Rajalakshmi Engineering College

Name: Ravi Sankar
Email: 240701424@rajalakshmi.edu.in
Roll no: 240701424
Phone: 8122932671
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Priya, a data analyst, is working on a dataset of integers. She needs to find the maximum difference between two successive elements in the sorted version of the dataset. The dataset may contain a large number of integers, so Priya decides to use QuickSort to sort the array before finding the difference. Can you help Priya solve this efficiently?

Input Format

The first line of input consists of an integer n, representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer, representing the maximum difference between

two successive elements in the sorted form of the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
10
Output: Maximum gap: 0

Answer

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

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
```

```c
}
int maximumGap(int arr[], int n) {
    if (n < 2) {
        return 0;
    }

    quickSort(arr, 0, n - 1);

    int max_diff = 0;
    for (int i = 1; i < n; i++) {
        int diff = arr[i] - arr[i - 1];
        if (diff > max_diff) {
            max_diff = diff;
        }
    }

    return max_diff;
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int max_gap = maximumGap(arr, n);

    printf("Maximum gap: %d\n", max_gap);

    return 0;
}
```

Status : Correct   *Marks : 10/10*

2. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor.

She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

Input Format

The first line of input consists of an integer n, representing the number of children.

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

Output Format

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
2
1 1
Output: The child with greed factor: 1

Answer

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

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
```

```c
        *b = temp;
    }

    int partition(int arr[], int low, int high) {
        int pivot = arr[high];
        int i = low - 1;

        for (int j = low; j <= high - 1; j++) {
            if (arr[j] < pivot) {
                i++;
                swap(&arr[i], &arr[j]);
            }
        }
        swap(&arr[i + 1], &arr[high]);
        return i + 1;
    }

    void quickSort(int arr[], int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

    int findContentChildren(int g[], int n, int s[], int m) {
        quickSort(g, 0, n - 1);
        quickSort(s, 0, m - 1);

        int i = 0, j = 0, happy = 0;
        while (i < n && j < m) {
            if (s[j] >= g[i]) {
                happy++;
                i++;
                j++;
            } else {
                j++;
            }
        }

        return happy;
    }
```

```c
int main() {
    int n, m;
    scanf("%d", &n);
    int g[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &g[i]);
    }

    scanf("%d", &m);
    int s[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &s[i]);
    }

    int happy = findContentChildren(g, n, s, m);

    printf("The child with greed factor: %d\n", happy);

    return 0;
}
```

Status : Correct   *Marks : 10/10*

3. Problem Statement

Meera is organizing her art supplies, which are represented as a list of
integers: red (0), white (1), and blue (2). She needs to sort these supplies
so that all items of the same color are adjacent, in the order red, white, and
blue. To achieve this efficiently, Meera decides to use QuickSort to sort the
items. Can you help Meera arrange her supplies in the desired order?

Input Format

The first line of input consists of an integer n, representing the number of items
in the list.

The second line consists of n space-separated integers, where each integer is
either 0 (red), 1 (white), or 2 (blue).

Output Format

The output prints the sorted list of integers in a single line, where integers are arranged in the order red (0), white (1), and blue (2).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
2 0 2 1 1 0
Output: Sorted colors:
0 0 1 1 2 2

Answer

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void sortColors(int nums[], int n) {
    int low = 0, mid = 0, high = n - 1;
    while (mid <= high) {
        if (nums[mid] == 0) {
            swap(&nums[low], &nums[mid]);
            low++;
            mid++;
        } else if (nums[mid] == 1) {
            mid++;
        } else {
            swap(&nums[mid], &nums[high]);
            high--;
        }
    }
}

int main() {
    int n;
    scanf("%d", &n);
```

```c
    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    sortColors(nums, n);

    printf("Sorted colors:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", nums[i]);
    }
    printf("\n");

    return 0;
}
```

Status : Correct   *Marks : 10/10*