DSA (CSE102) Practice Questions for lab

03-02-25

1 The Lost Treasure of Captain Jack

Captain Jack, the legendary pirate, has discovered an ancient scroll hidden deep inside the Cursed Caverns of Skull Island. The scroll contains a sorted sequence of numbers and a mystical matrix of runes, both believed to be keys to unlocking a legendary treasure chest.

However, the scroll presents two challenges:

> "Two numbers in this sacred list,
> Their sum unveils what you have missed.
> Find them true, find them fast,
> Or the treasure will never last!"

> "Two mighty squares, side by side,
> Multiply them well, let the rules be your guide.
> The answer reveals the pirate's gold,
> Solve it quick, let the legend unfold!"

Captain Jack must first solve the two-sum problem using pointers to reveal part of the treasure map. Then, he must perform matrix multiplication using pointers to decode the final location of the hidden gold.

Can you help Captain Jack before the cavern collapses?

—

2 Challenge 1: Two-Sum Using Pointers

Since the array is already sorted, we can efficiently find the two numbers using the two-pointer approach, which runs in $\mathcal{O}(n)$ time complexity.

2.1 C Implementation

```
#include <stdio.h>

void findTwoSum(int *arr, int size, int target) {
    int *left = arr;
```

```
    int *right = arr + size - 1;

    while (left < right) {
        int currentSum = *left + *right;

        if (currentSum == target) {
            printf("Captain Jack finds the secret numbers: (%d, %d)\n", *left, *right);
            return;
        }
        else if (currentSum < target) {
            left++;
        }
        else {
            right--;
        }
    }

    printf("No pair found, the treasure remains hidden!\n");
}

int main() {
    int arr[] = {1, 3, 5, 7, 9, 11, 15};
    int target = 12;
    int size = sizeof(arr) / sizeof(arr[0]);

    findTwoSum(arr, size, target);
    return 0;
}
```

—

3 Challenge 2: Matrix Multiplication Using Pointers

Captain Jack finds the next clue hidden within two square matrices. The final treasure location is encoded in their product. Your task is to multiply two given $N \times N$ matrices using C pointers.

3.1 C Implementation

```
#include <stdio.h>

#define N 3

void multiplyMatrices(int (*A)[N], int (*B)[N], int (*result)[N]) {
```

```
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            *(*(result + i) + j) = 0;
            for (int k = 0; k < N; k++) {
                *(*(result + i) + j) += *(*(A + i) + k) * *(*(B + k) + j);
            }
        }
    }
}

void printMatrix(int (*matrix)[N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", *(*(matrix + i) + j));
        }
        printf("\n");
    }
}

int main() {
    int A[N][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int B[N][N] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
    int result[N][N];

    multiplyMatrices(A, B, result);

    printf("The decrypted matrix is:\n");
    printMatrix(result);

    return 0;
}
```

—

4 Practice Questions

To further test your skills with pointers in C, try solving the following problems:

4.1 1. Binary Search in Two Sorted Arrays

You are given two sorted arrays of size m and n. Your task is to find the median of the two sorted arrays in $\mathcal{O}(\log(m+n))$ time complexity.

Example:

$$A = [1, 3, 8], \quad B = [7, 9, 10, 11]$$

Output: 8.0

Constraints: - The arrays are sorted. - You must use binary search.

—

4.2 2. Three-Sum Problem

Given an array of n integers, find all unique triplets in the array that sum up to zero.

Example:

$$\text{Input: } [-1, 0, 1, 2, -1, -4]$$

$$\text{Output: } [[-1, -1, 2], [-1, 0, 1]]$$

Constraints: - The solution must run in $\mathcal{O}(n^2)$ time complexity. - You cannot use extra space (i.e., no hash maps).

—

4.3 3. Largest Subarray Sum Less Than k

Given an array of positive and negative integers, find the largest sum of a contiguous subarray that is $\leq k$.

Example:

$$\text{Input: } A = [3, -2, 5, -1], \quad k = 6$$

$$\text{Output: } 6$$

Constraints: - The array may contain negative numbers. - The solution must be efficient ($\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$).