☰ Item Navigation

Congratulations! You passed!

Grade received 100% **To pass** 80% or higher

Go to next item

Dynamic Programming

Practice Quiz • 1h

Submit your assignment

Try again

Receive grade

To Pass 80% or higher

Your grade

100%

View Feedback

We keep your highest score

Like Dislike Report an issue

1. Implement the Dynamic Programming algorithm for the Traveling Salesman problem.

1 / 1 point

```
    n = g.number_of_nodes()

    # The variable power now contains a tuple for each subset of the set {1, ..., n-1}.
    power = powerset(range(1, n))
    # The variable T is a dictionary, where the element T[s, i] for a set s and an integer i
    # equals the shortest path going through each vertex from s exactly once,
    # and ending at the vertex i.
    # Note that i must be in s.
    # Also, we will always assume that we start our cycle from the vertex number 0.
    # Thus, for convenience, we will always exclude the element 0 from the set s.
    T = {}
    # For every non-zero vertex i, we say that T[ tuple with the element i only, i]
    # equals the weight of the edge from 0 to i.
    # Indeed, by the definition of T, this element must be equal to the weight of
    # the shortest path which goes through the vertices 0 and i and ends at the vertex i.
    for i in range(1, n):
        # Syntactic note: In Python, we define a tuple of length 1 that contains the element i as (i,) *with comma*.
        T[(i,), i] = g[0][i]['weight']

    # For every subset s of [1,...,n-1]
    for s in power:
        # We have already initialized the elements of T indexed by sets of size 1, so we skip them.
        if len(s) > 1:
            # For every vertex i from s which we consider as the ending vertex of a path going through vertices from 
            for i in s:
                # Define the tuple which contains all elements from s without *the last vertex* i.
                t = tuple([x for x in s if x != i])
                # Now we compute the optimal value of a cycle which visits all vertices from s and ends at the vertex
                for j in range(n):
                   if j == i or (not j in s):
                       continue
                   T[tuple(s), i] = min(T.get((tuple(s), i), float('inf')), T[t, j] + g[j][i]['weight'])

                # WRITE YOUR CODE HERE

    # Return the weight of on optimal cycle - this is the minimum of the following sum:
    # weight of a path + the last edge to the vertex 0.
    return min(T[tuple(range(1, n)), i] + g[i][0]['weight'] for i in range(1, n))


```

Run

Reset

```
No Output
```

✓ **Correct**

Good job!

Heads Up!

✕

Over **63%** of learners reviewed this practice assignment more than once and found it helpful. Pay attention!

Was this helpful? Yes No