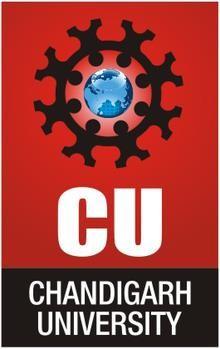
**CHANDIGARH UNIVERSITY**

UNIVERSITY INSTITUTE OF ENGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By:                                                                          Submitted To:**  Yash Gupta ER. Monika(E12802) | |
| **Subject Name** | Design Analysis and Algorithm |
| **Subject Code** | 20CSP\_312 |
| **Branch** | CSE |
| **Semester** | 5th |

**LAB -INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Program** | **Date** | **Evaluation** | | | | **Sign** |
| **LW(12)** | **VV(8)** | **FW(10)** | **Total (30)** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Experiment 5**

**1. Aim/Overview of the practical:**

Code and analyse to find an optimal solution to matrix chain multiplication using dynamic programming.

**2. Algorithm:**

* 1. Build a matrix dp[][] of size N\*N for memoization purposes.
  2. Use the same recursive call as done in the above approach:
  3. When we find a range (i, j) for which the value is already calculated, return the minimum value for that range (i.e., dp[i][j]).
  4. Otherwise, perform the recursive calls as mentioned earlier.
  5. The value stored at dp[0][N-1] is the required answer.

**3. Steps for experiment/practical/Code:**

#include <limits.h>

#include <stdio.h>

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

{

if (i == j)

return 0;

int k;

int min = INT\_MAX;

int 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 min;

}

int main()

{

int arr[] = { 10, 12, 30, 40, 13 };

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

printf("Minimum number of multiplications is %d ",

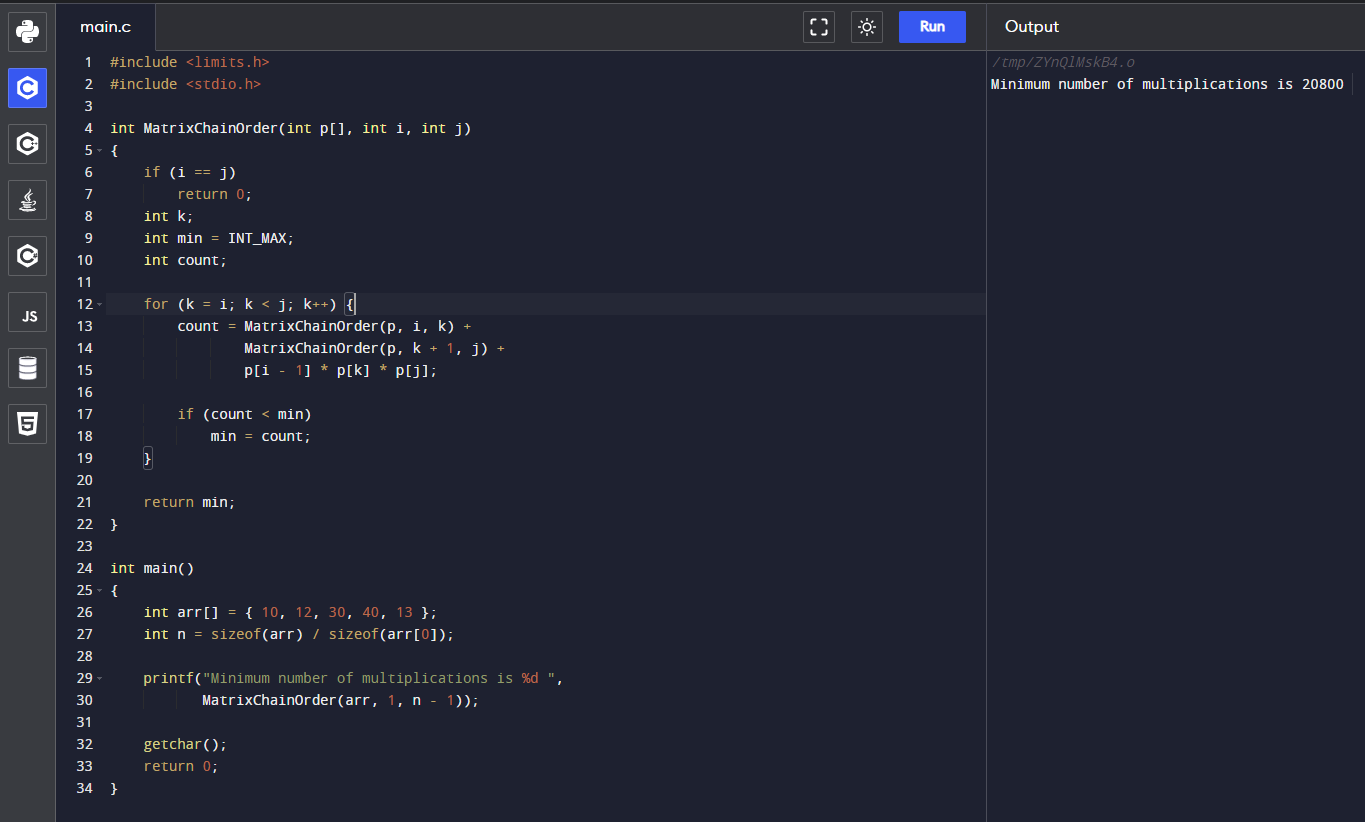
MatrixChainOrder(arr, 1, n - 1));

getchar();

return 0;

}

}**4. Result/Output/Writing Summary:**



**5. Observations/Discussions/ Complexity Analysis:**

Time complexity of finding frequency of elements of an array is O(n^3).

**6. Learning outcomes (What I have learnt):**

**1.** To learn how to calculate the frequency of the elements of an array.

**2.** To learn how to use for loop in these cases.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |