import math

import random

def create\_graph(branch\_fact, depth, min\_possible\_damage, max\_possible\_damage):

graph = []

for i in range(depth):

for j in range(branch\_fact\*\*i):

graph.append(0)

for i in range(branch\_fact\*\*depth):

graph.append(random.randint(min\_possible\_damage, max\_possible\_damage))

return graph

def minimax(index, level, alpha, beta):

global graph, compared\_nodes, depth, branch\_fact

if level == depth:

compared\_nodes = compared\_nodes + 1

else:

max\_damage = -math.inf

min\_damage = math.inf

for i in range(branch\_fact):

minimax(index\*branch\_fact+i+1, level+1, alpha, beta)

damage = graph[index\*branch\_fact+i+1]

if level % 2 == 0:

max\_damage = damage if damage > max\_damage else max\_damage

alpha = max\_damage if max\_damage > alpha else alpha

else:

min\_damage = damage if damage < min\_damage else min\_damage

beta = min\_damage if min\_damage < beta else beta

if alpha >= beta:

break

graph[index] = max\_damage if level % 2 == 0 else min\_damage

def print\_output(graph, depth, branch\_fact, initial\_hp, damaged\_hp, compared\_nodes):

print("\nOutput:")

print(f"1. Depth and Branches ratio is {depth}:{branch\_fact}")

strng = "2. Terminal States (leaf node values) are "

for i in range(len(graph)-branch\_fact\*\*depth, len(graph)):

strng += str(graph[i])

if i != len(graph)-1:

strng += ','

print(strng)

print("3. Left life(HP) of the defender after maximum damage caused by the attacker is",

initial\_hp-damaged\_hp)

print("4. After Alpha-Beta Pruning Leaf Node Comparisons", compared\_nodes)

id = input("1. Enter your student id: ")

min\_possible\_damage, max\_possible\_damage = [int(i) for i in input(

"2. Minimum and Maximum value for the range of negative HP: ").split()]

depth = int(id[0])\*2

branch\_fact = int(id[2])

initial\_hp = int(id[-1]+id[-2])

# Creating array to store the graph

graph = create\_graph(branch\_fact, depth,

min\_possible\_damage, max\_possible\_damage)

compared\_nodes = 0

minimax(0, 0, -math.inf, math.inf)

damaged\_hp = graph[0]

print\_output(graph, depth, branch\_fact, initial\_hp, damaged\_hp, compared\_nodes)