**Memo**

|  |  |  |
| --- | --- | --- |
|  | Time | Space |
| Germ: indegree(self) | O(1) | O(n) |
| PriorityQue: push(self, germ) | O(1) | O(n) |
| PriorityQue: deleteMin(self) | O(n) | O(n) |
| main() | O(n log n) | O(n) |

*Still unable to complete most of the test cases, however every test case that I tried and that Professor Elenbogen gave me worked, so I’m not sure if it’s a minor code issue regarding how the program is initialized or because its in python, or if I have to restart it..*

**Source Code Documentation**

# Srinivas Simhan

# CIS 350 - Elenbogen: Winter 2018

# Program 3

# Due: 3/27/18

class Germ:

def \_\_init\_\_(self):

self.children = []

self.ancestors = set()

self.index = None

"""

PRE:

- None

POST:

- returns the number of ancestors associated with the object

DESCRIPTION:

- method to find the number of ancestors of the object associated with the of Germ

"""

def indegree(self):

return len(self.ancestors)

class PriorityQue:

def \_\_init\_\_(self):

self.list = []

self.length = 0

"""

PRE:

- takes in the germ object

POST:

- appends the germ object onto the end of the priority list

- increases the length of the priority list by 1

DESCRIPTION:

- the purpose of this method is to apend the germ object onto the priority list and increases the length dynamically

"""

def push(self, germ):

self.list.append(germ)

self.length += 1

"""

PRE:

- None

POST:

- in a for loop, checks if the degrees of the germ being tested are less than the minGerm degrees, and reset it accordingly

- else it checks if the germ index is less than the min germ index and reset it accordingly

- then removed the knownMinGerm and decreases the length of the list by 1 (the absence of that value)

DESCRIPTION:

- uses the number of ancestors and the index value of the germ to remove the minimum valued germ

"""

def deleteMin(self):

knownMinGerm = self.list[0]

for germ in self.list:

if germ.indegree() < knownMinGerm.indegree():

knownMinGerm = germ

else:

if germ.index < knownMinGerm.index:

knownMinGerm = germ

self.list.remove(knownMinGerm)

self.length -= 1

return knownMinGerm

"""

PRE:

- input for germList

POST:

- print them out in topological order

DESCRIPTION:

- prints out the topologicalIndexes order without the new line issue when it gets posted

"""

def main():

totalGermCount = int(input())

if totalGermCount > 2000:

exit()

elif totalGermCount <= 0:

exit()

germList = []

for i in range(totalGermCount):

germ = Germ()

germ.index = i + 1

germList.append(germ)

priorityQueue = PriorityQue()

for germ in germList:

priorityQueue.push(germ)

childIndexes = input().split() # list of ints as strings

childIndexes.pop() # discard the ending zero

for childIndex in childIndexes:

childGerm = germList[int(childIndex) - 1]

childGerm.ancestors.add(germ)

germ.children.append(childGerm)

topological = []

while len(priorityQueue.list) > 0:

# priorityQueue.sort(key=lambda germ : (len(germ.ancestors), germ.index))

# removedGerm = priorityQueue.pop(0)

removedGerm = priorityQueue.deleteMin()

topological.append(removedGerm)

for childGerm in removedGerm.children:

childGerm.ancestors.add(removedGerm)

childGerm.ancestors.update(removedGerm.ancestors)

topologicalIndexes = []

topological.sort(key=lambda germ : (len(germ.ancestors), germ.index))

for germ in topological:

topologicalIndexes.append(germ.index)

print(\*topologicalIndexes, end='')

main()