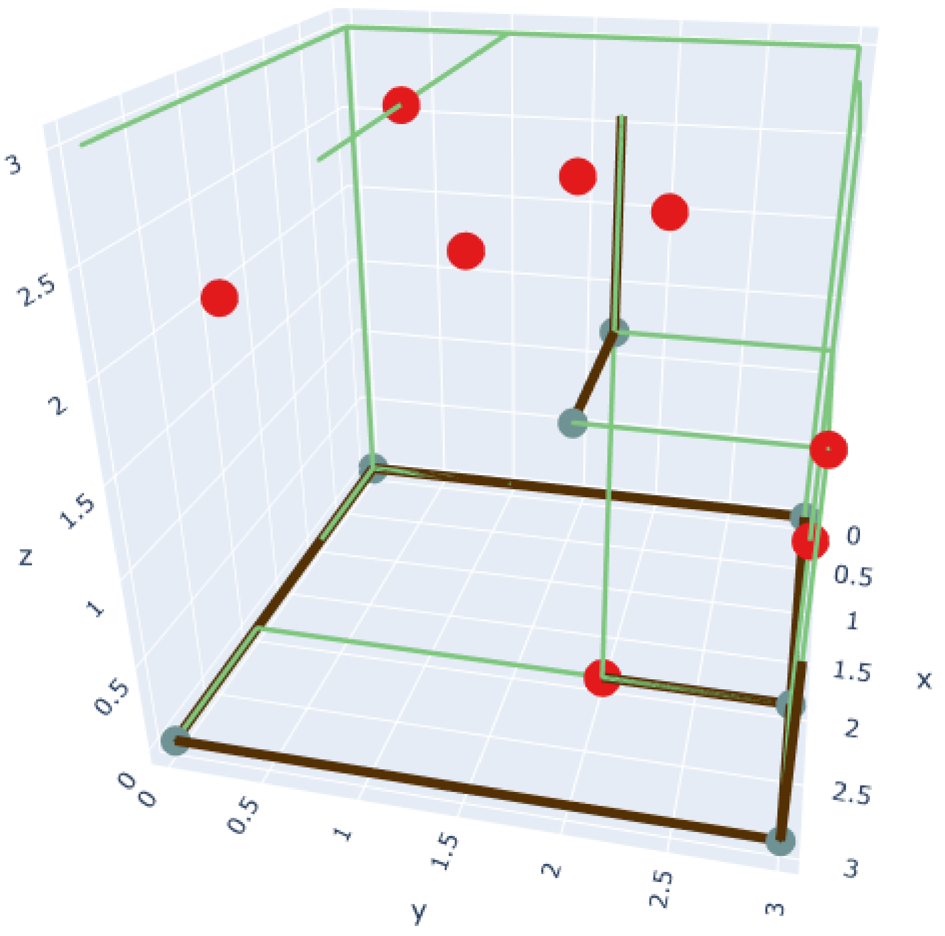
最终结果预览：



Notebook for house aera, by Guo Xufeng

Import data from xlsx

using pandas.read_excel() , and using openpyxl

In [118]:

```
import pandas as pd
import json
import matplotlib
import matplotlib.pyplot as plt
import plotly.graph_objs as go
from plotly.offline import plot
from IPython.core.display import HTML
import plotly.offline as offline
offline.init_notebook_mode(connected=True)
import numpy as np

df = pd.read_excel('./data/data.xlsx')
```

Print basic information

In [13]:

```
print('row number:  '+str(len(df.index.values)))
print('col number:  '+str(len(df.columns.values)))
print('************column vules***************')
print(df.columns.values)
print('************column vules***************')
```

```
row number:  636
col number:  7
************column vules***************
['时间戳' '测试名称' '准考证号' '任务ID' '任务名称' '该时间戳下记录的作答结果' '测试
来源']
************column vules***************
```

divide frame into different students

using groupby()

In [19]:

```
group = df.groupby( '准考证号')
group_list = list(group)
```

the student number is 59

In [28]:

```
print('the student number is:  ',len(group_list))
```

```
the student number is:   59
```

An exmaple frome the group_list

In [33]:

```
df_ex = group_list[0][1]
df_ex
```

Out[33]:

	时间戳	测试名称	准考证号	任务ID	任务名称	该时间戳下记录的作答结果	测试来源
347	2020-09-21T14:23:59+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[0,0,0,0,0,0]}}	bnuadmin
358	2020-09-21T14:24:13+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,0,0,0,0,0]}}	bnuadmin
359	2020-09-21T14:24:15+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,1,0,0,0,0]}}	bnuadmin
363	2020-09-21T14:24:17+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,1,1,0,0,0]}}	bnuadmin
365	2020-09-21T14:24:19+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,1,1,0,1,0]}}	bnuadmin
367	2020-09-21T14:24:20+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,1,1,1,1,0]}}	bnuadmin
368	2020-09-21T14:24:22+08:00	bnuadmin/小学试卷二	1031	1	房屋面积	{"canvas":{"swap":[1,1,1,1,1,1]}}	bnuadmin

Reshape data into a list

In [38]:

```
list_seq = []
df_ex.iloc[:, 5].values[0]
```

Out[38]:

```
'{"canvas":{"swap":[0,0,0,0,0,0]}}'
```

In [57]:

```
def df_to_list(df_ex):
    list_ex = []
    for i in range(len(df_ex)):
        df_ex_dic = json.loads(df_ex.iloc[:, 5].values[i])
        list_ex.append(df_ex_dic['canvas']['swap'])
    return list_ex

def reshape_group(group_list):
    dic_res = {}
    for i in range(len(group_list)):
        dic_res[i] = df_to_list(group_list[i][1])
    return dic_res
```

In [45]:

```
dic_res = reshape_group(group_list)
```

In [86]:

```
len(dic_res)
dic_res[0]
```

Out[86]:

```
[[0, 0, 0, 0, 0, 0],
 [1, 0, 0, 0, 0, 0],
 [1, 1, 0, 0, 0, 0],
 [1, 1, 1, 0, 0, 0],
 [1, 1, 1, 0, 1, 0],
 [1, 1, 1, 1, 1, 0],
 [1, 1, 1, 1, 1, 1]]
```

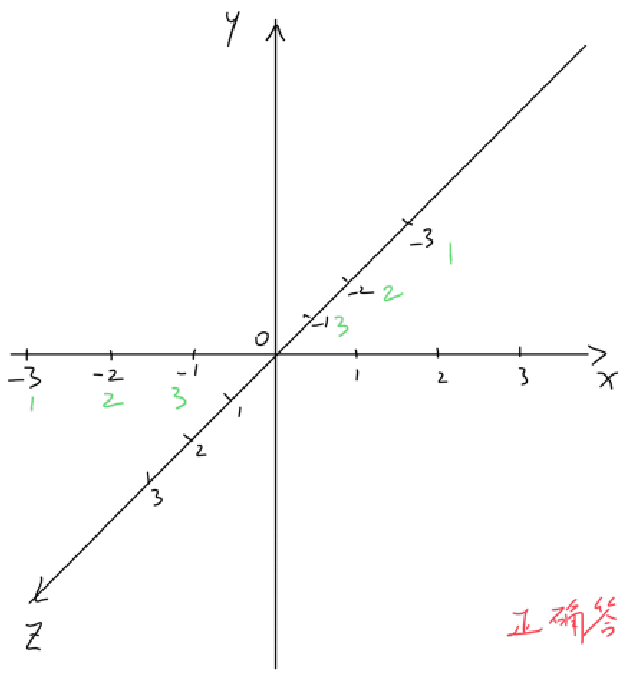
Plot

In [224]:

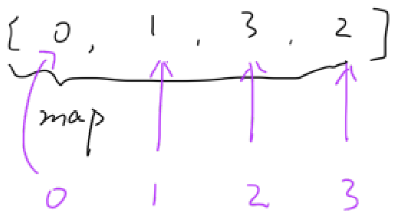
```
def plot_all(point_list = None, vector_list = None):

    data = point_list+vector_list
    layout = go.Layout(margin = dict( l = 0,
                                      r = 0,
                                      b = 0,
                                      t = 0)
                      )
    fig = go.Figure(data=data,layout=layout)
    plot(fig,filename="vector.html",auto_open=False,image='png',image_height=800,image_width=1500)
    offline.iplot(fig)
```

Bit to coordinate

首先采用格雷码对数据集的6位二进制数进行“2位+2位+2位”模式的编码



0	0	0	0
1	0	1	1
2	1	1	3
3	1	0	2



正确答案:			格雷码			坐标码.		
10	10	11	2	2	3	3	3	2
10	11	10	2	3	2	3	2	3
11	10	01	3	2	1	2	3	1
11	11	00	3	3	0	2	2	0
00	11	11	0	3	3	0	2	2
01	11	01	1	3	1	1	2	1
11	00	11	3	0	3	2	0	2
11	01	10	3	1	2	2	1	3
01	01	11	1	1	3	1	1	2

In [109]:

```
# def bit_to_coordinate(list_of_bits):
#     map_to_num = [0, 1, 3, 2, 7, 6, 4, 5]
#     x = map_to_num[list_of_bits[0]*4 + list_of_bits[1]*2 +list_of_bits[2]*1]
#     y = map_to_num[list_of_bits[3]*4 + list_of_bits[4]*2 +list_of_bits[5]*1]
#     return tuple((x, y))
def bit_to_coordinate(list_of_bits):
    map_to_num = [0, 1, 3, 2]
    x = map_to_num[list_of_bits[0]*2 + list_of_bits[1]*1]
    y = map_to_num[list_of_bits[2]*2 + list_of_bits[3]*1]
    z = map_to_num[list_of_bits[4]*2 + list_of_bits[5]*1]
    return [x, y, z]
```

In [110]:

```
dic_res[0]
```

Out[110]:

```
[[0, 0, 0, 0, 0, 0],
 [1, 0, 0, 0, 0, 0],
 [1, 1, 0, 0, 0, 0],
 [1, 1, 1, 0, 0, 0],
 [1, 1, 1, 0, 1, 0],
 [1, 1, 1, 1, 1, 0],
 [1, 1, 1, 1, 1, 1]]
```

In [111]:

```
bit_to_coordinate(dic_res[0][2])
```

Out[111]:

```
[2, 0, 0]
```

In [123]:

```
def bits_to_coordinate(dic_res):
    dic_coordinate = {}
    for i in range(len(dic_res)):
        tmp_list = []
        for j in range(len(dic_res[i])):
            coor = bit_to_coordinate(dic_res[i][j])
            tmp_list.append(str(coor[0])+str(coor[1])+str(coor[2]))
        dic_coordinate[i] = tmp_list
    return dic_coordinate

dic_coordinate = bits_to_coordinate(dic_res)
```

Lines between nodes

In [127]:

```
def coordinate_to_node(dic_coordinate):
    dic_line = {}
    for i in range(len(dic_coordinate)):
        tmp_list = []
        for j in range(len(dic_coordinate[i])-1):
            tmp_list.append(dic_coordinate[i][j] + dic_coordinate[i][j+1])
        dic_line[i] = tmp_list
    return dic_line

dic_line = coordinate_to_node(dic_coordinate)
```

In [144]:

```
lines_list = []
for i in range(len(dic_line)):
    lines_list+=dic_line[i]
len(lines_list)
```

Out[144]:

```
577
```

In [136]:

```
nodes_list = []
for i in range(len(dic_coordinate)):
    nodes_list+=dic_coordinate[i]
nodes_list
len(nodes_list)
```

Out[136]:

```
636
```

In [207]:

```
nodes_res = {}
for key in nodes_list:
    nodes_res[key] = nodes_res.get(key, 0) + 1
print(nodes_res)
```

```
{'000': 91, '300': 34, '200': 11, '230': 28, '233': 20, '223': 20, '222': 45, '100':
9, '001': 9, '030': 33, '033': 10, '133': 8, '132': 3, '122': 3, '330': 38, '320':
8, '323': 9, '220': 21, '221': 7, '301': 4, '331': 20, '332': 14, '231': 11, '232':
14, '123': 3, '113': 2, '112': 3, '111': 6, '101': 1, '020': 5, '321': 2, '031': 4,
'333': 6, '003': 20, '002': 7, '303': 4, '103': 1, '013': 8, '032': 4, '023': 3, '02
2': 2, '322': 25, '010': 8, '011': 3, '311': 1, '310': 3, '210': 4, '213': 4, '212':
5, '201': 2, '313': 7, '203': 4, '211': 4, '130': 4, '131': 1, '202': 5, '110': 3,
'012': 1, '302': 1}
```

In [146]:

```
lines_list_no_dir = []
for line in lines_list:
    tmp =''
    if line[0:3] > line[3:6]:
        tmp = line[3:6] + line[0:3]
    else:
        tmp = line[0:3] + line[3:6]
    lines_list_no_dir.append(tmp)
# print(len(lines_list_no_dir))
# print(lines_list_no_dir)
lines_res = {}
for key in lines_list_no_dir:
    lines_res[key] = lines_res.get(key, 0) + 1
lines_res
len(lines_res)
```

Out[146]:

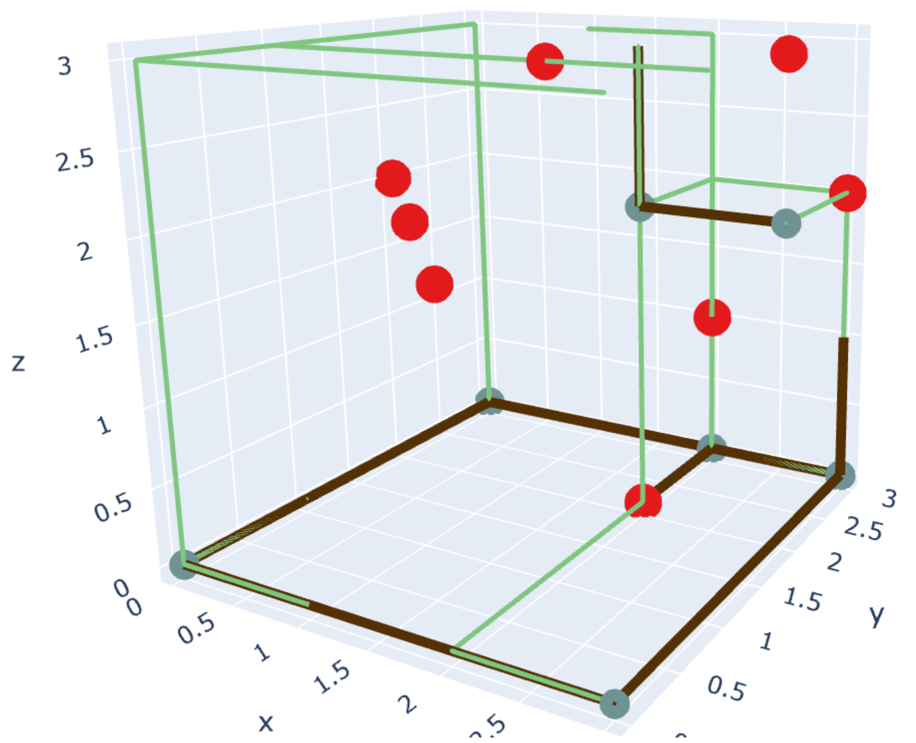
```
132
```

In [222]:

```
def create_vectors(lines_res):
    vector_list = []
    for line in lines_res:
        x1 = line[0]
        x2 = line[3]
        y1 = line[1]
        y2 = line[4]
        z1 = line[2]
        z2 = line[5]
        num = lines_res[line]
        if_plot = False
        if num <5:
            if_plot = False
            size = 1
            color = "rgb(208, 162, 159)"
        elif num < 15:
            if_plot = True
            size = 5
            color = "rgb(127, 198, 127)"
        else:
            if_plot = True
            size = 10
            color = "rgb(84,48,5)"

        if if_plot:
            vector = go.Scatter3d( x = [x1,x2],
                                  y = [y1,y2],
                                  z = [z1,z2],
                                  marker = dict( size = 1,
                                                 color = color),
                                  line = dict( color = color,
                                               width = size)
                                 )
            vector_list.append(vector)
    return vector_list
vector_list = create_vectors(lines_res)
def create_points(nodes_res):
    node_list = []
    x_red_list =[]
    y_red_list =[]
    z_red_list =[]

    x_green_list =[]
    y_green_list =[]
    z_green_list =[]
    for node in nodes_res:
        x = node[0]
        y = node[1]
        z = node[2]
        num = nodes_res[node]
        if node == '332' or node == '323' or node == '231' or node == '220' or node == '022' or r
            x_red_list.append(int(x))
            y_red_list.append(int(y))
            z_red_list.append(int(z))
        else:
            if num >20:
                x_green_list.append(int(x))
                y_green_list.append(int(y))
                z_green_list.append(int(z))
    red_points = go.Scatter3d( x = x_red_list,
                           y = y_red_list,
                           z = z_red_list,
                           mode = 'markers',
                           marker = dict( size = 10,
                                          color = "rgb(227,26,28)")
                         )
    green_points = go.Scatter3d( x = x_green_list,
                           y = y_green_list,
                           z = z_green_list,
                           mode = 'markers',
                           marker = dict( size = 8,
                                          color = "rgb(111, 146, 148)")
                         )
    return [red_points,green_points]

node_list = create_points(nodes_res)
```

In [225]:

```
plot_all(node_list, vector_list)
```



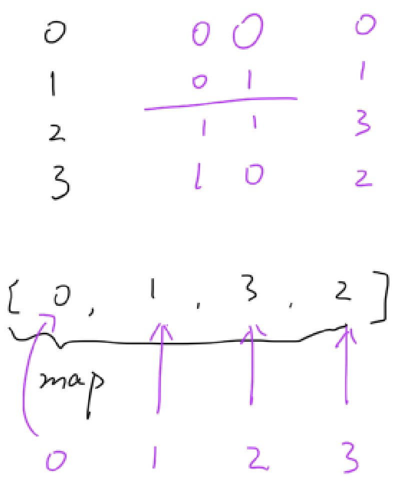
结果分析

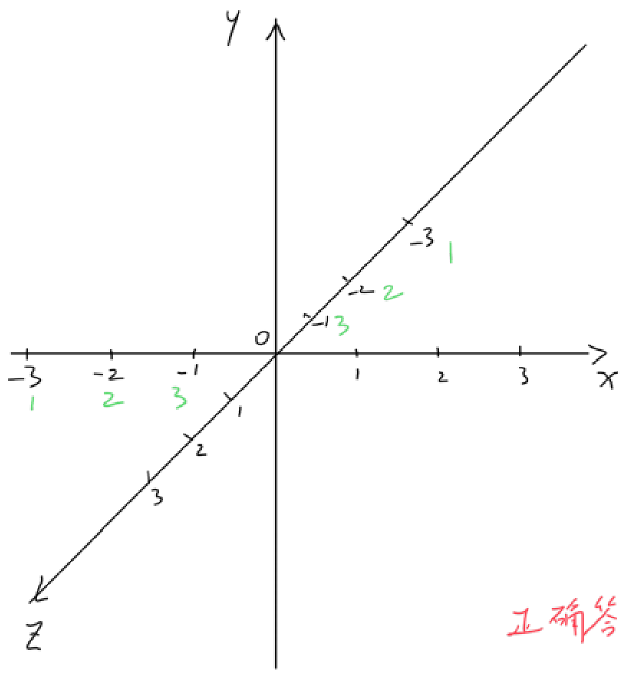
模型说明

利用plotly第三方库画出了一个3D图形，保存为“vector.html”，可以直接在浏览器中打开查看。

该三维图形中的整数三维坐标和6bit的答题状态是一一对应的关系。具体来讲，将6bit的二进制状态码转换为2bit + 2bit + 2bit的编码方式，每两个bit都采用格雷码的编码方式，保证每一步的操作在三维空间中的欧氏距离均为1。

具体的编码规则如下所示：

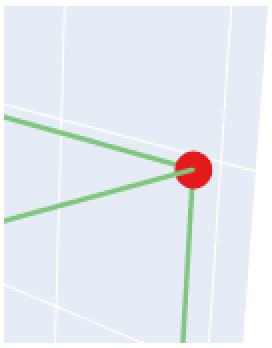




正确答案：			格雷码			坐标码		
10	10	11	2	2	3	3	3	2
10	11	10	2	3	2	3	2	3
11	10	01	3	2	1	2	3	1
11	11	00	3	3	0	2	2	0
00	11	11	0	3	3	0	2	2
01	11	01	1	3	1	1	2	1
11	00	11	3	0	3	2	0	2
11	01	10	3	1	2	2	1	3
01	01	11	1	1	3	1	1	2

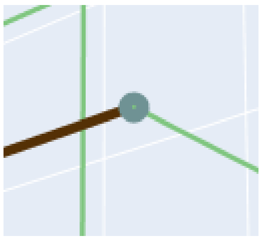
正确答案

正确答案用红色圆点标明：



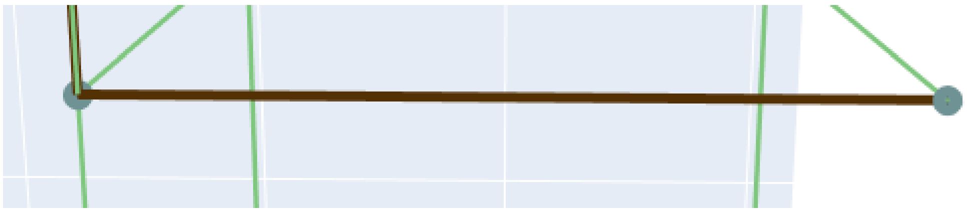
中间状态（重复出现20次）

虽然不是最终正确答案，但是出现频次较高，采用青色圆点表示：



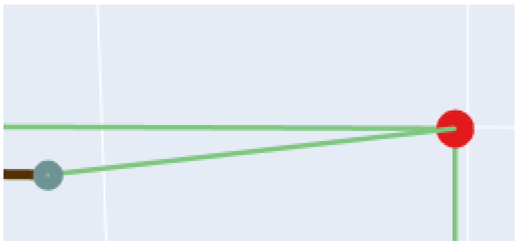
较为普遍的步骤（15次以上）

在图中，出现频次较多的步骤用棕色粗实线标出：



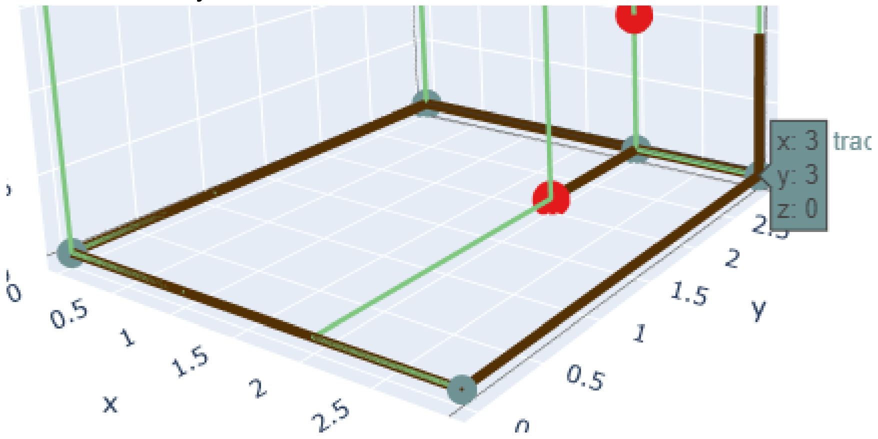
次为普遍的步骤（5~15次）

出现次数比较多的用绿色圆点表示：

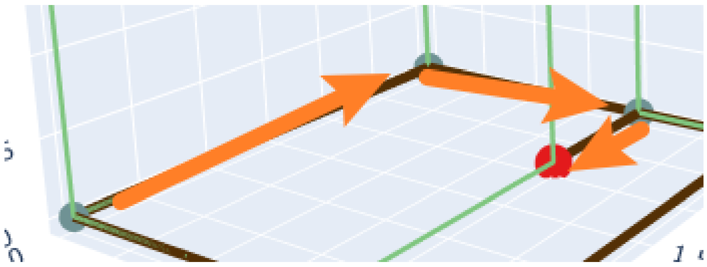


教育类结论

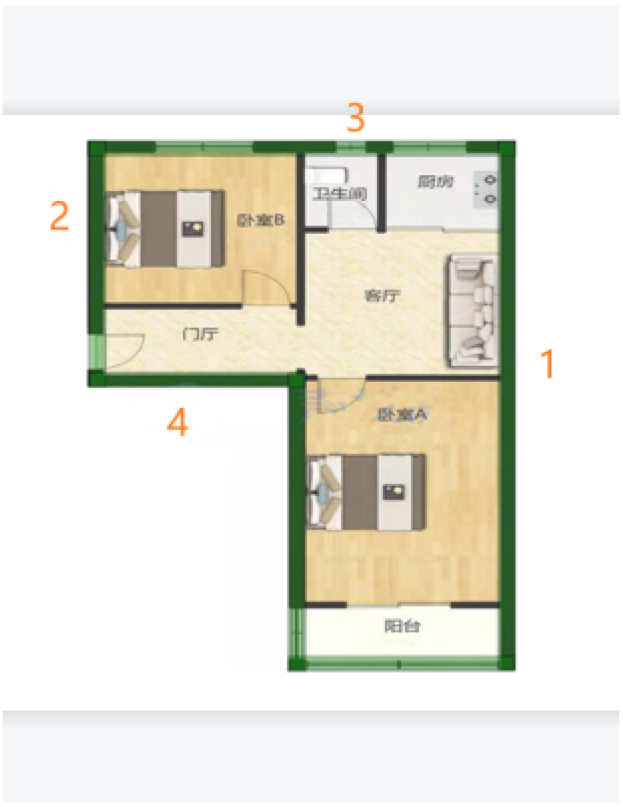
较为普遍的步骤和中间状态集中于xoy平面上



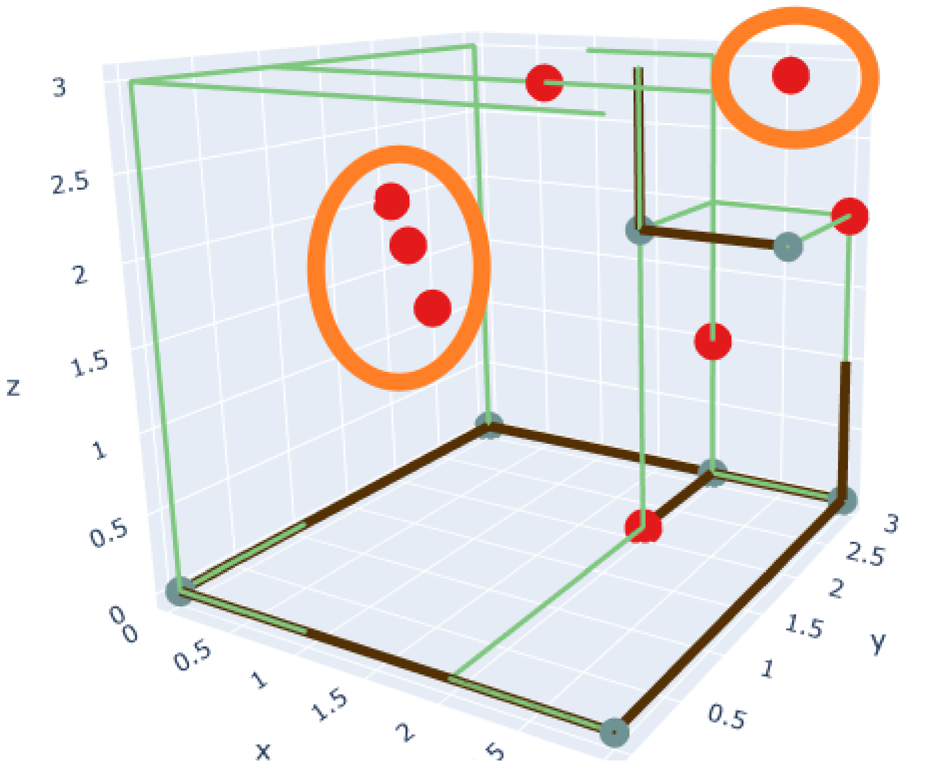
最为常见的解题思路如下



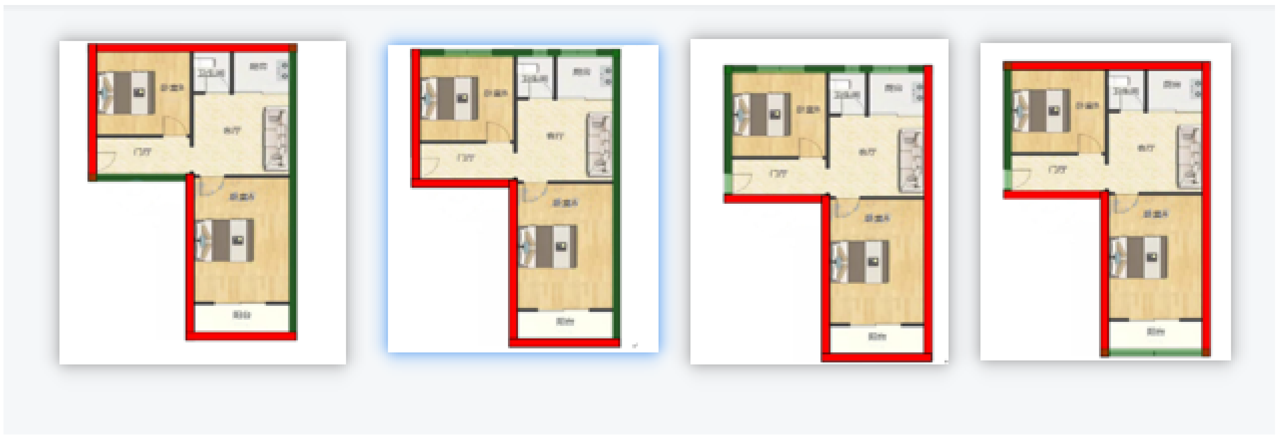
即，000000->001000->011000->111000->111100 在题目中为：



另外还有几种较为冷门的答案



分别为：110011,010111,001111，101110 在题目中为：



```
In [ ]:
```