INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR
Department of Computer Science and Engineering

CSL 210 – Data Structures with Applications

Module 1 –Coding Assignments Practice set

Q1) Lets turn our attention to a mathematical object that is used to solve many problems in the natural sciences, the matrix. As computer scientists, our interest centers not only on the specification of an appropriate ADT, but also on finding representations that let us efficiently perform the operations described in the specification.

We can use an array of triples to represent a sparse matrix. Since we want our transpose operation to work efficiently, we should organize the triples so that the row indices are in ascending order. We can go one step further by also requiring that all the triples for any row be stored so that the column indices are in ascending order. In addition, to ensure that the operations terminate, we must know the number of rows and columns, and the number of nonzero elements in the matrix. Putting all this information together suggests that we implement the *Create* operation as below:

structure *Sparse_Matrix* is
objects: a set of triples, <*row*, *column*, *value*>, where *row* and *column* are integers and form a unique combination, and *value* comes from the set *item*.
functions:
for all $a, b \in$ *Sparse_Matrix*, $x \in$ *item*, i, j, *max_col*, *max_row* $\in$ *index*

Sparse_Matrix Create(*max_row*, *max_col*) ::=
return a *Sparse_Matrix* that can hold up to *max_items* = *max_row* × *max_col* and whose maximum row size is *max_row* and whose maximum column size is *max_col*.

Sparse_Matrix Transpose(*a*) ::=
return the matrix produced by interchanging the row and column value of every triple.

Sparse_Matrix Add(*a*, *b*) ::=
if the dimensions of *a* and *b* are the same
return the matrix produced by adding corresponding items, namely those with identical *row* and *column* values.
else return error

Sparse_Matrix Multiply(*a*, *b*) ::=
if number of columns in *a* equals number of rows in *b*
return the matrix *d* produced by multiplying *a* by *b* according to the formula: $d[i][j] = \sum(a[i][k] \cdot b[k][j])$ where $d(i, j)$ is the (i, j)th element
else return error.

(a) Write C functions *read-matrix, print-matrix,* and *search* that read triples into a new sparse matrix, print out the terms in a sparse matrix, and search for a value in a sparse matrix. Analyze the computing time of each of these functions.

(b) Rewrite *fast-transpose* so that it uses only one array rather than the two arrays required to hold *row-terms* and *starting-pos.*

(c) Analyze the time and space requirements of *fast-transpose.* What can you say about the existence of a faster algorithm?

Q2) You are tasked with developing a C program for symbolic polynomial manipulation, which allows users to perform various operations on polynomials represented as linked lists. Your program should support addition, subtraction, multiplication, and symbolic differentiation of polynomials.

1. Implement a data structure for representing polynomials using a linked list. Each node in the list should store a coefficient and an exponent. Ensure that your implementation can handle both positive and negative exponents.
2. Create functions to perform the following operations on polynomials:
    - Addition: Add two polynomials together and return the result as a new polynomial.
    - Subtraction: Subtract one polynomial from another and return the result as a new polynomial.
    - Multiplication: Multiply two polynomials and return the result as a new polynomial.
    - Symbolic Differentiation: Compute the derivative of a given polynomial and return the result as a new polynomial.
3. Implement a function to simplify polynomials by combining like terms. For example, if the result of a polynomial addition contains terms with the same exponent, these terms should be combined.
4. Develop a user interface that allows users to input two polynomials and choose an operation (addition, subtraction, multiplication, or differentiation). Display the simplified result of the chosen operation.
5. Optimize your program to handle large polynomials efficiently. Consider using data structures like trees or hash tables for faster polynomial manipulation.

Q3) You are tasked with implementing a hash table in C that stores key-value pairs. Your goal is to develop a hash table that efficiently handles collisions by implementing open addressing with linear probing.

1. Implement a hash table data structure that supports key-value storage using open addressing with linear probing. Ensure it can handle collisions gracefully.
2. Develop functions for inserting key-value pairs, retrieving values by their keys, and deleting key-value pairs from the hash table.
3. Implement a function to dynamically resize the hash table when its load factor exceeds a certain threshold.
4. Write a program that tests your hash table implementation with a large dataset and a variety of operations (insertion, retrieval, deletion).

Q4) You are tasked with implementing a program that performs multi-precision arithmetic and uses a hash table for efficient storage and retrieval of results. The program will perform calculations on extremely large integers and store intermediate results in a hash table to save time on subsequent operations.

1. Implement a multi-precision arithmetic library in C that can handle integers of arbitrary length (more than 10^6 digits). Your library should include functions for addition, subtraction, multiplication, and division of large integers.
2. Develop a hash table data structure that supports storing and retrieving large integers as key-value pairs. Ensure that your hash table can handle collisions gracefully.
3. Create a program that uses your multi-precision arithmetic library to perform a series of mathematical operations on very large integers. These operations should include addition, subtraction, multiplication, and division.
4. Implement a caching mechanism using the hash table to store intermediate results of calculations. Before performing an operation, check if the result is already in the hash table. If it is, retrieve it from the cache; otherwise, perform the calculation and store the result in the hash table.

Q5) You are tasked with implementing a radix sort algorithm in C for sorting large arrays of integers efficiently.

1. Implement a radix sort algorithm that can handle large arrays of integers. Ensure that your implementation can handle both positive and negative integers.
2. Develop a function for sorting an array of integers using radix sort.
3. Optimize your radix sort implementation to minimize memory usage and maximize efficiency for large datasets.
4. Write a program that generates a large array of random integers, sorts it using your radix sort algorithm, and measures the execution time and memory usage.

Q6) You are tasked with implementing a hybrid sorting algorithm that combines radix sort with hash tables for efficient sorting of strings.

1. Implement a hash table data structure in C that supports key-value pairs for strings.
2. Develop a hybrid sorting algorithm that uses radix sort to first group strings by their first character and then uses a hash table to sort each group of strings.
3. Write a program that generates a large collection of random strings, sorts them using your hybrid sorting algorithm, and measures the execution time and memory usage.

Q7) You are tasked with implementing a C program that efficiently represents and performs operations on sparse matrices using multi-linked structures. Sparse matrices are matrices with a significant number of zero elements, and your program should optimize memory usage and support common matrix operations.

1. Implement a multi-linked structure to represent a sparse matrix. Your representation should include a structure for nodes that contain row, column, and value information for non-zero elements. Ensure efficient traversal and memory usage.
2. Create functions for the following operations on sparse matrices:
    - Matrix Addition: Add two sparse matrices and return the result as a new sparse matrix.
    - Matrix Multiplication: Multiply two sparse matrices and return the result as a new sparse matrix.
    - Scalar Multiplication: Multiply a sparse matrix by a scalar.

    - Transposition: Compute the transpose of a sparse matrix and return it as a new sparse matrix.
    - Matrix Display: Display the sparse matrix in a user-friendly format, showing only non-zero elements.
3. Implement an efficient mechanism for sparse matrix-vector multiplication. This operation multiplies a sparse matrix by a dense vector and returns the resulting vector.
4. Develop a user interface that allows users to input sparse matrices, choose an operation, and display the result. Ensure that the interface can handle large sparse matrices effectively.

Q8) Write a C program to add two polynomials, where the first polynomial is required to be represented using a linked list. For 2nd polynomial instead of creating a new linked list, as soon as you get the input values from the user traverse the linked list for correct exponent value and add coefficient.

Q9) Write a function to evaluate a polynomial represented as a linked list at a given value 'x.' Return the result of the evaluation. Consider x as some floating point number.

Q10) Write a function to simplify a polynomial represented as a linked list by combining like terms. The input and output should be linked lists.

Q11) Design a 3 variable polynomial calculator that allows users to input two polynomials and choose an operation (addition, subtraction, multiplication, etc.). Implement this calculator using linked list representations for polynomials.

Q12) Write a function to compare two long integer linked lists and determine which one is greater, less than, or equal to the other.

Q13) "A" is a 25 digit long integer which needs 25 nodes to represent it in linked list representation format.

A can be given as – 11112233344445556677788990.

Write a function to store above integer in the linked list representation format with total nodes less than 25.

********** Complex Questions **********

Q14) Consider a Sparse matrix is represented in the form of a triplet in row major format [*rowindex*,*columnindex*,*value*] .

You need to traverse the sparse matrix starting from an element for which the row index and the column index is given as an input. Then traverse the elements from the starting point in this order: *Down* (1), R*ight* (2), U*p* (3), and then L*eft* (4).

The output should be in the form of a quadruplet with rows arranged in the required order of traversal. The first three columns of this quadruplet are similar to those as given in the input triplet. The last column includes the direction information which is followed when the corresponding element is traversed. The allowed values in the last column are:

- 1: If element is traversed while moving *Down*.

- 2: If element is traversed while moving *Right*.
- 3: If element is traversed while moving *Up*.
- 4: If element is traversed while moving *Left*.

Input:

- Line 1 contains three space separated integers, *P*, *Q*, and *N*, where *P* is the first and *Q* is the second dimension of a Sparse matrix. *N* is the total number of non-zero entries in the Sparse matrix.
- Each of the following *N* lines contains three space separated integers representing a single row of the triplet, i.e. *rowindex*, *columnindex*, and *value*.
- Last line contains two space separated integers, *I* and *J*, respectively giving the starting row and column indices for helix traversal of the Sparse matrix.

Output:

- Each row of the quadruplet is contained in *N* separate lines. Each line in the output has four space separated integers in the format as is described above.

Constraints

- All integers range in between 1 and 100.

Sample Input: 445 008 116 125 239 327 11

Sample Output: 1161 1253 0084 3272 2393

EXPLANATION:

We have to start from [1][1] and direction to be followed is *Down* (1), so the first entry should be 1161. Next entries traversed in the reverse spiral form are at indices [2][1] and [2][2], but corresponding triplet entries are missing so no output is generated. Then 5 comes at [1][2] while moving *Up* (3). Triplet entries for [0][2] and [0][1] are again missing. Then at [0][0] element 8 comes while moving L*eft* (4). Again missing triplet entries for [1][0], [2][0], [3][0], and [3][1] generate no output. Element 7 is visited when moving *Right* (2) at index [3][2]. Triplet entry for [3][3] is missing. Lastly we found element 9 at [2][3] when moving *Up* (3). Rest of the triplet entries are missing, i.e. [1][3] and [0][3]. This completes the helix traversal.

Q15) **SaapSidi Ladders - Find Ladders from the Sparse Matrix**

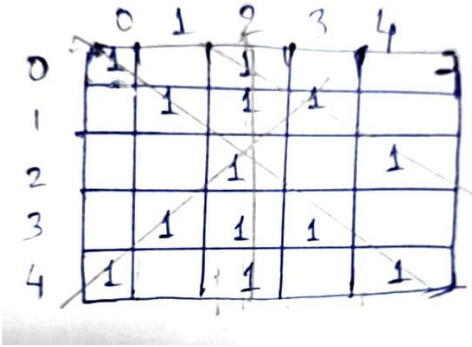
A Sparse matrix is given to you in the form of rowindex (i), columnindex (j) pair of its corresponding non-zero values. Your task in to find out the ladders of **length minimum 2 or more** present in the given matrix. A ladder should be considered for any pair of rowindex and columindex values, where maximum difference is 1.

Input will have two lines-
1st line indicate the maximum rows and columns i.e. M,N of sparse matrix A[M][N].
2nd line contains comma separated pairs of nonzero row index & column index

Output
Print total possible ladders.
Ladder indexes in the form of rowindex, columnindex pairs

Sample

Input
5,5
00, 02, 11, 12, 13, 22, 24, 31, 32, 33, 40, 42, 44



Output
4
00-11-22-33-44 (rowindex, columnindex pair)
02-12-22-32-42
02-13-24
13-22-31-40

Q16) **K-Zigzag Folds Linked List**

Create a linked list with the given set of integer sequence. A ***K*** value would be given to you to apply K-Zigzag folds of the linked list. In each fold you have to divide nodes in equal numbers. Whenever you fold, take the product of values for the nodes which are overlapping with each other and print the resultant linked list.

Input will have 3 lines-
On 1st line total number of nodes will be given,
2nd line contains comma separated integer sequence.
3rd line contains ***K*** value.

Output
The resultant linked list is required to be printed after applying K-Zigzag folds

Sample Test Case

Input

8
3, 6, 2, 8, 12, 20, 7, 9
4

Output
(3*8*12*9 =) **2592 → (**6*2*20*7=) **1680 → Null**

Explanation:
In input there are total 8 nodes and ***K*** value is 4, therefore **4 zigzag folds** will be performed on the linked list. In this case 8 nodes are equally divided into 4 parts. Here Nodes 1,4,5,8 are overlapped and product is stored in the resultant linked list. Similarly, nodes 2,3,6,7 are overlapped and there product is stored in the resultant.
Therefore in the resultant linked list we have only 2 nodes with values 2592 → 1680 → Null

Q17) Consider a variant of singly-linked list. In this linked list, instead of pointing towards a single linked-list node, some nodes of the linked-list are pointing towards two nodes. Each node in this linked-list has two pointers named: **next** and **child**

The "child" pointer of a node may again point towards a strange singly linked list.

You need to restructure this linked-list into a singly linked list. While doing so, she must follow the following rules.

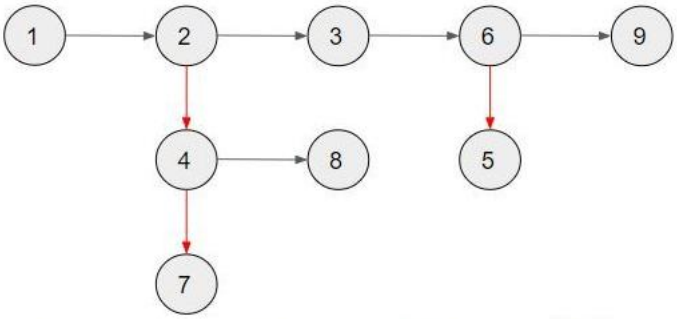
Let **cur** be a node in final restructured linked-list, then **cur.next** must occur after **cur** in the restructured linked-list.

All the nodes (if any) which are part of **cur.child** linked list must occur after the node **cur** and before the node **cur.next** in the restructured linked list.

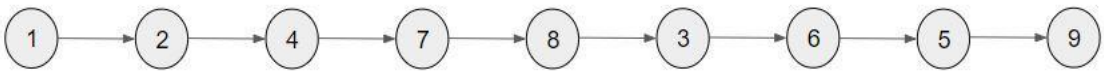
Let **cur** be a node in the restructured linked-list, then **cur.child** must be "null".

NOTE: Refer to sample case for a visual depiction.

Input: Black Arrows=Next pointer and Red Arrows=Child Pointer



Output:



Input Format:

The first line of the input contains a single integer T - the number of test cases. The description of T test cases follows.

The first line of each test case contains a single integer: N

The following N−1 lines contain three space separated integers u,v and type. If the value of type is equal to 0, then the next pointer of node with value equal to u points towards the node with value equal to v. If the value of type is equal to 1, then the child pointer of node with value equal to u points towards the node with value equal to v.

Output Format

For each test case, the function you complete should return the restructured linked-list.

Constraints

$1 \leq T \leq 100$

$1 \leq N \leq 2 * 10^5$

It is guaranteed that the sum of N over all test cases is less than or equal to $3 * 10^5$

Sample Input:

1

9

4 7 1

1 2 0

2 3 0

6 9 0

3 6 0

2 4 1

4 8 0

6 5 1

Output:

1 2 4 7 8 3 6 5 9

Q18) Chef and Squares Problem:

Chef has finished his freshman year in college. As a present, his parents gave him a new problem to solve:

Chef has to fill a K x K square grid of integers in a certain way. Let us say that such a grid is valid if:

Each cell contains an integer from 1 and K (inclusive).

No integer appears twice in the same row or the same column.

Let F(K) be maximum possible distance between the center of the square and the closest cell that contains 1, among all possible squares with the side length K.

Here, we use the following notions:

The distance between cell (x, y) and (i, j) is equal to |x-i|+|y-j|.

The center of a K x K square is cell ((K+1)/2, (K+1)/2) for odd K.

Input

The first line of input contains a single integer T denoting the number of test cases.
Each test case consists of a single line containing a single odd integer K.

Output

For each test case, print K lines each consisting of K space separated integers giving some square grid where the distance from the center of the grid to the nearest 1 is exactly F(K). If there's more than 1 possible answer output any of them.

Constraints

K_i is odd.

Subtask #1:

$1 \leq T \leq 50$

$1 \leq K_i \leq 5$

Sample Input:

2

1

3

Output:

1

3 2 1

1 3 2

2 1 3

Q19) Degree of Polynomial: https://www.codechef.com/MAY222C/problems/DPOLY

Q20) Sorted Linked List to BST:

https://www.codingninjas.com/studio/problems/sorted-linked-list-to-balanced-bst_842564

Q21) Reverse nodes in k-group:

https://leetcode.com/problems/reverse-nodes-in-k-group/?envType=study-plan-v2&envId=top-interview-150

Q22) Bandwidth of a matrix:

https://www.codechef.com/problems/BANDMATR

Q23) Reveres spiral traversal of a sparse matrix:

https://www.codechef.com/problems/MIDLE002

Q24) Remove Friends:

https://www.hackerearth.com/practice/data-structures/linked-list/singly-linked-list/practice-problems/algorithm/remove-friends-5/

Course Coordinators:

1. Dr Rahul Semwal
2. Dr Aishwarya Ukey
3. Dr Amol Bhopale
4. Mrs. Ruchira Selote

Overall Course Coordinator: Dr Richa Makhijani