```
import heapq

def uniform_cost_search(graph, start, goal):
    queue = [(0, start)]  # (cost, node)
    visited = set()
    while queue:
        cost, current = heapq.heappop(queue)
        if current in visited:
            continue
        visited.add(current)
        if current == goal:
            return cost
        for neighbor, edge_cost in graph[current]:
            if neighbor not in visited:
                heapq.heappush(queue, (cost + edge_cost, neighbor))
    return None

# Example usage:
graph = {
    'A': [('B', 2), ('C', 3)],
    'B': [('D', 4)],
    'C': [('D', 5)],
    'D': [('E', 6)],
    'E': []
}

print(uniform_cost_search(graph, 'A', 'E'))
```