**Universidad Nacional San Agustin de Arequipa**

Facultad de Ingenierias de Produccion y Servicios

# Escuela Profesional de Ingenieria de Sistemas

*Inteligencia Artificial*

Alumno:

Fuentes Paredes Nelson Alejandro

Mayo 2020

[ ]:

## 1 Clase Nodo

[1]:
```
class Node:

    def __init__(self, n=2):
        self.n = n
        self.data = []
        self.full = False

    def isFull(self):
        return len(self.data) == self.n

    def insert(self, value):
        if len(self.data) == 0:
            self.data.append(value)
            return None
        last_node = self.data[len(self.data) - 1]

        if isinstance(last_node, Node):
            value = last_node.insert(value)
            if value is None:
                return None
        if not self.isFull() and value is not None:
            self.data.append(value)
            return None
        else:
            new_Node = Node(self.n)
            new_Node.insert(value)
            return new_Node

    def __str__(self):

        strings = ""
        for dat in self.data:
            strings = strings + ", " + str(dat)
        return "[" + strings + "]"
```

## 2 Clare Árbol Alfa Beta

[2]:
```
class AlphaBethaTree:

    def __init__(self, n):
        self.__n__ = n
```

```
        self.__root__ = Node(n)

    def insert(self, value):
        new_Node: Optional[Node] = self.__root__.insert(value)
        if new_Node is not None:
            n = Node(self.__n__)
            n.data.append(self.__root__)
            n.data.append(new_Node)


            self.__root__ = n

    def pruning(self):
        return self.__pruning__(self.__root__ ,-1000, 1000, 0)

    # Algoritmo
    def __pruning__(self, node, alpha, betha , depth):
        if not isinstance(node, Node):
            return node
        if depth %2 == 0:
            for child in node.data:
                alpha = max(alpha, self.__pruning__(child, alpha, betha,␣
↪depth+1))
                if betha<=alpha:
                    break
            return alpha
        else:
            for child in node.data:
                betha = min(betha, self.__pruning__(child, alpha, betha,␣
↪depth+1))
                if betha<=alpha:
                    break
            return betha


    def __str__(self):
        return str(self.__root__)
```

## 3 Ejecución

[9]:
```
import random

def main():
    t = AlphaBethaTree(3)
    for i in range(0, 18):
        t.insert(random.randint(-100, 100))
```

```
    print('Árbol\t\t:\t', t)

    print('Resultado\t:\t',t.pruning())

if __name__ == '__main__':
    for i in range(1, 10):
        main()
        print('\n\n')
```

```
Árbol           :       [, [, [, -68, 44, 54], [, 63, -42, 66], [, 96, 97,
42]], [, [, 93, 74, -70], [, 76, -81, -40], [, -63, -60, 9]]]
Resultado       :       54



Árbol           :       [, [, [, -47, 79, 98], [, -39, 41, 17], [, 92, -49,
-48]], [, [, -16, -47, 48], [, -11, -45, -81], [, -38, 96, -64]]]
Resultado       :       41



Árbol           :       [, [, [, -78, -86, 89], [, -54, 42, 71], [, -40, -89,
-10]], [, [, -30, -99, 56], [, -65, 85, 33], [, -94, -40, 25]]]
Resultado       :       25



Árbol           :       [, [, [, -86, 12, 86], [, 32, -14, -22], [, -27, 99,
46]], [, [, 11, -38, 64], [, -45, -47, -90], [, -80, -80, -25]]]
Resultado       :       32



Árbol           :       [, [, [, 18, 36, -11], [, -22, -88, -45], [, -82, -40,
-19]], [, [, 50, 18, 7], [, -100, 13, -27], [, -77, 30, -64]]]
Resultado       :       13



Árbol           :       [, [, [, -54, -56, -42], [, 2, -69, 7], [, 34, -52,
-63]], [, [, 6, 64, 36], [, 91, 69, 8], [, -8, -66, 62]]]
Resultado       :       62



Árbol           :       [, [, [, 65, 19, -82], [, -72, -62, -53], [, 50, 64,
-92]], [, [, -88, -42, 12], [, -70, -55, 20], [, 70, -100, -49]]]
```

```
Resultado       :        12


Árbol           :        [, [, [, -32, -35, 13], [, -72, 33, -41], [, 20, -12,
-76]], [, [, -13, 53, 74], [, -22, 83, -38], [, -18, 13, 21]]]
Resultado       :        21


Árbol           :        [, [, [, -49, -50, -24], [, 92, 2, -65], [, -53, -60,
17]], [, [, -47, 9, 94], [, 90, 76, -9], [, 86, 95, -70]]]
Resultado       :        90
```

[ ]: