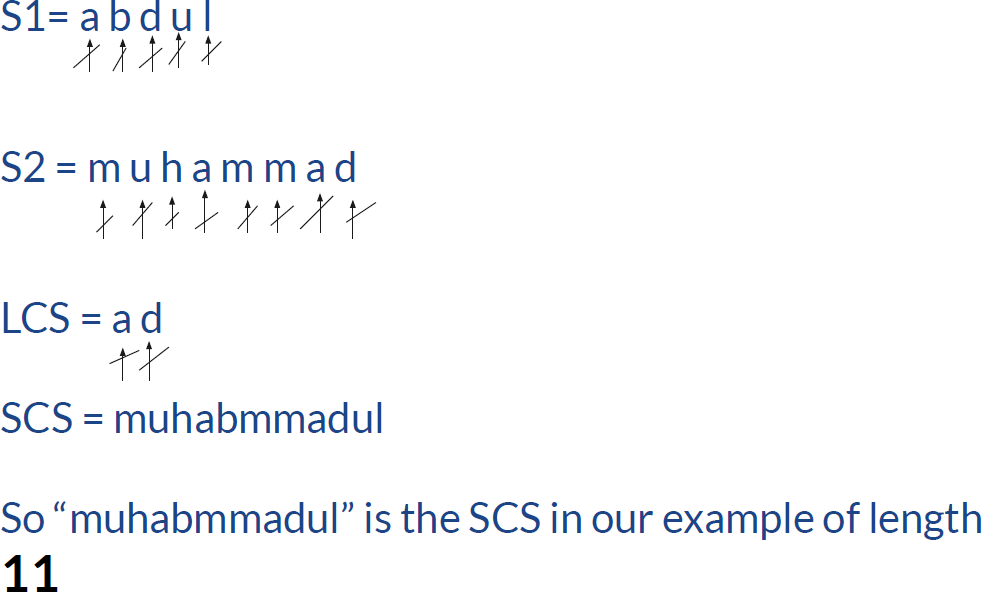


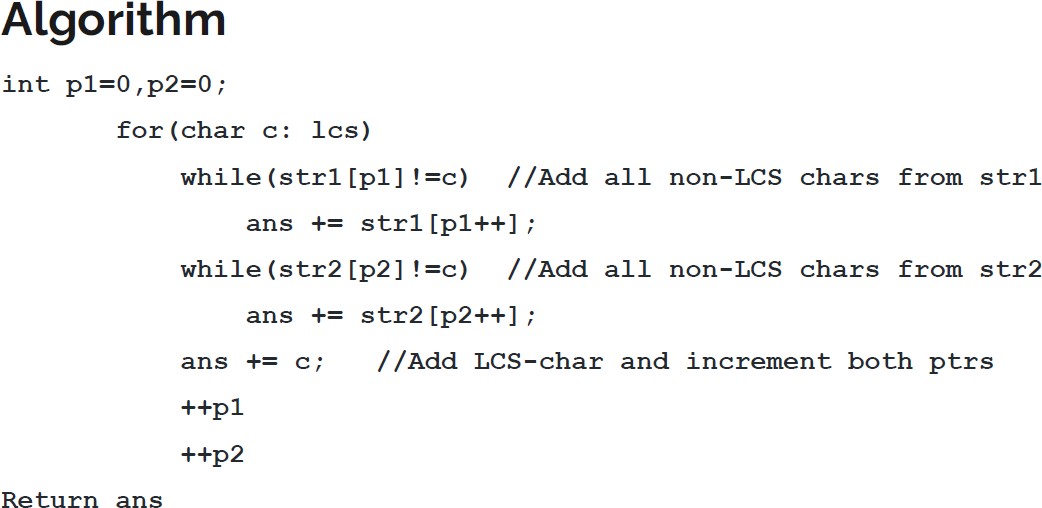


1. Shortest common sub sequence

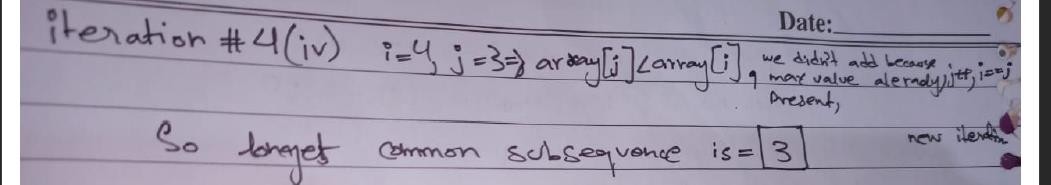
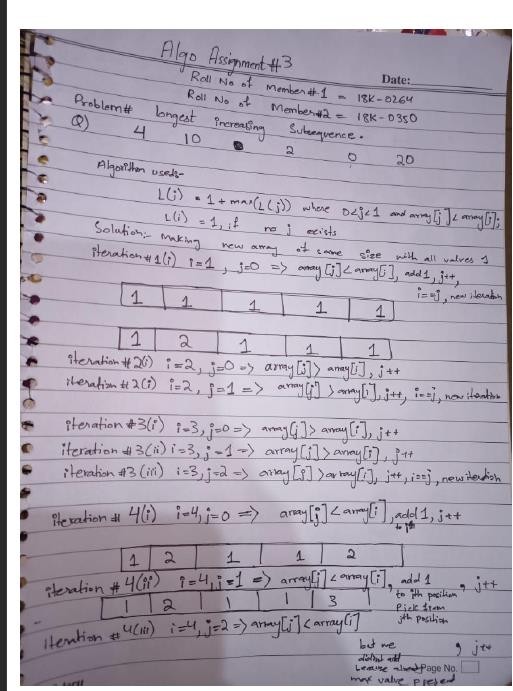








# Longest Increasing subsequence



1. Levenshtein-Distance

The minimum edit distance between two strings is the minimum number of editing operations:

* Insertion
* Deletion
* Substitution

Needed to transform one into the other.

Examples include Spell correction, computational biology etc.

Normally, each operation has cost of 1. However, in Levenshtein, the substitution cost can be taken as 2.

For two strings X of length n Y of length m. We define D(i,j) the edit distance between X[1..i] and Y[1..j] i.e., the first i characters of X and the first j characters of Y. The edit distance between X and Y is thus D(n,m).

# Algorithm

//Initialization D(i,0) = i

D(0,j) = j

//Recurrence Relation:

For each i = 1…n

For each j = 1…m

D(i,j)= min(

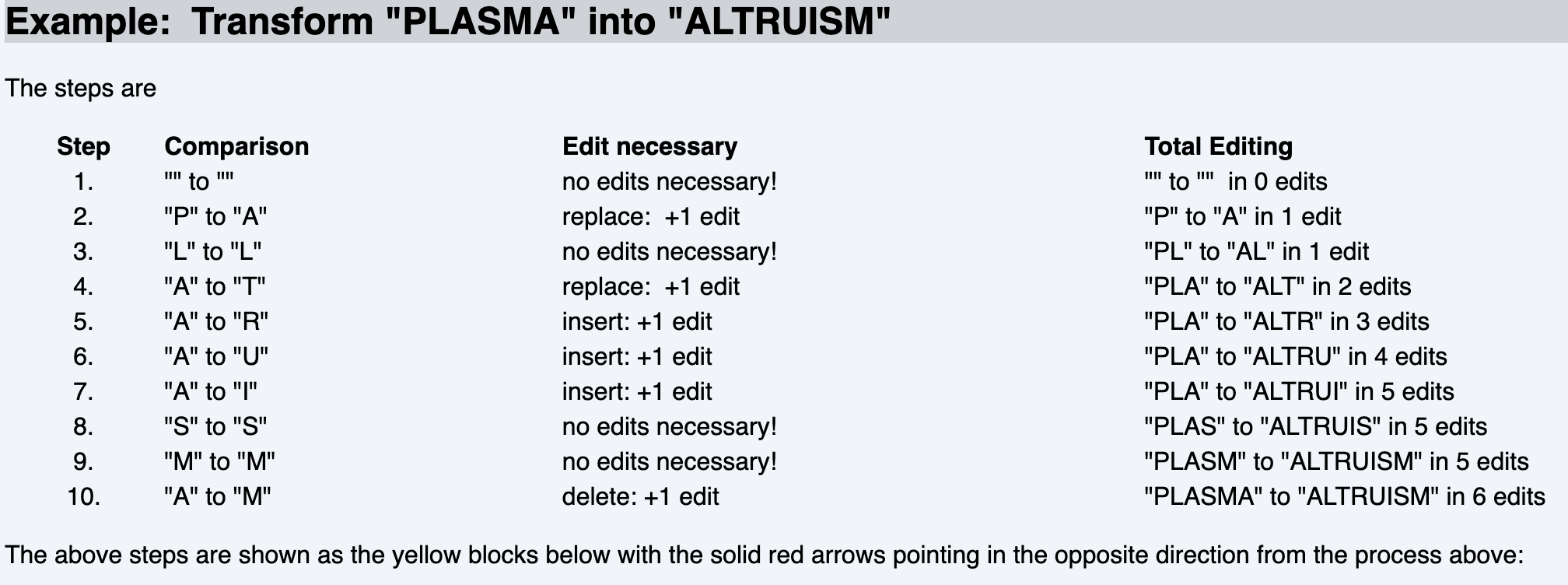
D(i-1, j) + 1,

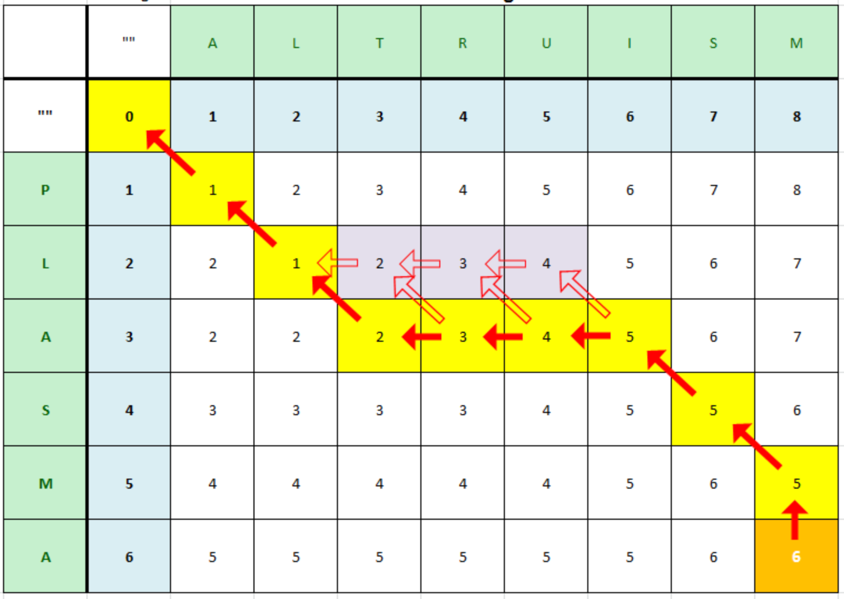
D(i, j-1) + 1,

D(i-1, j-1) + 2, if X(i) ≠ Y(j)

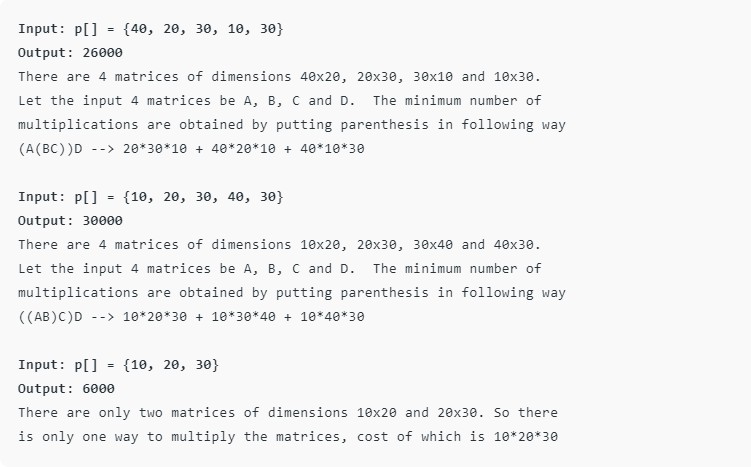
D(i-1, j-1) + 0 if X(i) = Y(j) )

//Termination: D(n,m) is distance





E) Matrix Chain Multiplication



#include <bits/stdc++.h> using namespace std;

// Matrix Ai has dimension p[i-1] x p[i]

// for i = 1..n

int MatrixChainOrder(int p[], int i, int j)

{

if (i == j)

return 0; int k;

int min = INT\_MAX; int count;

// place parenthesis at different places

// between first and last matrix, recursively

// calculate count of multiplications for

// each parenthesis placement and return the

// minimum count

for (k = i; k < j; k++)

{

count = MatrixChainOrder(p, i, k)

+ MatrixChainOrder(p, k + 1, j)

+ p[i - 1] \* p[k] \* p[j];

if (count < min) min = count;

}

// Return minimum count return min;

}

// Driver Code int main()

{

int arr[] = { 1, 2, 3, 4, 3 };

int n = sizeof(arr) / sizeof(arr[0]);

cout << "Minimum number of multiplications is "

<< MatrixChainOrder(arr, 1, n - 1);

}

1. Knapsack Problem

Given a set ‘S’ of ‘n’ items,

such that each item i has a positive value vi and positive weight wi

The goal is to find maximum-benefit subset that does not exceed the given weight W.

# Algorithm

Knapsack(j,w)

for i ← 0 to n

M[i,0] ← 0

for w ← 0 to W

M[0,w] ← 0

for j ← 1 to n

for w ← 0 to W

if wj > w

M[j,w] = M[j – 1, w))

else M[j,w] ← MAX(vj + M[j – 1, w – wj],

M[j – 1, w])

return M[n,W]

# Problem Question:

# 

The maximum benefit/value = 11

The selected items’ indexes are: si = [1,0,0,1,0]

1. Partition Problem

Given a set of positive integers, check if it can be divided into two subsets with equal sum.

First. The sum of all elements in the set is calculated. If sum is odd, we can’t divide the array into two sets. If sum is even, check if a subset with sum/2 exists or not.

For given example in the question:

Case 1:

S = {1, 8, 13, 23, 1, 17}.

The sum is 63, we can’t divide the array into two sets. Case 2: (last number of Set is modified)

Let S = {1, 8, 13, 23, 1, 18}, the sum is 64, now we can check if a subset with sum/2 exists or not. The result showed that set can be partitioned into S1 = {1, 8, 23}, and S2 = {13, 1, 18}, with each subset having a sum of 32.

Case 3: (3rd number of Set is modified)

Let S = {1, 8, 13, 23, 1, 16}, the sum is 62, now we can check if a subset with sum/2 exists or not. The result showed that set can be partitioned into S1 = {8, 23}, and S2 = {1,13, 1, 16}, with each subset having a sum of 31. However, note that the size of sets is not equal.

Case 3: (3rd number of Set is modified)

Let S = {1, 8, 13, 32, 1, 17}, the sum is 82, now we can check if a subset with sum/2 exists or not. The result showed that set cannot be partitioned.

For code program algorithm, refer to the website referred in question.

1. Rod Cutting Problem

Rod cutting problem is a type of allocation problem. Allocation problem involves the distribution of resources among the competing alternatives in order to minimize the total costs or maximize total return (profit).

In the rod cutting problem we have given a rod of length n, and an array that contains the prices of all the pieces smaller than n, determine the maximum profit you could obtain from cutting up the rod and selling its pieces.

Length[] = {1,2,3,4,5,6,7,8}

price[] = {1,5,8,9,10,16,18,20}

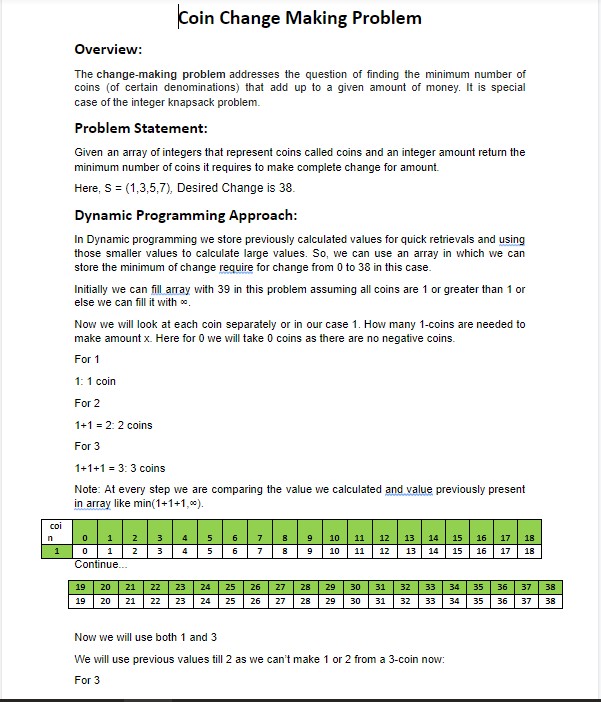
Rod Length: 8

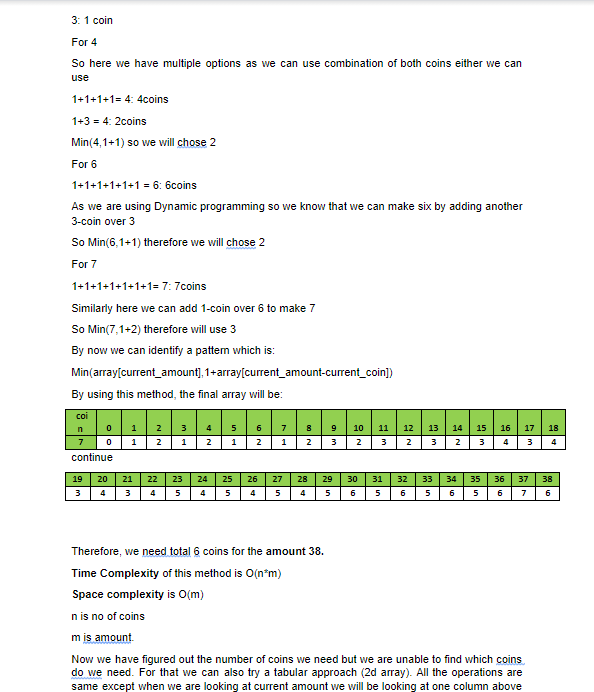
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Price  (i) | Length  (i) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 5 | 2 | 0 | 1 | 5 | 6 | 10 | 11 | 15 | 16 | 20 |
| 8 | 3 | 0 | 1 | 5 | 8 | 10 | 13 | 16 | 17 | 21 |
| 6 | 4 | 0 | 1 | 5 | 8 | 10 | 13 | 16 | 17 | 21 |
| 10 | 5 | 0 | 1 | 5 | 8 | 10 | 13 | 16 | 17 | 21 |
| 16 | 6 | 0 | 1 | 5 | 8 | 10 | 13 | 17 | 17 | 21 |
| 18 | 7 | 0 | 1 | 5 | 8 | 10 | 13 | 17 | 18 | 19 |
| 20 | 8 | 0 | 1 | 5 | 8 | 10 | 13 | 17 | 18 | 20 |

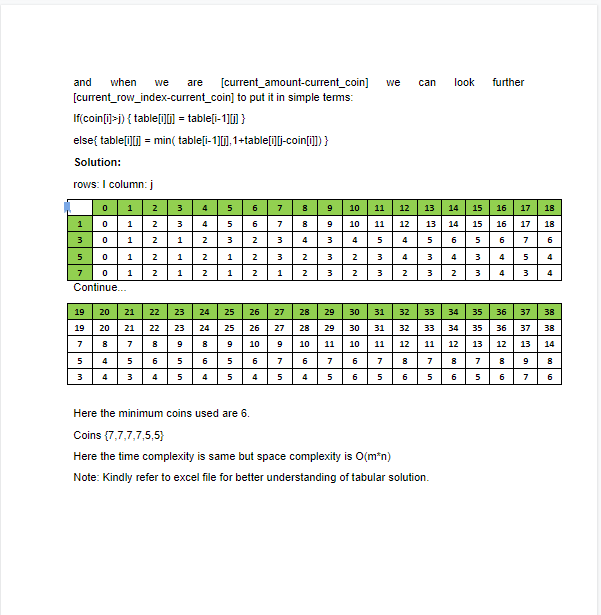
Maximum Profit = 20

Selected Pieces = 8 pieces of Rod Length 2, to get maximum profit for rod length 20.

1. Coin Change Making Problem







1. Word Break

