Adil Tanveer
CS 430: Prof. Wan's Introduction to Algorithms
October 23, 2023

CS430 Project Instructions

Evaluation Scheme

- Illustration of design and implementation [**15 %**]
  - Group must briefly illustrate the design and implementation of the algorithm
- Evaluation Report [**15 %**]
  - Group must report the evaluation of the algorithm with test data
- Correctness [**70 %**]
  - Code will be executed against pre-defined test cases

Test Cases:

Each test case will be input in the form of a text file

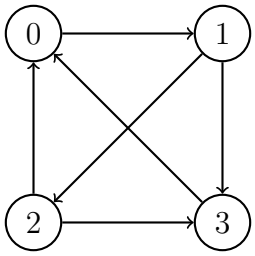
Input format

- First line - n (number of nodes)
- Second line - m (number of edges)
- Next m lines - u<space>v, where $(u, v) \in E$ and $0 \leq u, v \leq n-1$ [**Refer below for example**]
- Last line - s (Source node for DFS), where $0 \leq s \leq n-1$
- Only directed graphs will be given as input

Output format

- First n lines - Pre-order of DFS
- Next n lines - Post-order of DFS [**Refer below for example**]

Example Test Case



Sample Input

Below is the input for the above graph according to the format given above

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

Sample Output

Below is the sample output for the above graph according to the format given above

```
0
1
3
2
3
2
1
0
```

Details :

- The neighbourhood of every vertex during `DFS` must be uniquely ordered according to the input
  - `DFS(u)` must visit v before w if and only if the edge (u, v) precedes the edge (u, w) in the input
  - For example, in the example given above, `DFS(1)` must visit `3` before `2` since $(1, 3)$ precedes $(1, 2)$ in the input
- The DFS pre-order of the above graph starting from 0 is 0 1 3 2
- The DFS post-order of the above graph starting from 0 is 3 2 1 0
- This explains the output given above. **Note that the above output is unique**
  - By the ordering constraint given above on the neighbourhood of every vertex, `DFS(s)` will have a unique traversal on any input graph

Test Case Execution

- The input file name will be given as a command line argument with the option `-i` or `--input` [**See below for example**]
- Assume the file exists in the current directory where the program is located
- Output file must be named `<input-file-name>_output.txt` [**See below for example**]
- Output file must be created in the current directory where the program is located

See the following commands for python (but could be extrapolated for any other language) :

```
$ python3 dfs.py -i test_1.txt
$ python3 dfs.py --input test_1.txt
```

In this case, the output file must be created in the current directory with the name `test_1_output.txt`

Evaluation requirements:

- No code changes must be made to the core logic post submission of project except as required to adapt to the input-output requirements
  - It is suggested to write an input-wrapper that reads the input file, converts it into the format used by the code and invokes the actual code of `DFS`
  - Also, another output wrapper to generate output might be required
    - Minor Changes to accommodate for outputting both pre-order and post-order are allowed as long as the core logic of the algorithm does not change
  - There is no need to modify any output to `stdout` or any other test cases that already exist within the code
- The algorithm must be *linear*-time complexity

Procedure for Scheduling a demo

- A spreadsheet will be shared with timeslots
- Every group needs to book any one timeslot
- During the timeslot:
  - The group will present an illustration of code and design
  - The group will present self-evaluation of the code on test cases
  - The code will be executed by the TA against predefined test cases