HW3 Tree and Graphs

HW3-1

Descriptions:

Write an algorithm to construct the binary tree with given

- Preorder sequence and inorder sequence
- Postorder sequence and inorder sequence

Input Format:

Input consists of $1 + 4m$ line

- First line contains one number m, represent how many test dataset in the following input.
- Following input contains $4m$ lines describe m test dataset, each test dataset composed of 4 lines.
  - First line of the test dataset contains one string S, it will be or **preorder-inorder** or **postorder-inorder**.
  - Second-line of the test dataset contains one number n, represent how many nodes in the binary tree.
  - Third-line of the test dataset contains a series of numbers. $i_0, i_1, \dots, i_n$. It represent the preorder or postorder sequence of specific binary tree, The decision is based on what S is. If S is **preorder-inorder**, then it is preorder sequence. If S is **postorder-inorder**, then it is postorder sequence.
  - Forth-line of the test dataset contains a series of numbers. $j_0, j_1, \dots, j_n$. It represents the inorder sequence of specific binary tree.

Output Format:

The output should consist of m lines. For each test dataset, output one line contains a series of numbers. Each is separated by one whitespace.

- If the S in test dataset is preorder-inorder, then output the **postorder** sequence of the reconstructed binary tree.
- If the S in test dataset is postorder-inorder, then output the **preorder** sequence of the reconstructed binary tree.

Technical Specification:

- $1 \leq m \leq 10^3$
- $1 \leq n \leq 10^3$
- $1 \leq i_0, i_1, \ldots, i_n \leq n$, all i_x in $i_0, i_1, \ldots, i_n$ are unique
- $1 \leq j_0, j_1, \ldots, j_n \leq n$, all j_x in $j_0, j_1, \ldots, j_n$ are unique

Sample Input:

```
2
preorder-inorder
7
1 2 3 4 5 6 7
3 2 4 1 6 5 7
postorder-inorder
10
5 6 4 7 3 8 2 10 9 1
5 4 6 3 7 2 8 1 9 10
```

Sample Output:

```
3 4 2 6 7 5 1
1 2 3 4 5 6 7 8 9 10
```

HW3-2

Descriptions:

Rewrite **Depth-First Search** (DFS) so that it uses an adjacency matrix representation of graphs.

Input Format:

First line of the input consist of one number n, represent how many datasets in the following input.
Each dataset consists of $m + 1$ line.

- First line of the dataset contains two numbers m, t.
  - m represents the number of vertices in the given graph.
  - t is a vertex index, it represents the entry point of the DFS traversal.
- The rest of the m lines in the dataset describe a $m \times m$ matrix which describes an **undirected graph** in the adjacency matrix.

Output Format:

For each dataset, output one line.
Each line consist of n numbers. It represents the DFS visit order of the given graph.

Technical Specification:

- $1 < n < 100$
- $1 < m < 100$
- The given graph will be **undirected**.
- When there are multiple vertices available, always start from the vertex with the smallest index.

Sample Input:

```
2
6 0
0 1 0 0 1 0
1 0 1 0 1 0
0 1 0 1 0 0
```

```
0 0 1 0 1 1
1 1 0 1 0 0
0 0 0 1 0 0
7 0
0 0 1 0 1 0 0
0 0 1 1 0 0 0
1 1 0 0 0 1 0
0 1 0 0 1 1 1
1 0 0 1 0 0 1
0 0 1 1 0 0 1
0 0 0 1 1 1 0
```

Sample Output :

```
0 1 2 3 4 5
0 2 1 3 4 6 5
```

HW3-3

Descriptions:

Rewrite **Breadth-First Search** (BFS) so that it uses an adjacency matrix representation of graphs.

- When there are multiple vertices available, always start from the vertex with smallest index.

Input Format:

First line of the input consists of one number n. It represents how many datasets are in the following input.
Each dataset consists of $m + 1$ line.

- First line of the dataset contains two numbers m, t.
  - m represents the number of vertices in the given graph.
  - t is a vertex index, it represents the entry point of the BFS traversal.
- The rest of the m lines in the dataset describe a $m \times m$ matrix which describes an **undirected graph** in the adjacency matrix.

Output Format:

For each dataset, output one line. Each line consists of n numbers. It represents the BFS visit order of the given graph.

Technical Specification:

- $1 < n < 100$
- $1 < m < 100$
- The given graph will be **undirected**.
- When there are multiple vertices available, always start from the vertex with the smallest index.

Sample Input:

```
2
6 0
```

```
0 1 0 0 1 0
1 0 1 0 1 0
0 1 0 1 0 0
0 0 1 0 1 1
1 1 0 1 0 0
0 0 0 1 0 0
7 0
0 0 1 0 1 0 0
0 0 1 1 0 0 0
1 1 0 0 0 1 0
0 1 0 0 1 1 1
1 0 0 1 0 0 1
0 0 1 1 0 0 1
0 0 0 1 1 1 0
```

Sample Output:

```
0 1 4 2 3 5
0 2 4 1 5 3 6
```

HW3-4

Descriptions:

Write a C++ function that finds a minimum cost spanning tree using Kruskal's algorithm.
You may use the **union** and **find** functions from Chapter 5 and the **sort** function from Chapter 1 or the **min heap** functions from Chapter 5.

Input Format:

First line of input consists of two numbers V and E.
V represents the vertices count.
E represents the number of edges in this undirected graph.
The rest of the input contains E lines. Each line contains three numbers u_e, v_e, and c_e. It declares an undirected edge from u_e vertex to v_e vertex with a cost c_e along with this path.

Output Format:

The output consists of one number, C. It represents the sum of all edge cost in the minimum spanning tree in terms of the given graph.

Technical Specification:

- $1 < V \leq 10^6$
- $\mathrm{V} - 1 \leq E \leq \min(\frac{V(V-1)}{2}, 2 \times 10^6)$
- $1 < c_0, c_1, \ldots, c_E, < 10^6$

Sample Input:

```
7 9
0 1 28
0 5 10
1 2 16
5 4 25
6 4 24
1 6 14
3 4 22
2 3 12
3 6 18
```

Sample Output:

```
99
```

HW3-5

Descriptions:

Let T be a tree with root v. The edges of T are undirected. Edge in T has a nonnegative length. Write a C++ function to determine the length of the shortest paths from v to the remaining vertices of T. Your function should have complexity $O(n)$, where n is the number of vertices in T. Show that this is the case.

Input Format:

The input describes a tree topology in a graph way.
The first line of the input is a number V.
It represents the vertices count. Each vertex has one id, it ranges from 1 to V.
The rest of the input contains $V-1+1$ lines.

- For the first $V-1$ lines, ecah line consists of three numbers s, t and c. It describes a edge between s vertex and t vertex in this undirected graph.
- The last line of input consists of one number v, which represents the root of the tree. You are going to calculate the shortest path to each tree child node from here.

Output Format:

The output consists of V lines.
Each line i consists of two numbers i and C_i.

- i represnet the index of vertex i.
- C_i represents the cost to walk from vertex v to this vertex i.

Technical Specification:

- The graph in input has no cycle.
- Tree definition, $\forall v, u \in T$, there is only one path to connect v and u.
- $1 < V \leq 10^6$
- $1 \leq s, t \leq V$, For each edge $s \neq t$
- $1 \leq c \leq 500$

Sample Input:

```
10
1 2 110
1 3 150
1 4 100
2 5 50
2 6 80
3 7 120
3 8 150
4 9 200
4 10 400
1
```

Sample Output:

```
1 0
2 110
3 150
4 100
5 160
6 190
7 270
8 300
9 300
10 500
```

Deadline:

Regular: at 11:59 p.m., Friday Jan. 10, 2022.
Note: Handing late will lead to a perceptibly lower score.

Hand in:

1. StudentID_hw3-1.cpp to StudentID_hw3-5.cpp
2. readme.pdf includes the execution results of hw3-1 to hw3-5 and the information you would like to let TAs know, and then upload the zip file to Moodle.

The score will be deducted if the filename is not named correctly, please DO follow the rule.

If you have any questions about the homework, you can e-mail to the TAs:
Eric: p76101039@gs.ncku.edu.tw
Kevin: p76101479@gs.ncku.edu.tw

*Discussion is encouraged, but DO NOT share your code to your classmate. TA will check the plagiarism issue. If plagiarism is found, both students will get 0 score in this assignment.