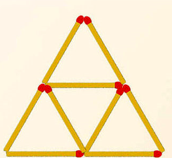
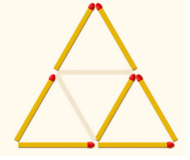
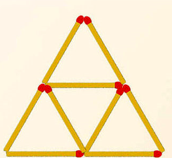
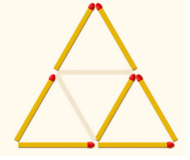
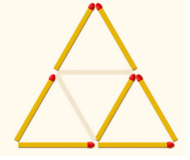
Estado Inicial

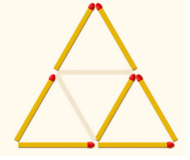


Estado final

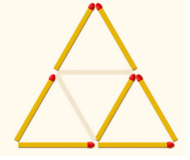


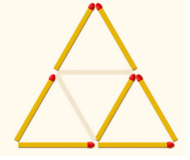


1

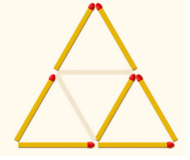
0

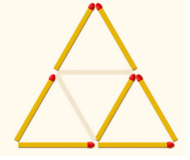
1

0

3

2

6

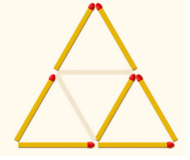
7

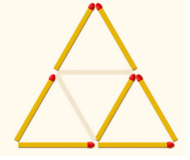
4

7

6

3

8

5

8

5

012345678 0135678

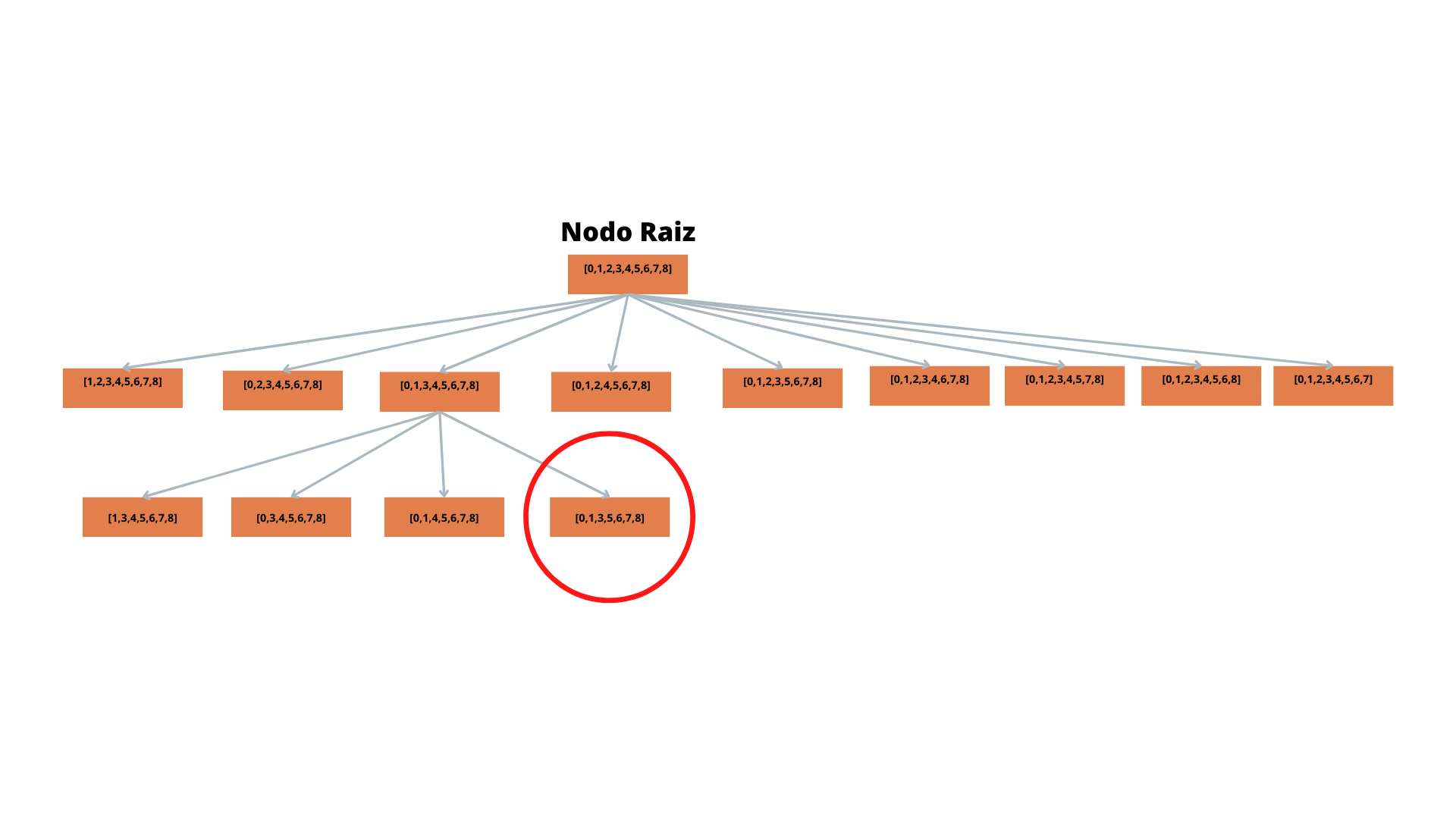
estado inicial es el nodo raíz del cual saldrán otros 9 estados o hijos

012345678

Estado final representado en un nodo

0135678

A continuación, se representarán los nodos.

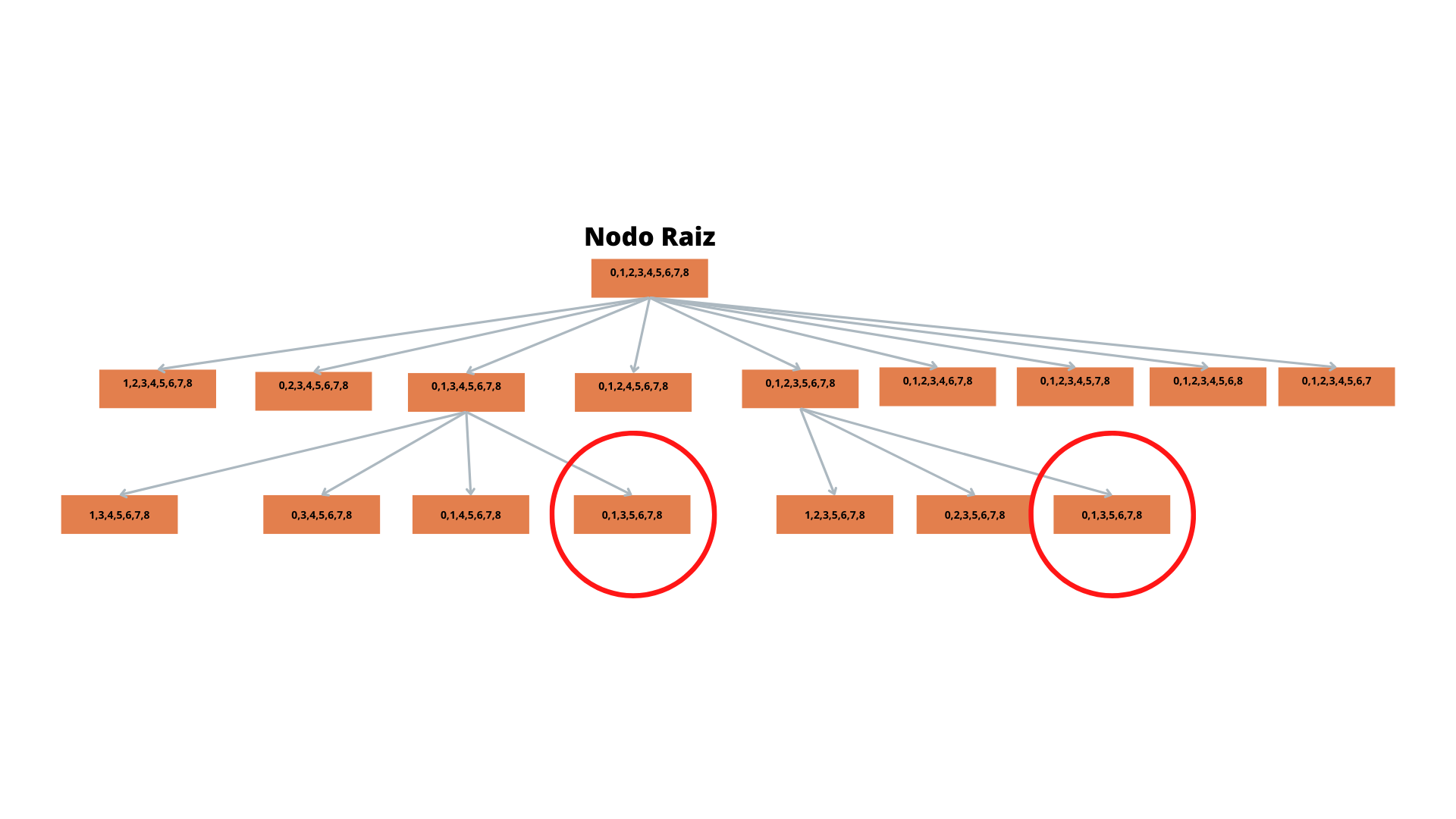


El nodo enmarcado con rojo es el estado final que debemos alcanzar, así que replanteemos las condiciones del problema:

-Solo podemos quitar un elemento a la vez

-en este caso no aplicamos costos al problema

-solo tenemos un estado inicial y uno final



El nodo enmarcado con rojo es el estado final que debemos alcanzar, así que replanteemos las condiciones del problema:

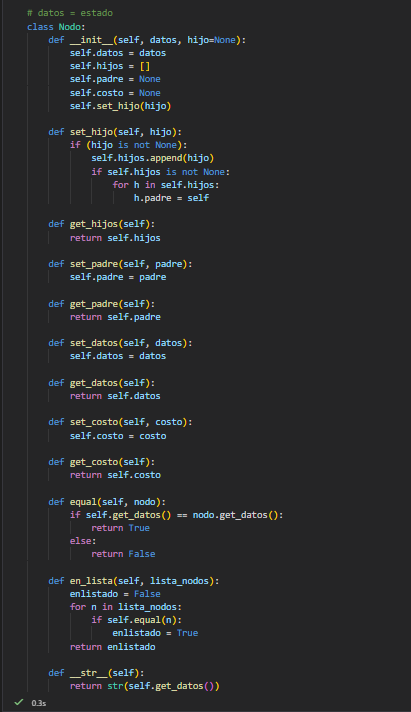
-Solo podemos quitar un elemento a la vez

-en este caso no aplicamos costos al problema

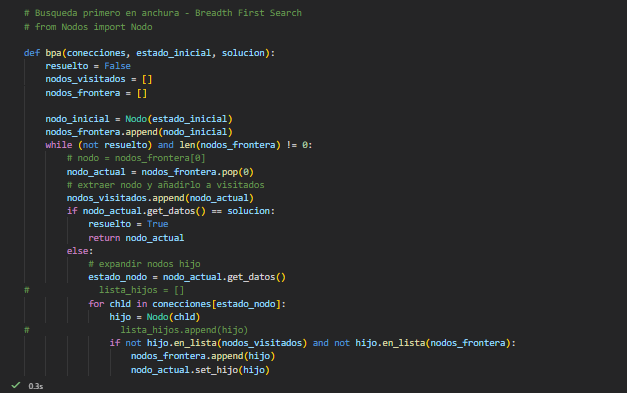
-solo tenemos un estado inicial y uno final

Representación en código

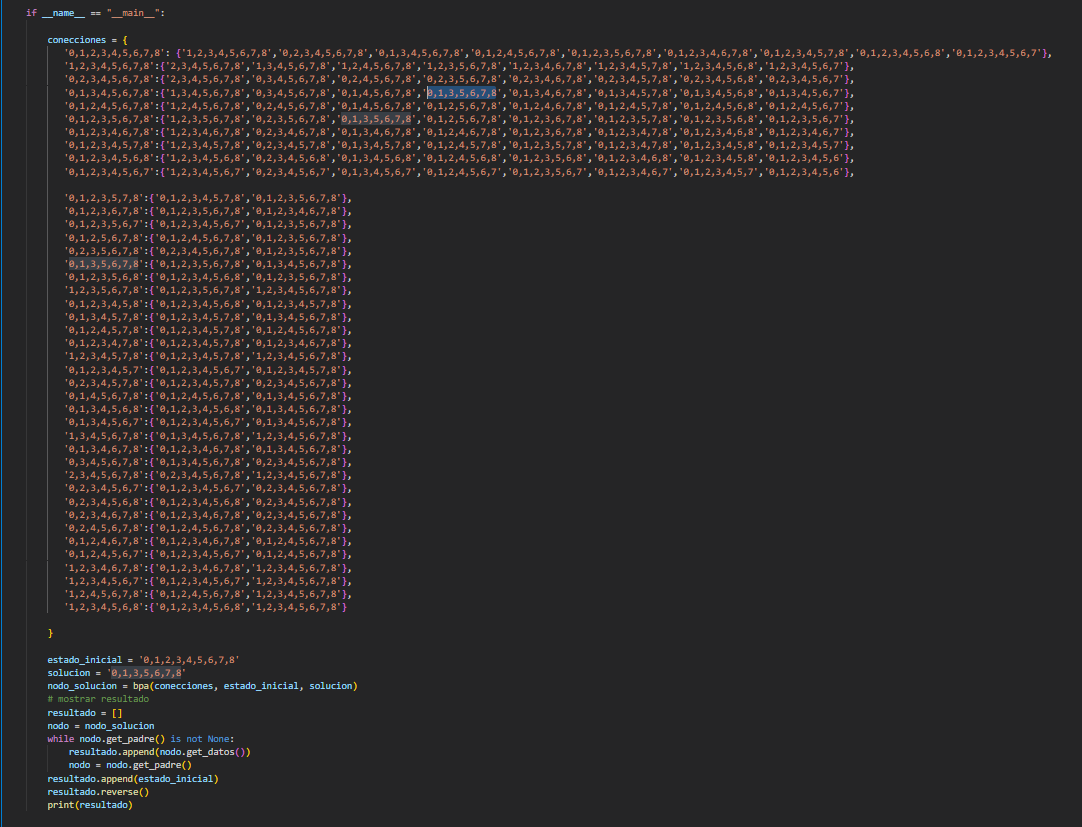
Creación de la clase nodo con sus respectivos geters y setters



Creación de la función Búsqueda por anchura representada (bpa)



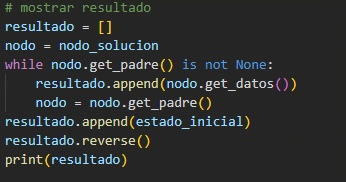
Creamos las conexiones que tendran los nodos



Mandamos un estado inicial , uno final y la conexiones a la función bpa(búsqueda por anchura)



Mostramos el resultado



Resultado

