Name: Ashutosh kumar

Reg. Num: 231070006

Batch: A

DAA Lab Experiment 4

Aim:

- 1. To find inversion count of course choice of students.
- 2. To multiply two integers using brute force and divide-and-conquer methods

Program:

1. Inversion count:

Brute-force approach:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

using namespace std;

// Function to read course choices from CSV file
vector<vector<int>> readCourseChoicesFromCSV(const string& filename) {
    ifstream file(filename);
    vector<vector<int>> courseChoices;
    string line, value;

// Skip the header line
```

```
getline(file, line);
    while (getline(file, line)) {
        vector<int> studentChoices;
        stringstream ss(line);
        getline(ss, value, ',');
        while (getline(ss, value, ',')) {
            studentChoices.push_back(stoi(value)); // Convert string to
        courseChoices.push_back(studentChoices);
    file.close();
    return courseChoices;
int main() {
    string filename = "students_course_choices.csv";
    vector<vector<int>> courseChoices = readCourseChoicesFromCSV(filename);
    for (int i = 0; i < courseChoices.size(); i++) {</pre>
        cout << "Student " << i + 1 << " course choices: ";</pre>
        for (int j = 0; j < courseChoices[i].size(); j++) {</pre>
            cout << courseChoices[i][j] << " ";</pre>
        cout << endl;</pre>
    return 0;
```

Efficient Approach (Merge Sort):

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
using namespace std;
int mergeAndCount(vector<int>& arr, int left, int mid, int right) {
    int count = 0;
    vector<int> temp(right - left + 1);
    int i = left, j = mid + 1, k = 0;
    while (i <= mid && j <= right) {
        if (arr[i] <= arr[j]) {
            temp[k++] = arr[i++];
        } else {
            temp[k++] = arr[j++];
            count += (mid - i + 1);
        }
    while (i <= mid) {</pre>
        temp[k++] = arr[i++];
    while (j <= right) {</pre>
        temp[k++] = arr[j++];
    for (i = left, k = 0; i <= right; i++, k++) {
        arr[i] = temp[k];
    return count;
int mergeSortAndCount(vector<int>& arr, int left, int right) {
    if (left >= right) {
       return 0;
```

```
int mid = left + (right - left) / 2;
    int count = 0;
    count += mergeSortAndCount(arr, left, mid);
    count += mergeSortAndCount(arr, mid + 1, right);
    count += mergeAndCount(arr, left, mid, right);
    return count;
vector<vector<int>> readCourseChoicesFromCSV(const string& filename) {
    ifstream file(filename);
    vector<vector<int>> courseChoices;
    string line, value;
    getline(file, line);
    while (getline(file, line)) {
        vector<int> studentChoices;
        stringstream ss(line);
        getline(ss, value, ',');
        while (getline(ss, value, ',')) {
            studentChoices.push_back(stoi(value)); // Convert string to
        courseChoices.push_back(studentChoices);
    file.close();
    return courseChoices;
int main() {
    string filename = "students_course_choices.csv";
    vector<vector<int>>> courseChoices = readCourseChoicesFromCSV(filename);
```

```
// Process each student's course choices and calculate inversion count
for (int i = 0; i < courseChoices.size(); i++) {
    int inversionCount = mergeSortAndCount(courseChoices[i], 0,
courseChoices[i].size() - 1);
    cout << "Student " << i + 1 << " inversion count: " << inversionCount
<< endl;
    }
    return 0;
}</pre>
```

Output:

```
Student 1 inversion count: 7
Student 2 inversion count: 7
Student 3 inversion count: 6
Student 4 inversion count: 3
Student 5 inversion count: 10
Student 6 inversion count: 9
Student 7 inversion count: 6
Student 8 inversion count: 6
Student 9 inversion count: 8
Student 10 inversion count: 5
```

2. Integer Multiplication

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

using namespace std;

// Function to perform brute force multiplication
int bruteForceMultiply(int a, int b) {
    return a * b;
```

```
int multiplyDivideAndConquer(int x, int y) {
    if (x < 10 || y < 10) return x * y;
    int n = max(to_string(x).length(), to_string(y).length());
    int half = n / 2;
   int a = x / pow(10, half);
    int b = x % static_cast<int>(pow(10, half));
    int c = y / pow(10, half);
    int d = y % static cast<int>(pow(10, half));
   int ac = multiplyDivideAndConquer(a, c);
    int bd = multiplyDivideAndConquer(b, d);
    int ab_cd = multiplyDivideAndConquer(a + b, c + d);
   return ac * pow(10, 2 * half) + (ab_cd - ac - bd) * pow(10, half) + bd;
vector<pair<int, int>> readNumbersFromCSV(const string& filename) {
    ifstream file(filename);
   vector<pair<int, int>> numberPairs;
    string line, value;
   getline(file, line);
   while (getline(file, line)) {
        stringstream ss(line);
        string num1, num2;
        getline(ss, num1, ',');
        getline(ss, num2, ',');
        numberPairs.push_back(make_pair(stoi(num1), stoi(num2)));
    }
    file.close();
    return numberPairs;
```

```
int main() {
    // File containing pairs of numbers
    string filename = "numbers_for_multiplication.csv";

    // Read the number pairs from the CSV file
    vector<pair<int, int>> numberPairs = readNumbersFromCSV(filename);

    // Perform multiplication using brute force and divide-and-conquer methods
    for (int i = 0; i < numberPairs.size(); i++) {
        int a = numberPairs[i].first;
        int b = numberPairs[i].second;

        // Brute-force multiplication
        int bruteResult = bruteForceMultiply(a, b);

        // Divide-and-conquer multiplication
        int divAndConqResult = multiplyDivideAndConquer(a, b);

        cout << "Pair " << i + 1 << ": " << a << " * " << b << endl;
        cout << "Brute-force result: " << bruteResult << endl;
        cout << "Divide-and-conquer result: " << divAndConqResult << endl;
        cout << "Divide-and-conquer result: " << endl;
        cout << "Divide-and-conquer result: " << endl;
        cout << "content condition of the conditio
```

Output:

```
ashutosh kumar@Ashutosh-PC MINGW64 /e/programs/code forces/output
Enter first large integer: 1234
Enter second large integer: 4321
Product of the two large integers: 5332114

ashutosh kumar@Ashutosh-PC MINGW64 /e/programs/code forces/output
Enter first large integer: 8765
Enter second large integer: 0
Product of the two large integers: 00000000

ashutosh kumar@Ashutosh-PC MINGW64 /e/programs/code forces/output
Enter first large integer: -9876
Enter second large integer: 3456
Product of the two large integers: 1929451456

ashutosh kumar@Ashutosh-PC MINGW64 /e/programs/code forces/output
Enter first large integer: -1234
```

Enter second large integer: -1234
Product of the two large integers: 1029682756

ashutosh kumar@Ashutosh-PC MINGW64 /e/programs/code forces/output
Enter first large integer: 1234
Enter second large integer: 1234
Product of the two large integers: 1522756

Conclusion:

- 1. We have found the inversion count of course choice of students using divide and conquer method, which reduces the time complexity to $O(n\log n)$ from $O(n^2)$ of the brute force method. We classified the students according to their inversion count.
- 2. We have multiplied two integers using the divide and conquer method, which reduces the time complexity to O(nlogn) from O(n^2) of the brute force method. We reduced the number of recursive calls to the function, which increased the efficiency. This algorithm can be used to multiply numbers of large size efficiently.



fune CountBrut (arr): 11 Input: Asuray of number 11 Output: No. of inverted Value & index pairs int Court = 0 for (i:0 - len (arr)-1) for (j: i+1 → len (arr) -1) # arr [i] > arr [i]: int Count ++ return inv Count

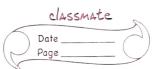
Time Complexity Since there one 2 Nested Poops So time Complexity is O(N2) which is not optimal.

Optimal Solution:

June merged Count (arr): I sutbut: Away of Numbers
1/outbut: Number of inversions of Value & index and
Sorted array. if (len (arr) < 2) retwin Oyn D

mid = length (arr)/2 I, left Inv = merge And Count (arr [o: mid])

M, Might Inv = merge And Count (arr [o: lenform)]) merged, Split Inv = merge (1, 21)
total Inv = left Inv + right Inv + Aplit Inv
yetwen merged, total Inv



Hunction to merge two merge two array
funct merge [left, right] {
17 Input: 2 arrays
Il Dutput : Merged array and Count of Invers Merged =[]
t=0
7=0
Split Inv=0
while i < len (left) & & j < len (right)
I left [i] z = Hight [j] E nerged append (left [i])
S moved abband (left [i])
1+10 }
merged · append [right [i])
j++ ?
, , , , , , , , , , , , , , , , , , ,
append remaining elements of left or right to
merged
J
return merged, Split Inv
9
San the contract of the san th
Is to Count no of inversion:
func Gunt Student Inv (arr):
Zero Inv =0
One Inv = 0
two Iny = 0
three Inv Fo

for Pach Source Code in arr

in rersion = Merge And Count Carr)

If inversions == 0;

ZeroInvt = 1

Else If inversions == 1!

Else If inversions == 2;

two Inv += 1

Else If inversion == 3

threeInv += 1.

Time Complexity using biggyback on merge Sort:

Divide Step: overay is recursively divided in hat until it reaches Subarray of Size 1.

D(n) = O(log n)

Conquer step: During Merge process, two sorted halres one Combined, there Mequires time Complexity of O(n)

: total time Complexity

T(n) = 2 T(n/2) + O(n)

Using Moster theorem a = 2, l = 2, d = 1 $a = b^d \quad as \quad 2 = 2^l$ T(n) = o(n log n)

Experia	nent	task	-2
func	m_{an}	ual P	Julti
func :	, 2	\sum_{n}	tegl

func manual Multiply (numl, num)

1 Input: 2 Integers with with digits in 2 arrays
11 Out put: product

for (i: len (Num) \rightarrow 0)

for (j: len (Num 2) -1 \rightarrow 0)

mul = run | [i] + num 2 [j]

Position | = itj

Position 2 = itj+1

Sum = mul + result [basition 2]

Yesult [Position 2] = Sum 1/10

resut [Pointon] t = Sum/10

time Complexity

O(n, x n.) where n, and n, wie the length

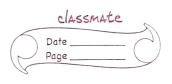
of the input avorage.

Optimal Solution

func Karatsuba (24)

[11 Input: 2 large integer os array of digits
11 Output: There product.

if (x < 10 or y < 10)return $x^{*}y$ m = max (len (x), len (y))holy = m/2high x, low x = div mod (x, 10, holy)high y, low y = div mod (y, 10, holy)



Zo = Karatsuba (low X, low Y)
Z1 = Karatsuba (low X + high X, low Y + high Y)
Z3 = Karatsuba (high X, high Y)

Veturn (22*10 (2*half) + (21-22-20) + 10 half + 30)

-lest Coses:

(2) Apr $2 = \{0\}$ Apr $2 = \{9,9,9,9,9,9\}$ Outbut = 0

3 Aur 1 = $\{9, 9, 9, 9, 9, 9\}$ Aur 2 = $\{1, 2, 3\}$ Output = 12299877

(4) Arr $l = \{9,9,8,7,6,5,4\}$ Arr $2 = \{0\}$ output = 0

(3) $Arr1 = \{ -8, 6, 5, 4 \}$ $Arr2 = \{ 6, 5, 4 \}$ Output = -5, 6, 59716