A * algorithm

```
import networkx as nx
def a_star_pathfinding(graph, start_node, goal_node, heuristic):
    try:
        path = nx.astar_path(graph, start_node, goal_node, heuristic=heuristic, weight='weight')
        return path
    except nx.NetworkXNoPath:
        return None
# Example Usage:
if __name__ == "__main__":
    # Create a sample graph
    G = nx.Graph()
    G.add_edge('A', 'B', weight=6)
    G.add_edge('A', 'F', weight=3)
    G.add_edge('B', 'C', weight=3)
    G.add_edge('B', 'D', weight=2)
    G.add_edge('C', 'E', weight=5)
    G.add_edge('D', 'E', weight=8)
    G.add_edge('F', 'G', weight=1)
```

```
    G.add_edge('G', 'H', weight=7)
    G.add_edge('H', 'I', weight=2)
    G.add_edge('E', 'I', weight=5)
    G.add_edge('I', 'J', weight=3)
    # Define a heuristic function (example: straight-line distance or estimated cost)
    # In a real-world scenario, this would be more complex and domain-specific.
    heuristic_values = {
        'A': 11, 'B': 6, 'C': 5, 'D': 7, 'E': 3,
        'F': 6, 'G': 5, 'H': 3, 'I': 1, 'J': 0
    }
    def example_heuristic(u, v):
        return heuristic_values.get(u, 0) # Simple example, usually depends on v as well
    start = 'A'
    goal = 'J'
    path = a_star_pathfinding(G, start, goal, example_heuristic)
    if path:
        print(f"Path found from {start} to {goal}: {path}")
        length = nx.astar_path_length(G, start, goal, heuristic=example_heuristic, weight='weight')
        print(f"Path length: {length}")
```

```
else:
    print(f"No path found from {start} to {goal}.")
```