Assignment No: 01

Department of Computer Science Iqra University Islamabad

Data Structures and Algorithms

Maqsood Ahmed

ID: 38186

Problem # 1: [CLO2]

Write an algorithm to remove duplicate elements from an unsorted array of integers, keeping only the first occurrence of each unique value. You should not use extra data structures (e.g.,hash sets) for this task.

Time Complexity: Calculate and explain the time complexity of your algorithm.

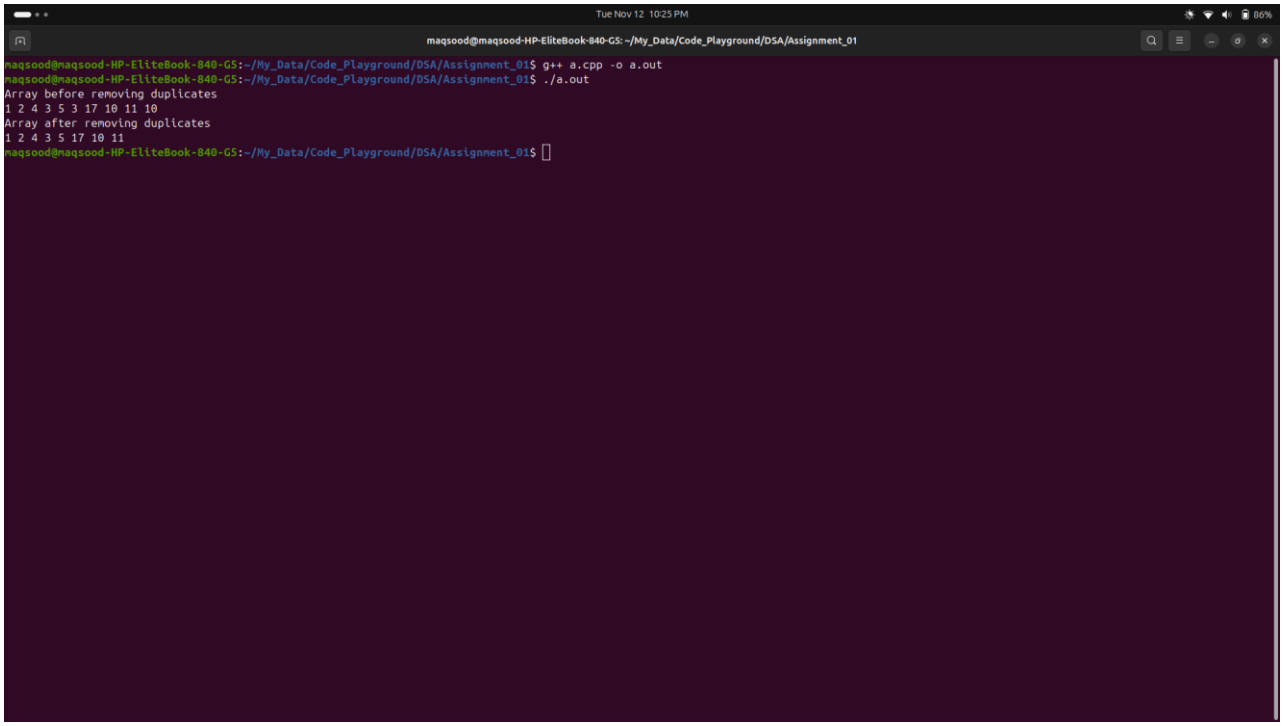
Source Code:

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

int removeDuplicates(int arr[], int &size) {
   int cnt = 0;
   for(int i = 0; i < size; i++) {
      bool flag = true;
      for(int j = 0; j < cnt; j++) {
         if(arr[i] == arr[j]) {
            flag = false;
            break;
         }
      }

      if(flag) {
         arr[cnt++] = arr[i];
      }
   }

   size = cnt;  // update the size
   return cnt;
}

int main(void) {
   int size = 10;
   int arr[size] = {1, 2, 4, 3, 5, 3, 1, 10, 11, 10};
   removeDuplicates(arr, size);

   for(int i = 0; i < size; i++) {
      cout << arr[i] << ' ';
   }

   return EXIT_SUCCESS;
}
```

Time Complexity:

The time complexity of this algorithm is **$0(n^2)$**.

OUTPUT:


Tue Nov 12 10:25 PM
86%
maqsood@maqsood-HP-EliteBook-840-G5: ~/My_Data/Code_Playground/DSA/Assignment_01
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ g++ a.cpp -o a.out
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ ./a.out
Array before removing duplicates
1 2 4 3 5 3 17 10 11 10
Array after removing duplicates
1 2 4 3 5 17 10 11
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$

Problem # 2: [CLO2]

You are given a list of functions. For each function, identify its asymptotic time complexity in Big-O notation:

1. $n^{3/2}$
2. $8 \log_2 n$
3. $2 \cdot 3^n$
4. $3 \cdot n^2$
5. $3 \cdot 2^n$
6. $3 \cdot n^2 + 2 \cdot n^3$
7. $n \log_2 7$
8. $\log_2 n^n$

Order the functions from smallest to largest complexity and provide brief explanations.

Solution:

1. $n^{3/2}$

   Complexity: $O(n^{3/2})$

2. $8 \log_2 n$

   Complexity: O(log n)

3. $2 . 3^n$

   Complexity: $O(3^n)$

4. $3 . n^2$

   Complexity: $O(n^2)$

5. $3 . 2^n$

   Complexity: $O(2^n)$

6. $3 . n^2 + 2 . n^3$

   Complexity: $O(n^3)$

7. $n \log_2 7$

Complexity: O(n)

8. $\log_2 n^n$

Complexity: $O(n \log_n)$

Order from Smallest to Largest Complexity

1. **$8 \log_2 n$ - $O(\log n)$**
2. **$n \log_2 7$ - $O(n)$**
3. **$\log_2 n^n$ - $O(n \log_n)$**
4. **$n^{3/2}$ - $O(n^{3/2})$**
5. **$3 . n^2$ - $O(n^2)$**
6. **$3 . n^2 + 2 . n^3$ - $O(n^3)$**
7. **$3 . 2^n$ - $O(2^n)$**
8. **$2 . 3^n$ - $O(3^n)$**

Problem # 3: [CLO2]

Implement a C++ class ArrayUtility that provides static methods for one-dimensional arrays.

The following methods should be included:

1. **static int findMax(int[] A, int i, int j)**
2. **static int findMaxPos(int[] A, int i, int j)**
3. **static int findMin(int[] A, int i, int j)**
4. **static int findMinPos(int[] A, int i, int j)**
5. **static void swap(int[] A, int i, int j)**

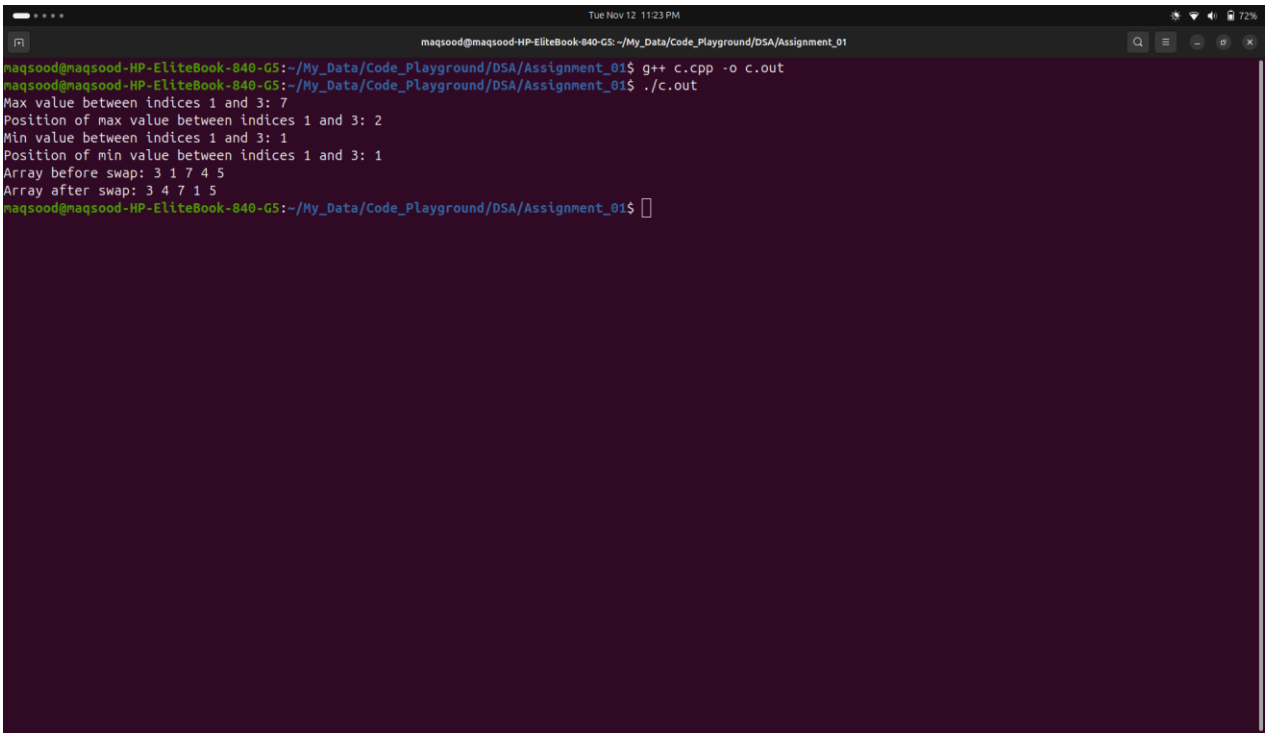
Source Code:

```
#include <iostream>
#include <climits>
using namespace std;
class ArrayUtility {
public:
   static int findMax(int A[], int i, int j) {
      int maxVal = INT_MIN;
      for (int k = i; k <= j; ++k) {
         if (A[k] > maxVal) {
            maxVal = A[k];
         }
      }
      return maxVal;
   }
   static int findMaxPos(int A[], int i, int j) {
      int maxVal = INT_MIN;
      int maxPos = -1;
      for (int k = i; k <= j; ++k) {
         if (A[k] > maxVal) {
            maxVal = A[k];
            maxPos = k;
```

```
            }
        }
        return maxPos;
    }
    static int findMin(int A[], int i, int j) {
        int minVal = INT_MAX;
        for (int k = i; k <= j; ++k) {
            if (A[k] < minVal) {
                minVal = A[k];
            }
        }
        return minVal;
    }
    static int findMinPos(int A[], int i, int j) {
        int minVal = INT_MAX;
        int minPos = -1;
        for (int k = i; k <= j; ++k) {
            if (A[k] < minVal) {
                minVal = A[k];
                minPos = k;
            }
        }
        return minPos;
    }
    static void swap(int A[], int i, int j) {
        int temp = A[i];
        A[i] = A[j];
        A[j] = temp;
    }
};
```

```
int main() {
   int arr[] = {3, 1, 7, 4, 5};
   int n = sizeof(arr) / sizeof(arr[0]);
   cout << "Max value between indices 1 and 3: " << ArrayUtility::findMax(arr, 1, 3) << endl;
   cout << "Position of max value between indices 1 and 3: " << ArrayUtility::findMaxPos(arr, 1, 3) << endl;
   cout << "Min value between indices 1 and 3: " << ArrayUtility::findMin(arr, 1, 3) << endl;
   cout << "Position of min value between indices 1 and 3: " << ArrayUtility::findMinPos(arr, 1, 3) << endl;
   cout << "Array before swap: ";
   for (int i = 0; i < n; ++i) cout << arr[i] << " ";
   cout << endl;
   ArrayUtility::swap(arr, 1, 3);
   cout << "Array after swap: ";
   for (int i = 0; i < n; ++i) cout << arr[i] << " ";
   cout << endl;
   return EXIT_SUCCESS;
}
```

OUTPUT:


Tue Nov 12 11:23 PM
72%
maqsood@maqsood-HP-EliteBook-840-G5: ~/My_Data/Code_Playground/DSA/Assignment_01
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ g++ c.cpp -o c.out
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ ./c.out
Max value between indices 1 and 3: 7
Position of max value between indices 1 and 3: 2
Min value between indices 1 and 3: 1
Position of min value between indices 1 and 3: 1
Array before swap: 3 1 7 4 5
Array after swap: 3 4 7 1 5
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$

Problem # 4: [CLO2]

Design and implement a simple Inventory Management System using arrays. Each product has a unique ID and a price. Implement the following functionalities:

1. Add a product by its ID and price (ensure uniqueness).

2. Remove a product by its ID.

3. Sort products by price using a sorting algorithm.

4. Search for a product by its ID using binary search (after sorting).

5. Display the inventory.

Source Code:

```
#include <iostream>
using namespace std;
class Product {
public:
   int unique_id;
   int price;
   Product() {
      unique_id = -1;
      price = 0;
   }
};


class Inventory {
public:
   Product products[100];
   int productCnt;
   // default constructor, it initialize the productCnt to zero
   Inventory() : productCnt(0) {}
   void addProduct(int id, int price) {
      if (hasUniqueId(id)) {
         products[productCnt].unique_id = id;
```

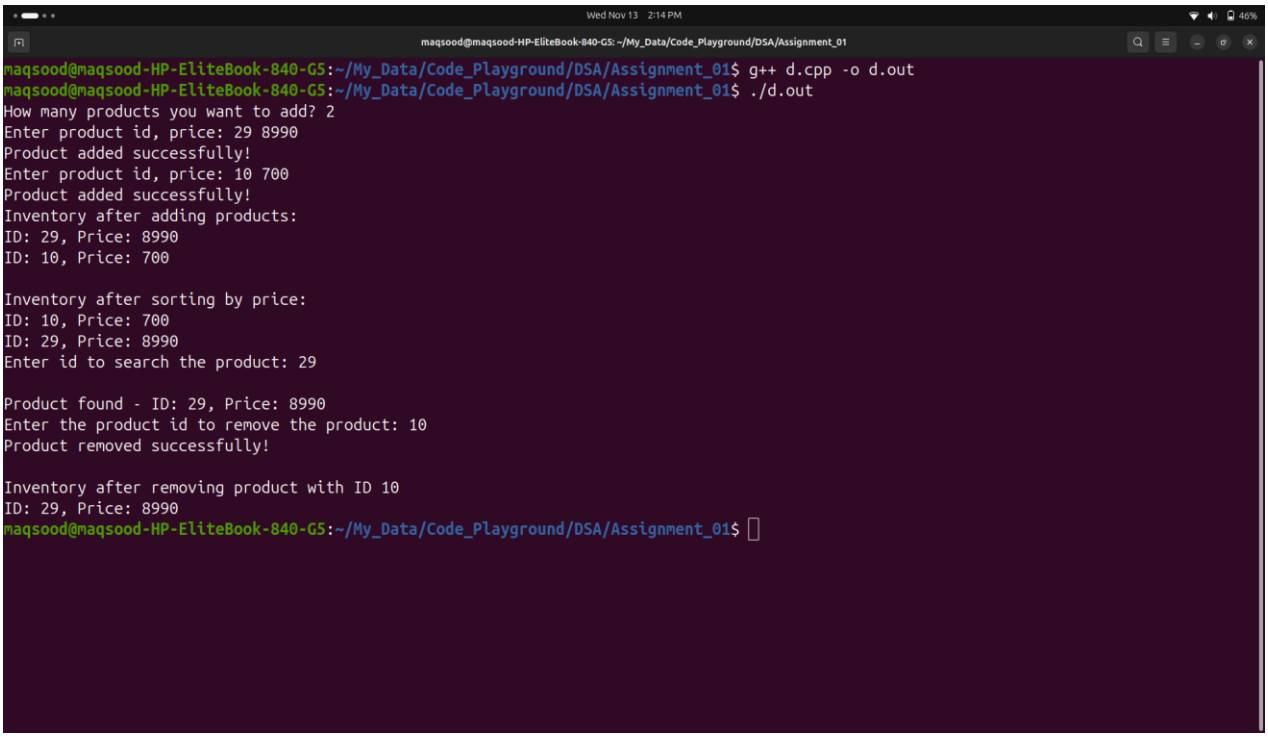
```
        products[productCnt].price = price;
        productCnt++;
        cout << "Product added successfully!\n";
    } else {
        cout << "ID is already present!\n";
    }
}
bool hasUniqueId(int id) {
    for (int i = 0; i < productCnt; i++) {
        if (products[i].unique_id == id) {
            return false;
        }
    }
    return true;
}
void removeProduct(int id) {
    int index = -1;
    for (int i = 0; i < productCnt; i++) {
        if (products[i].unique_id == id) {
            index = i;
            break;
        }
    }

    if (index == -1) {
        cout << "Product not found!\n";
        return;
    }

    for (int i = index; i < productCnt - 1; i++) {
        products[i] = products[i + 1];
```

```
    }
    productCnt--;
    cout << "Product removed successfully!\n";
}
void sortProductsById() {
    for (int i = 0; i < productCnt - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < productCnt; j++) {
            if (products[j].unique_id < products[minIndex].unique_id) {
                minIndex = j;
            }
        }
        Product temp = products[minIndex];
        products[minIndex] = products[i];
        products[i] = temp;
    }
}
int binarySearch(int id) {
    sortProductsById();

    int left = 0;
    int right = productCnt - 1;

    while (left <= right) {
        int mid = (left + right) / 2;
        if (products[mid].unique_id == id) {
            return mid; // return the index if found
        } else if (products[mid].unique_id < id) {
            left = mid + 1;
        } else {
            right = mid - 1;
```

```
            }
        }
        return -1; // not found
    }
    void displayInventory() {
        for (int i = 0; i < productCnt; i++) {
            cout << "ID: " << products[i].unique_id << ", Price: " << products[i].price << '\n';
        }
    }
};

int main() {
    Inventory inventory;
    int total_products;
    cout << "How many products you want to add? ";
    cin >> total_products;
    if (total_products > 0) {
        while (total_products--) {
            int id, price;
            cout << "Enter product id, price: ";
            cin >> id >> price;
            inventory.addProduct(id, price);
        }
    } else {
        cout << "Invalid number!\n";
    }
    cout << "\nInventory after adding products:\n";
    inventory.displayInventory();
    inventory.sortProductsById(); // Sort by unique_id
    cout << "\nInventory after sorting by ID:\n";
    inventory.displayInventory();
```

```
    int searchId;
    cout << "Enter id to search for the product: ";
    cin >> searchId;
    int index = inventory.binarySearch(searchId);
    if (index != -1) {
        cout << "\nProduct found - ID: " << inventory.products[index].unique_id
             << ", Price: " << inventory.products[index].price << '\n';
    } else {
        cout << "\nProduct with ID " << searchId << " not found.\n";
    }
    int removeProductById;
    cout << "Enter the product id to remove the product: ";
    cin >> removeProductById;
    inventory.removeProduct(removeProductById);
    cout << "\nInventory after removing product with ID " << removeProductById << "\n";
    inventory.displayInventory();
    return EXIT_SUCCESS;
}
```

OUTPUT:


Wed Nov 13 2:14 PM
46%
maqsood@maqsood-HP-EliteBook-840-G5: ~/My_Data/Code_Playground/DSA/Assignment_01
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ g++ d.cpp -o d.out
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$ ./d.out
How many products you want to add? 2
Enter product id, price: 29 8990
Product added successfully!
Enter product id, price: 10 700
Product added successfully!
Inventory after adding products:
ID: 29, Price: 8990
ID: 10, Price: 700
Inventory after sorting by price:
ID: 10, Price: 700
ID: 29, Price: 8990
Enter id to search the product: 29
Product found - ID: 29, Price: 8990
Enter the product id to remove the product: 10
Product removed successfully!
Inventory after removing product with ID 10
ID: 29, Price: 8990
maqsood@maqsood-HP-EliteBook-840-G5:~/My_Data/Code_Playground/DSA/Assignment_01$

The End