CSE211 DATA STRUCTURES

LAB 2 FALL 2024

STACK OPERATIONS

Prerequisites

Open the terminal and execute the following commands after downloading the `tarball` file:

```
cd /mnt/c/Users/user/Downloads && tar -xvf lab2_3.tar.gz --one-top-
level=lab2_3
cd /mnt/c/Users/user/Downloads/lab2_3 && make all
code .
```

Introduction

In this lab, you will implement advanced operations on a Stack data structure using C++. The Stack is implemented as a template class that can store elements of any type T. Your task is to implement the following challenging operations:

1. `stackPermutation`: Check if one stack can be converted to another through push/pop operations
2. `sortedInsert`: Insert elements while maintaining descending order
3. `maxHistogramArea`: Find largest rectangular area in histogram represented by stack
4. `maxDiffNeighbors`: Find maximum absolute difference between adjacent elements

Project Structure

```
.
├── bin/
│   └── stack
├── include/
│   ├── Stack.hpp
│   └── Color.hpp
├── obj/
│   ├── Stack.o
│   ├── Color.o
│   └── main.o
├── src/
│   ├── Stack.cpp
│   ├── Color.cpp
│   └── main.cpp
├── instructions.md
└── Makefile
```

Implementation Details

1. stackPermutation

- **Purpose**: Check if one stack can be converted to another using only push/pop operations with one auxiliary stack
- **Parameters**: target stack to compare against
- **Return**: true if permutation is possible, false otherwise
- **Example**:

```
Input Stack:  [1, 2, 3] (top)
Target Stack: [2, 1, 3] (top)
Output: true  // Possible through push/pop operations
```

2. sortedInsert

- **Purpose**: Insert a new element while maintaining stack in descending order (largest at top)
- **Parameters**: value to insert
- **Example**:

```
Initial: [9, 5, 1] (top)
Insert 2
Result: [9, 5, 2, 1] (top)
```

3. maxHistogramArea

- **Purpose**: Given heights of bars in histogram, find largest rectangular area possible
- **Return**: Maximum area value
- **Example**:

```
Input:  [6, 1, 5, 4, 5, 2, 6] (top)
Output: 12  // Area = height(4) * width(3)
<==========================================>
6 █               █
5 █     █   █     █
4 █   ->█ █ █<-   █  // This is the maximum area
3 █   | █ █ █ |   █  // height = 4, width = 3
2 █   | █ █ █ | █ █  // area = 4 * 3 = 12
1 █ █ | █ █ █ | █ █
```

4. maxDiffNeighbors

- **Purpose**: Find maximum absolute difference between any two adjacent elements
- **Return**: Maximum difference value
- **Example**:

```
Input:  [2, 8, 1, 5, 3] (top)
Output: 7  // |8-1| = 7 is max difference
```

Testing

1. Build and run:

```
make clean  # Clean previous builds
make all    # Compile all files
make run    # Execute the program
```

Restrictions

❌ Do not modify:

- Stack.hpp interface
- main.cpp test cases
- Project structure
- Build system

❌ Do not use:

- External libraries
- Global variables
- Additional data structures (except where specified)

Academic Integrity

- Individual work only
- No code sharing
- No plagiarism
- Violations result in zero grade

Submission

1. Test thoroughly
2. Clean build files: `make clean`
3. Send only the `Stack.cpp` file to the course portal

Good luck with your implementation!