**Input as a file:**

(input.txt)

1

3

1 20

**The Code:**

import math

import random

import numpy as np

# Getting the inputs from file

file = open('input.txt', 'r')

# Getting depth and branch from input

depth = int(file.readline())\*2

branch = int(file.readline())

leaf\_node\_count = pow(branch, depth)

# Getting the range

range\_line = file.readline()

start = int((range\_line.split())[0])

end = int((range\_line.split())[1])

# Generating the tree

branches = 0

for i in range(depth+1):

branches += pow(branch, i)

tree = np.zeros((branches, branches), int)

limit = 0

for d in range(depth):

limit += pow(branch, d)

temp = 0

for i in range(limit):

for j in range(branch):

temp += 1

tree[i][temp] = 1

leaf\_nodes = np.zeros(branches, int)

for s in range((branches - leaf\_node\_count), branches):

leaf\_nodes[s] = random.randint(start, end)

# Number of comparison in minimax

comparison1 = 0

# Minimax Function

def minimax(position, depth, maximizingPlayer):

global comparison1

if depth == 0:

return leaf\_nodes[position]

if maximizingPlayer is True:

maxEval = -math.inf

for i in range(branches):

if tree[position][i] == 1:

eval = minimax(i, depth-1, False)

maxEval = max(maxEval, eval)

return maxEval

else:

minEval = math.inf

for j in range(branches):

if tree[position][j] == 1:

eval = minimax(j, depth - 1, True)

comparison1 += 1

minEval = min(minEval, eval)

return minEval

minimax\_output = minimax(0, 2, True)

# Number of comparison in Alpha-beta pruning

comparison2 = 0

# Alpha-Beta Pruning Function

def alpha\_beta\_pruning(position, depth, alpha, beta, maximizingPlayer):

global comparison2

if depth == 0:

return leaf\_nodes[position]

if maximizingPlayer is True:

maxEval = -math.inf

for i in range(branches):

if tree[position][i] == 1:

eval = alpha\_beta\_pruning(i, depth - 1, alpha, beta, False)

maxEval = max(maxEval, eval)

alpha = max(alpha, eval)

if beta <= alpha:

break

return maxEval

else:

minEval = math.inf

for j in range(branches):

if tree[position][j] == 1:

eval = alpha\_beta\_pruning(j, depth - 1, alpha, beta, True)

comparison2 += 1

minEval = min(minEval, eval)

beta = min(beta, eval)

if beta <= alpha:

break

return minEval

abpruning\_output = alpha\_beta\_pruning(0, 2, -math.inf, math.inf, True)

# Printing the outputs

print("Depth:", depth)

print("Branch:", branch)

print("Terminal States(Leaf Nodes):", leaf\_node\_count)

print("Maximum amount:", abpruning\_output)

print("Comparisons:", comparison1)

print("Comparisons:", comparison2)