

Guia de datos

Al ejecutar las simulaciones se generan datos que su algoritmo va recibir (definen el escenario para este evento o simulación de ciudad), datos de la simulación y resultados. En esta guía vamos a repasar cada uno de ellos.

Datos de Input

La estructura de código de los inputs lo pueden ver al inicio de la función de `plan_evacuation` en el `solution.py`.

```
def plan_evacuation(self, city: CityGraph, proxy_data: ProxyData,
                    max_resources: int) -> PolicyResult:
    """
    Plan the evacuation route and resource allocation.

    Args:
        city: The city layout with nodes and edges
            - city.graph: NetworkX graph with the city layout
            - city.starting_node: Your starting position
            - city.extraction_nodes: List of possible extraction points

        proxy_data: Information about the environment
            - proxy_data.node_data[node_id]: Dict with node indicators
            - proxy_data.edge_data[(node1,node2)]: Dict with edge indicators

        max_resources: Maximum total resources you can allocate

    Returns:
        PolicyResult with:
            - path: List[int] - List of node IDs forming your evacuation path
            - resources: Dict[str, int] - How many of each resource to take:
              {'explosives': x, 'ammo': y, 'radiation_suits': z}
              where x + y + z <= max_resources
    """
    print(f'City graph: {city.graph} \n')
    print(f'City starting_node: {city.starting_node} \n')
    print(f'City extraction_nodes: {city.extraction_nodes} \n')
    print(f'Proxy node_data: {proxy_data.node_data} \n \n')
    print(f'Proxy edge_data: {proxy_data.edge_data} \n \n')
    print(f'Max Resources: {max_resources} \n \n')
```

PROF

Su algoritmo en `public/tools/simulator.py` recibirá 3 datos (variables input):

1. `city` [Objeto con `networkx` y otros valores], representa lo que conocemos de la ciudad. Conexiones entre nodos de la ciudad, nodo inicial, y nodos de extracción.
 - 1.1 `city.graph` [`nx.Graph`]: Es un archivo de tipo `networkx` que es la librería de grafos más usada en python. [Video Introductorio a NetworkX](#). Representa las conexiones "**esperadas**" de la ciudad. La ventaja de utilizar este objeto es que pueden aplicar algoritmos, métodos o lo que deseen de la librería `networkx` sin necesidad de hacer el código manualmente, esto incluye visualizaciones y otras cosas interesantes. Esto se ve reflejado en el algoritmo **Naive** que se proporciona de inicio, ya que usamos el método de esta librería para calcular el **shortest** path de manera automática sobre este objeto. Mi recomendación es que utilicen esta estructura y esta librería para ejecutar sus algoritmos, sin embargo pueden convertirla a una **linked_list** y usar las cosas de manera manual. Para entender mejor que puedes hacer con este objeto lee la documentación '`docs/b_tutorial_nx.md`'.

```
path = nx.shortest_path(city.graph, city.starting_node, target,
                        weight='weight')
```

1.2 `city.starting_node [Dict]`: es un entero que representa el nodo donde empezaran su equipo al que tienen que extraer.

1.3 `city.extraction_nodes [Dict]`: es una lista con los nodos que son de extraccion. tienen que llegar a cualquiera de los dos para que la extraccion se de como valida.

```
City graph: Graph with 5 nodes and 9 edges
City starting_node: 4
City extraction_nodes: [0, 2]
```

2. `proxy_data [Objeto de Diccionarios]`: es un diccionario (json) que contiene los proxies o datos que reflejan las condiciones o el estado de la ciudad o grafica.

2.1 `proxy_data.node_data [Dict]`: So los datos relacionados al **nodo**, la llave es el nodo, y el diccionario las variables y los valores para cada nodo. `{'nodo': {'caracteristica': 'valor'}}`. Son las características que conocemos del lugar.

```
proxy_data.node_data = {
    '0':
        {'caracteristica_1':valor, ..., 'caracteristica_n':valor},
    ...,
    'n':
        {'caracteristica_1':valor, ..., 'caracteristica_n':valor}
}
```

```
Proxy node data: {0: {'seismic_activity': 0.78179544902325, 'radiation_readings': 0.6484176415293725, 'population_density': 0.13925749224698464, 'emergency_calls': 0.972773406739206, 'thermal_readings': 0.3008342374664341, 'signal_strength': 0.2648065249719594, 'structural_integrity': 0.279251955899699}, 1: {'seismic_activity': 0.67122348334622, 'radiation_readings': 0.7212042463994993, 'population_density': 0.2173553202037166, 'emergency_calls': 0.821833664466666, 'thermal_readings': 0.3283180144233366, 'signal_strength': 0.166375253160666, 'structural_integrity': 0.818834649411954973}, 2: {'seismic_activity': 0.47003094040978, 'radiation_readings': 0.659568713876666, 'population_density': 0.22489710801416, 'emergency_calls': 0.8158769776748579, 'thermal_readings': 0.5366044489463405, 'signal_strength': 0.37633766264217503, 'structural_integrity': 0.3462484123869807}, 3: {'seismic_activity': 1.0, 'radiation_readings': 0.973864029308939, 'population_density': 0.052651212991281, 'emergency_calls': 0.818652552267672, 'thermal_readings': 0.52571108083951, 'signal_strength': 0.017208754868469, 'structural_integrity': 0.0}, 4: {'seismic_activity': 0.7462716815942878, 'radiation_readings': 1.0, 'population_density': 0.23752581013914578, 'emergency_calls': 0.8529274242803054, 'thermal_readings': 0.290410756702853, 'signal_strength': 0.2121997788363637, 'structural_integrity': 0.353836147186665}}
```

2.2 `proxy_data.edge_data [Dict]`: Son los datos relacionados al **edge**, la llave es una tupla que representa la conexion entre dos nodos (**nodo1, nodo2**), y el diccionario las variables y los valores para cada nodo. `{('nodo1, nodo2'):{'caracteristica': 'valor'}}`. Son las características que conocemos de las calles o conexiones entre los nodos.

```
proxy_data.edge_data = {
    '(0,20)':
        {'caracteristica_1':valor, ..., 'caracteristica_n':valor},
    ...,
    '(n,m)':
        {'caracteristica_1':valor, ..., 'caracteristica_n':valor}
}
```

```
Proxy edge data: {(0, 1): {'structural_damage': 1.0, 'signal_interference': 0.4188574669247157, 'movement_sightings': 0.4386745836899659, 'debris_density': 0.9388025877191596, 'hazard_gradient': 0.89667238569685744}, (0, 3): {'structural_damage': 1.0, 'signal_interference': 0.7375266611955345, 'movement_sightings': 0.2055884845321894, 'debris_density': 0.891576554557689, 'hazard_gradient': 0.2621776556315086}, (0, 4): {'structural_damage': 0.9181410244754, 'signal_interference': 0.65283851596717, 'movement_sightings': 0.3988742846727852, 'debris_density': 0.6914002765590577, 'hazard_gradient': 0.348338659683453}, (0, 2): {'structural_damage': 0.463974261661595, 'signal_interference': 0.260333441894399, 'movement_sightings': 0.3363389999465899, 'debris_density': 0.8535722352579, 'hazard_gradient': 0.12058369064654291}, (1, 4): {'structural_damage': 0.9137042655489794, 'signal_interference': 0.65378455133802, 'movement_sightings': 0.585448547816689, 'debris_density': 0.528735868355862, 'hazard_gradient': 0.4388081726074889}, (1, 3): {'structural_damage': 1.0, 'signal_interference': 0.7559731893661106, 'movement_sightings': 0.17275879278132816, 'debris_density': 0.864351560719089, 'hazard_gradient': 0.318555080617547}, (1, 2): {'structural_damage': 1.0, 'signal_interference': 0.4216316923087105, 'movement_sightings': 0.3746461808240524, 'debris_density': 0.92506013103023, 'hazard_gradient': 0.0123577253115209}, (2, 0): {'structural_damage': 1.353123928627076, 'signal_interference': 0.466368032901597, 'movement_sightings': 0.42172417891258574, 'debris_density': 0.5994833902870241, 'hazard_gradient': 0.4362846135027738}, (3, 4): {'structural_damage': 1.0, 'signal_interference': 0.5018630913768494, 'movement_sightings': 0.25912726468443107, 'debris_density': 0.9325981720177893, 'hazard_gradient': 0.0}}
```

3. `max_resources [int]`: Es el numero maximo de recursos que pueden cargar las personas que

seran extraidas en esta ciudad. Puede variar, no es fijo.

Max Resources: 4