M22CS1.304 Data Structures and Algorithms for Problem Solving

Assignment 1

Deadline: 11:59 pm. August 22, 2022

Important Points:

1. Only C/C++ is allowed.
2. Submission Format: Put all the files in a folder with the folder name as your roll number and submit the zip file on moodle. Use the following naming formats:
    a. For cpp files: <RollNo>_q<Question_No>.cpp
        E.g. 2021201001_q1.cpp
    b. For folder: <RollNo>_<Assignment_No>
        E.g. 2021201001_A1
    c. For Zip File: <RollNo>_<Assignment_No>.zip
        E.g. 2021201001_A1.zip

    Note: All submissions which are not in the specified format or submitted after the deadline will be awarded **0** in the assignment.
3. C++ STL is **not allowed** for any of the questions unless specified otherwise in the question. So “#include <bits/stdc++.h>” is not allowed.
4. You can ask queries by posting on the moodle.

Any case of plagiarism will lead to a 0 in the assignment or “F” in the course.

1. Big Integer Library

Problem Statement: Create a big integer library, similar to the one available in Java. The library should provide functionalities to store arbitrarily large integers and perform basic math operations.

Operations:

1. Addition(+), subtraction(-), multiplication(x, lowercase “X”).
   E.g.
   Input: 32789123**+**99893271223**x**9203232392**-**4874223
   Output: 919340989462382970316
   Note: Negative numbers won’t be present in the intermediate or final output (i.e. No need to consider cases like 2-3)
2. Exponentiation.
   a. Base will be a big int and exponent will be < 2^{63}
3. GCD of two numbers.
4. Factorial.

Note: For all the operations, input will be such that the number of digits in output won’t exceed 3000 digits.

Input Format

First line will contain an integer value which denotes the type of operation. The integer value and operation mapping is as follows:

1. Addition, Subtraction & Multiplication
2. Exponentiation
3. GCD
4. Factorial

The following line will contain input according to the type of operation. For 1st and 4th type of operation, there will be one string and for the 2nd & 3rd type of operation, there will be 2 space separated strings.

Evaluation parameters:

Accuracy of operations and performance.

Example:

- **Sample input:** 1
  1+2+3x5-2
  Sample output: 16

- **Sample input:** 2
  2 10
  Sample output: 1024

- **Sample input:** 3
  9 15
  Sample output: 3

- **Sample input:** 4
  12
  Sample output: 362880

2. Deque

Problem Statement: Implementation of Deque

What is deque?

- Deque is the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container
- They support insertion and Deletion from both ends in amortised constant time.
- Inserting and erasing in the middle is linear in time.

Expected Solution:

The C++ standard specifies that a legal (i.e., standard-conforming) implementation of deque must satisfy the following performance requirements: (consider the data type as **T**)

1. **void deque()** - initialise a empty deque.Time complexity: O(1)
2. **void deque(n, x)** - initialise a deque of length n with all values as x. Time complexity: O(n)
3. **void push_back(x)** - append data x at the end. Time complexity: O(1)
4. **void pop_back()** - erase data at the end. Time complexity: O(1)
5. **void push_front(x)** - append data x at the beginning. Time complexity: O(1)
6. **void pop_front()** - erase an element from the beginning. Time complexity: O(1)
7. **T front()** - returns the first element(value) in the deque. Time complexity: O(1)
8. **T back()** - returns the last element(value) in the deque. Time complexity: O(1)
9. **T D[n]** - returns the nth element of the deque. You need to overload the [ ] operator. Time complexity: O(1)
10. **bool empty()** - returns true if deque is empty else returns false. Time complexity: O(1)
11. **int size()** - returns the current size of deque(i.e. the number of elements present in the deque). Time complexity: O(1)

12. **void resize(x, d)** - change the size dynamically to new size **x**. Time complexity: O(n)
    - If the new size **x** is greater than the current size of the deque, then insert new elements with default value **d** at the end of the queue.
    - If the new size **x** is smaller than the current size, then keep **x** elements from the beginning of the deque.
13. **void clear()** - remove all elements of deque. Time complexity: O(1)

Note: Your deque should be generic type i.e. it should be datatype independent and can support primitive datatypes like integer, float, string, etc.
Hint: Use template in C++ (link)

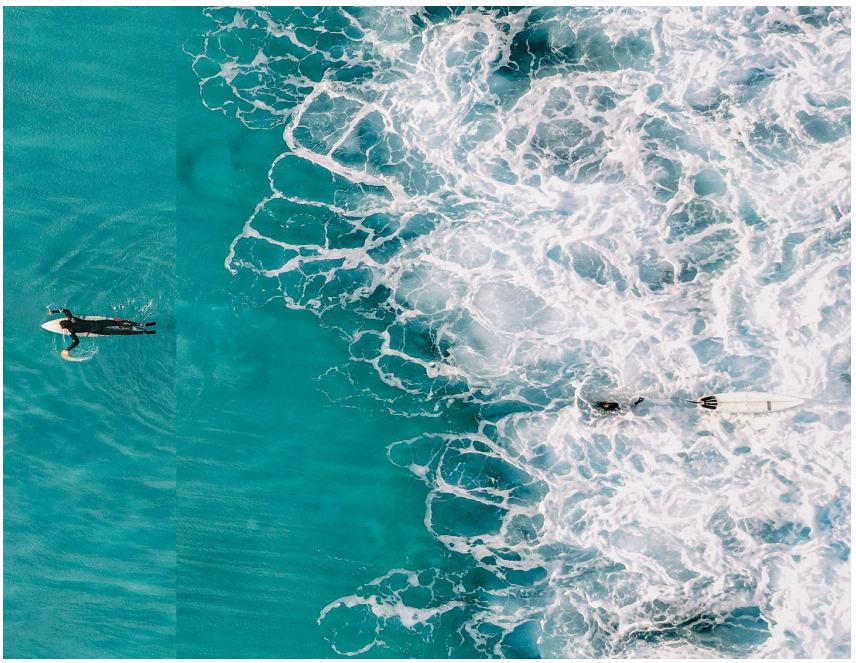
Evaluation parameters: Accuracy of operations and performance.

3. Implement Seam Carving Algorithm

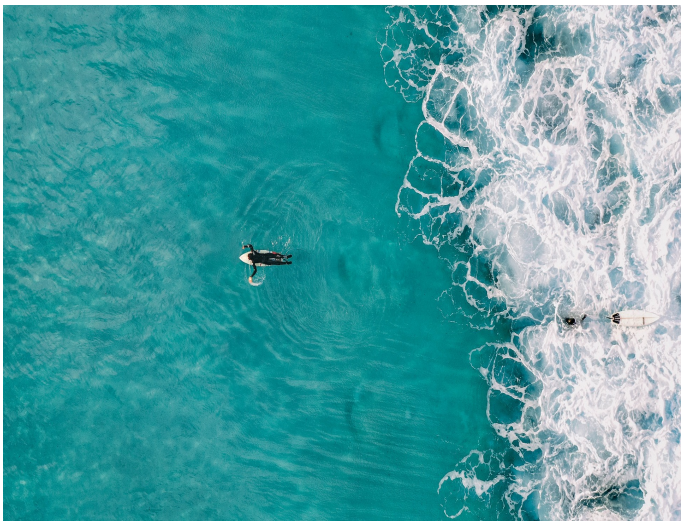
Problem Statement: Apply seam carving content aware image-resizing algorithm on a given image. Take the height and width (in pixels) of the output image as inputs from the user.

Original Image:

Resize Image by cropping

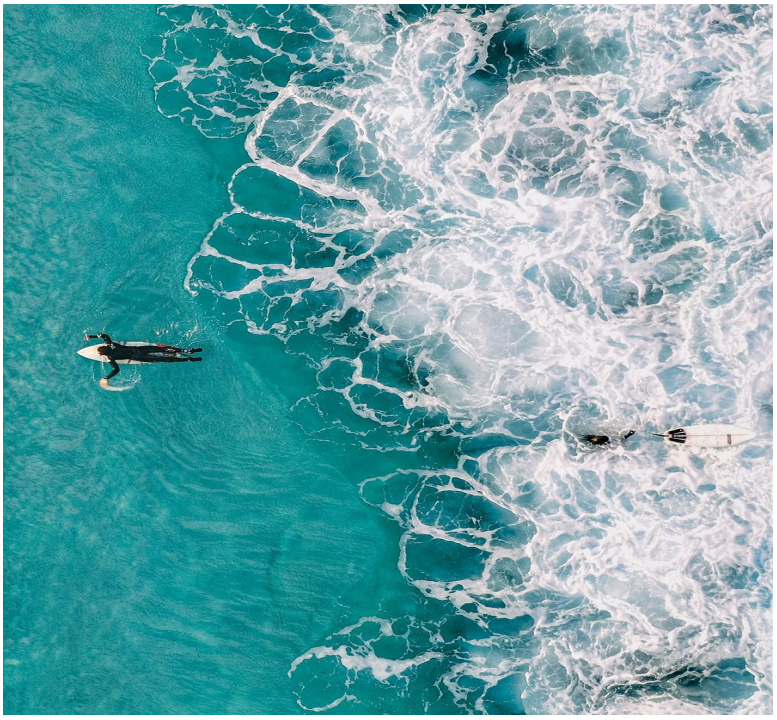


Resize Image by Scaling



Lowest Energy Seam

Resize using Seam Carving



What is Seam Carving?

- Seam-carving is a content-aware image resizing technique where the image is reduced in size by one pixel of height (or width) at a time.
- A vertical seam in an image is a path of pixels connected from the top to the bottom with one pixel in each row.
- A horizontal seam is a path of pixels connected from the left to the right with one pixel in each column.
- Steps:
    - Energy Calculation: Each pixel has some RGB values. Calculate energy for each pixel. For ex.- You can use *dual-gradient energy function* but you are free to use any energy function of your choice. Also, you can refer to this link for details.
    - Seam Identification: Identify the lowest energy seam.
    - Seam Removal: Remove the lowest energy seam.

Program Flow:

1. Extract individual pixel's RGB values from the sample image ./data/input/sample.jpeg and write them into ./data/input/rgb_in.txt
2. Load the RGB values from ./data/input/rgb_in.txt in a 3D matrix.
3. Apply seam carving algorithm.
4. Write the individual pixel's RGB values for resized image into the ./data/output/rgb_out.txt
5. Generate sample image output ./data/output/sample_out.jpeg using the ./data/output/rgb_out.txt.

For your convenience we've provided you a python script ./src/driver.py which will perform 1st and 5th task. Also, there is a corresponding ./src/main.cpp file which will be executed by the python script which will perform the 2nd and 4th task.

Your Task:

You have to write the code for seam carving algorithm inside the solve() function of main.cpp. i.e. You just have to perform the 3rd task.

Dependencies:

You'll need to install the python image library Pillow to extract RGB values of each pixel and to generate images back from RGB values of each pixel.

To install Pillow: `pip install Pillow`

The python script is only compatible with Linux/Mac operating systems. If there are enough Windows users, we might release the script for Windows too.

If you're a windows user you can request us for the input text file(rgb_in.txt) so that you can start working on the problem.

How to Run:
Open the Q3 directory in the terminal and run the python file driver.py located in /src. Also, you need to pass the input image filename(must be present inside /data/input) as a command line argument.
Ex: **`python ./src/driver.py sample1.jpeg`**

Submission Format:
You only have to submit the CPP file. Rename the cpp file to <roll_no.>_q3.cpp and submit it according to the submission format mentioned at the beginning of the document.

Evaluation parameters:
Accuracy and Performance of the code.