```
from search import *
class Problema(Problem):

    def __init__(self, initial=(0, 0, 0), goal=(50, 60, 70)):
        super.__init__(initial,goal)
    def actions(self,state):
        acciones = []
            if self.puedohacerlo(state,”accion1”):
                acciones.append(“accion1”)
        return acciones
    def accion1(self,state):
        state[1] += 9
    def accion2(self,state):

    def puedohacerlo(self,state,accion):
        puedo = True
        if accion == “accion1”:
            if state[0]>5 || state[1] == 3 && state[2] ¡=4:
                puedo = False
        if acción == “accion2”:
            bla
        return puedo

    def result(self,state,accion):
        new_state = list(state)
        if accion = “accion1”:
            self.accion1(new_state)
        if accion = “accion2”:
            self.accion2(new_state)
```

```
                return tuple(new_state)


        def goal_test(self,state):

                return state == self.goal


        def h(self,node):

                result node.state[1] + node.state[2]


        def __name__=’__main__’:

                p = Problema()

                print("Resultados con A*:", astar_search(myc).solution())


        print("Resultados con
breadth_first_tree_search:",breadth_first_tree_search(myc).solution())


     print("Resultados con breadth_first_graph_search:",
breadth_first_graph_search(myc).solution())


     print("Resultados con depth_first_graph_search:", depth_first_graph_search(myc).solution())


print("Resultados con uniform_cost_search:", uniform_cost_search(myc).solution())

print("REsultados con A*:", astar_search(myc).solution())
```