Edge.py (Field)

Name	Type	Description
i1	Vertex	vertex 1
i2	Vertex	vertex 2

Vertex.py (Field)

Name	Type	Description
lattice	(int, int)	represents which lattice this vertex belongs to
n	int	the n^{th} vertex on this lattice, n in {0, 1, 2, 3, 4, 5}
sign	(int, int, int)	the signature of vertex, use as search key in dictionary
unionSign	(int, int, int)	the parent sign of this vertex
union	Vertex	the parent vertex, if the vertex has union as itself, then it is the root of the union

Vertex.py (Method)

Name	Input	Output	Description
join	Vertex	None	join two vertices together, make the input vertex as the union of current vertex
find_root_vertex	None	Vertex	find the root vertex of the current vertex
find_root_sign	None	(int, int, int)	find the signature of the root vertex

Map.py (Field)

Name	Type	Description
map_type	int	represents the type of map
size	int	represents the map size
gird	dict{(int, int) : Lattice}	represents the lattices in the map, use the signature of lattice as search key
vertices	dict{(int, int, int) : Vertex}	represents all the vertices on the lattice in map before joining them. The total number of vertices will large then the actual number of intersections in map. Use the signature of vertex as search key
edges	dict{((int, int, int), (int, int, int)) : Edge}	represents all the actual edges on the map, use two signatures of vertex in a tuple as search key

Map.py (Method)

Name	Input	Output	Description
generate_map	None	None	create a complete map
generate_numbers	None	list[int]	create a list of numbers to fill in the map
generate_resources	None	list[str]	create a list of resources to fill in the map
generate_grid	None	dict{(int, int) : Lattice}	create a dict contain all the lattices in the map, use the signature of lattice as search key
get_total_lattice	None	int	return the number of lattices in map
get_total_edges	None	int	return the number of edges in map
join_vertices	None	None	join all the repeated vertex, the final number of unions will equal the actual number of vertices.

generate_edges	None	dict{((int, int, int), (int, int, int)) : Edge}	create a dict contain all the actual edges on map, use two signatures of vertex as search key
get_resources	Vertex	list[str]	return a list of resources belong to the input vertex

Lattice.py (Field)

Name	Type	Description
x	int	x coordinate
y	int	y coordinate
sign	(int, int)	signature of lattice
n	int	the number on this lattice, 0 represents a desert
resource	str	represents the resource on this lattice
isRobbed	bool	represents weather the resource is robbed on this lattice

Lattice.py (Method)

Name	Input	Output	Description
set_number	int	None	set number on lattice
set_resource	str	None	set resource on lattice

Player.py (Field)

Name	Type	Description
name	str	player's name
vertices	list[Vertex]	represents all the vertices that the player has reached
resources	list[str]	represents all the resources the player has
road	list[Edge]	represents all the roads the player built
house	list[Vertex]	represents all the houses the player built
city	list[Vertex]	represents all the cities the player built

Player.py (Method)

Name	Input	Output	Description
information	None	str	return the information of resources, house, roads and cities of the player in string
build_road	Edge, list[Player]	None	build road for the player if it has enough resources and the road is not being built by other player
build_house	Vertex, list[Player]	None	build house for the player if it has enough resources and the vertex is not conflict to other players
build_city	Vertex	None	build city for the player if it has enough resources and it has a house on the input vertex
longest_road	None	int	return the length of the player's longest road
bfs	Vertex	int	return the length of the longest road start in the input vertex
get_point	None	int	return the basic score of the player

Game.py (Field)

Name	Type	Description
map	Map	the map of game
players	list[Player]	the players of game
turn	int	the turns of game
score_board	dict{Player : int}	recording the score of players
dice	Dice	the dice of game

Game.py (Method)

Name	Input	Output	Description
before_game	list[Player]	None	the initial build round of all the players
start	None	None	start the game
end	None	None	check if the game ends
no_man_vertex	Vertex	bool	return true if no player has the input vertex
no_man_road	Edge	bool	return true if no player has built road on the input edge
select_vertex	None	Vertex	show a message and get a vertex that guarantees to be on the map
select_road	None	Edge	show a message and get an edge that guarantees to be on the map
add_resources	int	None	add resources to players if they have vertex on the lattice that has input number