Lab 03

Krity Haque Charu

```
import math
import random as ra

id="19101173"
depth=int(id[0])*2

hp=int(id[-2:][::-1])
print("HP:",hp)
branches = int(id[2])
print("Depth and Branch ratio ", depth , ":" ,branches)

print("Enter Range: ")
min, max = [int(y) for y in input().split()]

num_of_bullets= (branches**depth)

count= 1
bullets = []


while count <= num_of_bullets :
    bullets.append(ra.randrange(min,max+1))
    count += 1
print("Terminal states :" ,bullets)


len_bullets=len(bullets)
print(len_bullets)

def AlphaBeta(node, depth, alpha, beta, maxP, bullets, pruning):

  if depth==0:
    return bullets[node], pruning

  if maxP:
    best= -math.inf
    i=0
    while i< branches:
      val,pruning = AlphaBeta(node*branches+i,depth-
1,alpha,beta,False,bullets,pruning)
      i = i + 1

      #pruning += 1
      #best = max(best,val)
      #alpha = max(alpha,val)
      if val>best:
        best=val
      else:
        best=best
      if alpha>best:
        alpha=alpha
      else:
        alpha=best

      if beta<=alpha:
        pruning += 1
        #explored[node*branches+i]=0
        break
    return best,pruning

  else:
    i = 0
    best = math.inf
    while i < branches:
      val,pruning = AlphaBeta(node*branches+i,depth-
1,alpha,beta,True,bullets,pruning)

      i = i + 1

      #best = min(best,val)
      #beta = min(beta,val)
      if val<best:
        best=val
      else:
        best=best
      if beta<val:
        beta=beta
      else:
        beta=val

      if beta<=alpha:
        pruning += 1
        break
    return best,pruning

optimal_val,pruning=AlphaBeta(0,depth,-math.inf,math.inf,True,bullets,0)
left_hp=hp-optimal_val
left_without_pruning=num_of_bullets-pruning
print("Optimal Value ", optimal_val)
print("Left life(HP)", left_hp)
```

```python
print("After Alpha-Beta Pruning Leaf Node Comparisons ",left_without_pruning-1)
```