



**NORTHERN UNIVERSITY**  
**B A N G L A D E S H**

### **Assignment on**

**1. 0/1 Knapsack:** **Input :** Knapsack Capacity: 10 ; Item Weights: [2, 3, 4, 5] ; Item Values: [3, 4, 5, 6]

**Output :** Maximum Value: 13.

**2. Longest Common Subsequence (LCS):** **Input :** String 1: "ABCDGH" ; String 2 :  
"AEDFHR"

**Output :** Length of LCS: 3.

**3. Edit Distance:** **Input :** String 1: "kitten" ; String 2: "sitting" **Output :** Minimum Edit Distance: 3.

**4. Huffman Coding:** **Input :** Character Frequencies: {'a': 5, 'b': 9, 'c': 12, 'd': 13, 'e': 16, 'f': 45}

**Output:** Huffman Codes: {'a': '1100', 'b': '1101', 'c': '100', 'd': '101', 'e': '11', 'f': '0'}

### **Course Title: Algorithms Design & Analysis**

**Submitted to:**

**Md Mahadi Hasan**

Lecturer, Department of CSE

Northern University Bangladesh

**Submitted by:**

**Name: Urmi Binte Shahid**

**ID: 41220300507**

**Section : 4B**

### 1. 0/1 Knapsack:

```
#include<iostream> using  
namespace std;
```

```
int main() {    int item;    cout << "Enter  
the number of items: ";    cin >> item;
```

```
    int value[item];  
    int weight[item];
```

```
    cout << "Enter the value and weight for each item:" << endl;  
    for(int i = 0; i < item; i++) {        cin >> value[i] >> weight[i];  
    }
```

```
    int capacity;    cout << "Enter the  
knapsack capacity: ";    cin >> capacity;
```

```
    int dp[item + 1][capacity + 1];  
  
    for(int i = 0; i <= item; i++) {  
    for(int j = 0; j <= capacity; j++) {  
    if (i == 0 || j == 0) {                dp[i][j]  
= 0;  
        }                else if (weight[i  
- 1] > j) {                dp[i][j] =  
dp[i - 1][j];
```

```

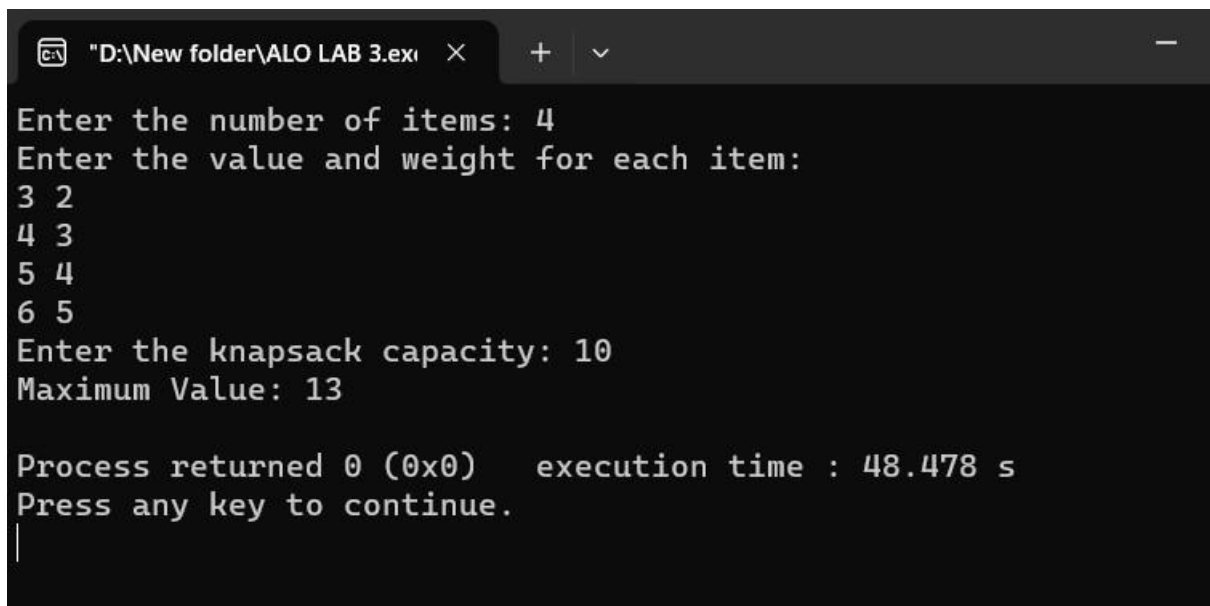
        } else {
            dp[i][j] = max(value[i - 1] + dp[i - 1][j -
weight[i - 1]], dp[i - 1][j]);
        }
    }
}

cout << "Maximum Value: " << dp[item][capacity] << endl;

return 0;
}

```

### Output:



The screenshot shows a Windows command prompt window with the title bar "D:\New folder\ALO LAB 3.exe". The program prompts the user for the number of items (4), then for the value and weight of each item (3 2, 4 3, 5 4, 6 5), and finally for the knapsack capacity (10). The program outputs the maximum value (13) and the execution time (48.478 s). The prompt "Press any key to continue." is shown at the bottom with a cursor.

```

Enter the number of items: 4
Enter the value and weight for each item:
3 2
4 3
5 4
6 5
Enter the knapsack capacity: 10
Maximum Value: 13

Process returned 0 (0x0)   execution time : 48.478 s
Press any key to continue.
|

```

## 2. Longest Common Subsequence (LCS):

```
#include<iostream> using
namespace std; const int
MAX = 500;

int maxDistance(const string & word1, const string & word2)
{   int m =
word1.length();   int n =
word2.length();   int
dp[MAX][MAX];

    for (int i = 0; i <= m; ++i) {
for (int j = 0; j <= n; ++j) {
if (i == 0 || j == 0) {
dp[i][j] = 0;

        } else if (word1[i - 1] == word2[j - 1]) {
dp[i][j] = dp[i - 1][j - 1] + 1;

        } else {          dp[i][j] = max (dp[i -
1][j], dp[i][j - 1]);

        }

    }

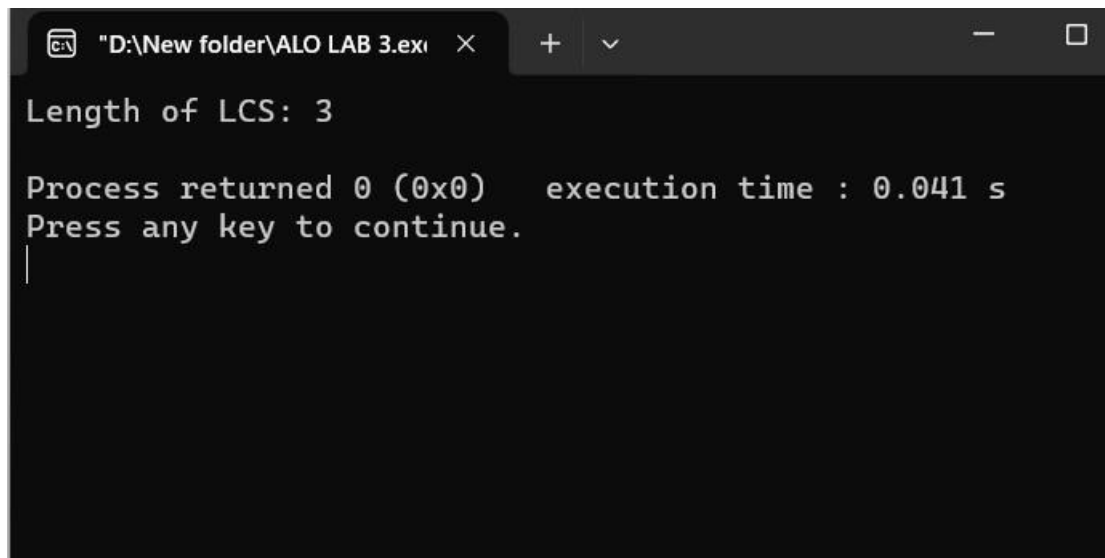
    }

    return dp[m][n];
}

int main() {   string word1 =
"ABCDGH";   string word2 =
"AEDFHR";
```

```
int distance = maxDistance(word1, word2);  
cout << "Length of LCS: " << distance << endl;  
  
return 0;  
}
```

### Output:



The screenshot shows a Windows command prompt window with a dark background. The title bar at the top indicates the file path "D:\New folder\ALO LAB 3.exe" and includes standard window controls (minimize, maximize, close). The command prompt displays the following text: "Length of LCS: 3", "Process returned 0 (0x0) execution time : 0.041 s", and "Press any key to continue." followed by a vertical cursor line.

```
"D:\New folder\ALO LAB 3.exe" X + v - □  
Length of LCS: 3  
Process returned 0 (0x0) execution time : 0.041 s  
Press any key to continue.  
|
```

### 3. Edit Distance:

```
#include<iostream> using
```

```
namespace std; const int
```

```
MAX = 500;
```

```
int minDistance(const string & word1, const string & word2)
```

```
{    int m =
```

```
word1.length();    int n =
```

```
word2.length();    int
```

```
dp[MAX][MAX];
```

```
    for (int i = 0; i <= m; ++i) {
```

```
    for (int j = 0; j <= n; ++j) {
```

```
    if (i == 0 || j == 0) {
```

```
    dp[i][j] = 0;
```

```
        } else if (word1[i - 1] == word2[j - 1]) {
```

```
    dp[i][j] = dp[i - 1][j - 1] + 0;
```

```
        } else {                dp[i][j] = min(min(dp[i - 1][j] + 1, dp[i][j - 1] + 1), dp[i
```

```
- 1][j - 1] + 2);
```

```
        }
```

```
    }
```

```
}
```

```
    return dp[m][n];
```

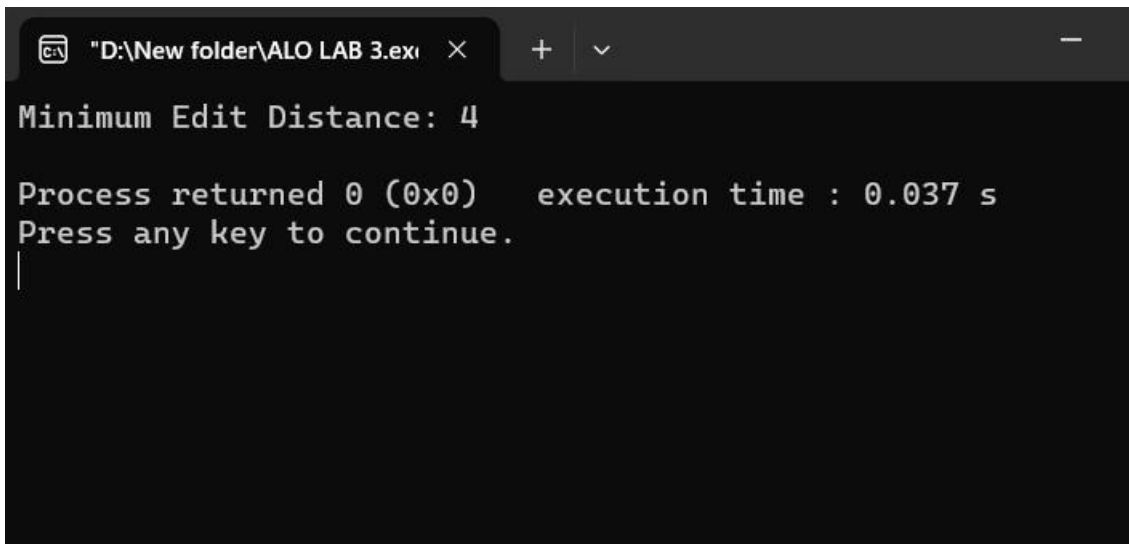
```
}
```

```
int main() {    string word1
= "kitten";    string word2
= "sitting";

    int distance = minDistance(word1, word2);
    cout << "Minimum Edit Distance: " << distance << endl;

    return 0;
}
```

### Output:



```
"D:\New folder\ALO LAB 3.exe"
Minimum Edit Distance: 4
Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
|
```

#### 4. Huffman Coding:

```
#include <iostream>
```

```
#include <queue>
```

```
#include <map>
```

```
#include <vector>
```

```
using namespace std;
```

```
struct HuffmanNode {
```

```
    char data;
```

```
    int freq;
```

```
    HuffmanNode *left, *right;
```

```
    HuffmanNode(char data, int freq) : data(data), freq(freq), left(nullptr),  
    right(nullptr) {}
```

```
};
```

```
struct CompareNodes {    bool operator()(HuffmanNode*
```

```
lhs, HuffmanNode* rhs) {    return lhs->freq > rhs->freq;
```

```
    }
```

```
};
```

```
void generateHuffmanCodes(map<char, string>& huffmanCodes,  
HuffmanNode* root, string code) {
```

```
    if (root == nullptr)    return;
```



```

    if (root->data != '\0') {        huffmanCodes[root->data] = code;
    }
    generateHuffmanCodes(huffmanCodes, root->left, code + "0");
    generateHuffmanCodes(huffmanCodes, root->right, code + "1");
}

```

```

map<char, string> buildHuffmanTree(map<char, int>& charFrequencies) {
    priority_queue<HuffmanNode*, vector<HuffmanNode*>, CompareNodes>
    minHeap;

```

```

    for (auto& entry : charFrequencies) {        minHeap.push(new
    HuffmanNode(entry.first, entry.second));
    }

```

```

    while (minHeap.size() > 1) {
        HuffmanNode* left = minHeap.top();
    minHeap.pop();

```

```

        HuffmanNode* right = minHeap.top();
    minHeap.pop();

```

```

        HuffmanNode* newNode = new HuffmanNode('\0', left->freq + right->freq);
    newNode->left = left;        newNode->right = right;

```

```

        minHeap.push(newNode);
    }

```

```

    HuffmanNode* root = minHeap.top();

```

```

        map<char,                string>                huffmanCodes;
generateHuffmanCodes(huffmanCodes, root, "");    return huffmanCodes;

}

int main() {

    map<char, int> charFrequencies = {{'a', 5}, {'b', 9}, {'c', 12}, {'d', 13}, {'e', 16},
{'f', 45}};

    map<char, string> huffmanCodes = buildHuffmanTree(charFrequencies);

    cout << "Huffman Codes:" << endl;    for (auto&
entry : huffmanCodes) {        cout << entry.first << ": "
<< entry.second << endl;
    }

    return 0;
}

```

**Output:**

"D:\New folder\ALO LAB 3.exe" X

+

▼

—

□

Huffman Codes:

a: 1100

b: 1101

c: 100

d: 101

e: 111

f: 0

Process returned 0 (0x0) execution time : 0.043 s

Press any key to continue.

|