```
def nearest_neighbor(G, v=0):
    visited = [v]
    for _ in range(len(G) - 1)::
        row = G[visited[-1]]
        min = (-1, math.inf)
        for j in range(len(G)):
            if 0 < row[j] < min[1] and j not in visited:
                min = (j, row[j])
        if min[0] == -1:
            return None
        visited.append(min[0])

    if G[visited[-1]][v] == 0:
        return None

    return visited + [v]
```

الگوریتم ابتدا از راس v شروع می‌کند و نزدیک‌ترین همسایه‌اش را به لیست $visited$ اضافه می‌کند. سپس الگوریتم این کار را برای بقیه راس‌ها نیز انجام می‌دهد بدین صورت که نزدیک‌ترین راسی که در لیست $visited$ نباشد را به عنوان نزدیک‌ترین همسایه به این لیست اضافه می‌کند اگر در بررسی یکی از راس‌ها هیچ راس کناری‌ای چنین شرطی را نداشت، الگوریتم به بن‌بست می‌رسد که می‌تواند نشان دهنده‌ی نبودن دور باشد البته ممکن است دور وجود داشته باشد اما الگوریتم نتوانسته باشد آن‌را پیدا کند. این موضوع به راس اولیه نیز مرتبط است اما کاملا تابع این موضوع نیست.