United International University

Department of Computer Science and Engineering

Course Code: CSI 217 | Course name: Data Structure and Algorithms - I

Laboratory Section O

Spring 24 # Mid Class Performance

Total Marks: 20 Time: 1.00 hour

Question 1: Sorting problem (Mark 5)

Given an unsorted array of integers, sort the array into a wave array. An array arr(O..n-11 is sorted in wave

form if:

arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= .....

Examples:

Input: arr[] = {10, 5, 6, 3, 2, 20, 100, 80}

Output: arr[] = {10, 5, 6, 2, 20, 3, 100, 80}

Explanation:

here you can see {10, 5, 6, 2, 20, 3, 100, 80} first element is larger than the second and the same thing is

repeated again and again. large element — small element-large element -small element and so on. it can be small element-larger element — small element-large element -small element too. all you need to

maintain is the up-down fashion which represents a wave. there can be multiple answers.

Input: arr[] = {20, 10, 8, 6, 4,2}

Output: arr[] = {20, 8, 10, 4, 6, 2}

```
#include <stdio.h>

void print(int ar[], int len)
{
    for (int i = 0; i < len; i++)
    {
        printf("%d ", ar[i]);
    }
    printf("\n");
}
```

```c
int main()
{
    // int arr[] = {10, 5, 6, 3, 2, 20, 100, 80};
    int arr[] = {20, 8, 10, 4, 6, 2};

    int len = sizeof(arr) / sizeof(int);

    printf("Unsorted Array:\n");
    print(arr, len);

    for (int i = 1; i < len; i++)
    {

        if (i % 2 == 0)
        {

            if (arr[i] > arr[i + 1])
            {
                i++;
                continue;
            }

            else
            {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }

        else
        {
            if (arr[i] < arr[i + 1])
            {
                i++;
                continue;
            }

            else
            {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }

    printf("Sorted Array:\n");

    print(arr, len);

    return 0;

}
```

Question 2: Binary Search problem (Mark 5)

Given a sorted array of non-negative distinct integers, find the smallest missing non-negative element in it.

For example,

Input: nums[] = [0, 1, 2, 6, 9, 11, 15]

Output: The smallest missing element is 3

Input: nums[] = [1, 2, 3, 4, 6, 9, 11, 15]

Output: The smallest missing element is 0

Input: nums[] = [0, 1, 2, 3, 4, 5, 6]

Output: The smallest missing element is 7

```
#include <stdio.h>

int findSmallestMissing(int nums[], int size)
{
    int left = 0, right = size - 1;

    while (left <= right)
    {

        if (nums[left] != left)
            return left;

        int mid = left + (right - left) / 2;


        if (nums[mid] != mid)
            right = mid - 1;

        else
            left = mid + 1;
    }

    return left;
}

int main()
{

    int nums[] = {0, 1, 2, 6, 9, 11, 15};
    // int nums[]={1,2,3,4,6,9,11,15};

    // int nums[]={0,1,2,3,4,5,6};

    int size = sizeof(nums) / sizeof(nums[0]);

    int smallestMissing = findSmallestMissing(nums, size);

    printf("The smallest missing element is %d\n", smallestMissing);

    return 0;

}
```

Question 3: Link List problem (Mark 5)

Write a function that takes a list sorted in non-decreasing order and deletes any duplicate nodes from the list. The list should only be traversed once.

For example,

if the linked list is 11->11->11->21->43->43->60

then removeDuplicates() should convert the list to 11->21->43->60

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node *push(int value)
{

    struct node *node = malloc(sizeof(struct node));
    node->data = value;
    node->next = NULL;

    return node;
}

void duplicate(struct node *head)
{

    struct node *current = head;
    struct node *temp = current;

    while (temp != NULL)
    {
        temp = temp->next;
        if (current->data == current->next->data)
        {
            current->next = current->next->next;
            continue;
        }

        current = current->next;
        temp = current->next;
    }

}

void print(struct node *head)
{
    struct node *current = head;

    while (current != NULL)
    {
        printf("%d ", current->data);
        current = current->next;
    }

    printf("\n");
}

int main()
{

    struct node *head = NULL;
    struct node *position = head;

    head = position = push(11);

    position->next = push(11);
    position = position->next;

    position->next = push(11);
    position = position->next;

    position->next = push(21);
    position = position->next;

    position->next = push(43);
    position = position->next;

    position->next = push(43);
    position = position->next;

    position->next = push(60);
    position = position->next;


    printf("Linked list is:\n");
    print(head);
    duplicate(head);
    printf("\nAfter Removing Duplicate Element:\n");
    print(head);
    return 0;
}
```

github link: 5S4D1