University of Washington March 13, 2020
Department of Computer Science and Engineering
CSE 417, Winter 2020

Homework 9, Problem 2

Problem 2 (10 points):

Give an algorithm, which given a directed graph $G = (V, E)$, with vertices $s, t \in V$ and an integer k, determines the number of paths from s to t of length k. Your algorithm should be polynomial in k, $|V|$ and $|E|$.

Answer:

algorithm: Assume the graph is provided in form of adjacency matrix, if not, it cost $O(m+n)$ to reconstruct adjacency list to matrix. The adjacency matrix is given by adj[][].

```
def number_of_path(adj, k):
    for p in range(k):
        for i in range(n):
            for j in range(n):
                dp[i][j][p] = 0
                if p == 0:
                    dp[i][j][p] = 1
                if p == 1 or adj[i][j]:
                    dp[i][j][p] = 1
                if p > 1:
                    for b in range(n):
                        if adj[i][b]:
                            dp[i][j][p] += dp[b][j][p - 1]
    return dp[s][t][k]
```

proof:
base case:
For $p = 0$:
Because the length of path is zero, there are only one way that no path from any i to j, hence dp[i][j][p] = 1.
For $p = 1$:
Because the length of path is one, there can only be one way from i to j with 1 length path, either it has a path or not, hence dp[i][j][p] = 1.

induction hypothesis:
For $p > 1$:
for b in range(n): if adj[i][b]: dp[i][j][p] += dp[b][j][p - 1]

At each path length, the dp trace back the number of path of length p - 1 from b to j if there is a edge from i to b.
Since at each path length p, for each node i, there are only two possibilties for each edge b-i:
(1) if b has path to j with length p-1, then at length p, dp[i][j] can inherit the number of path with b as a intermediary.
(2) if b has no path to j with length p-1, then at length p, dp[i][j] can not inherit the number of path with b as a intermediary.
Since the algorithm always correctly compute subproblem (the number of path from node i to any other node) with 0 to $p_i - 1$ path length, at path length p_i, by updating value from precalculated value, it garantees to get all possibilties path with length k.

complexity:
Since on each subproblem, the algorithm will traces the number of path from i to j with p edges, and it will iterate all node and its neighbor, hence cost $O(n^2)$ with k subproblems. Therefore the time complexity is $O(k * n^2)$.