Floyd-Warshall algorithm

Problem

Given a weighted graph G(V,E) with positive or negative edge weights (but with no negative cycles). find the lengths (summed weights) of shortest paths between all pairs of vertices

`Input :` G(V,E), V = {1,2,3...n}

`Output` : Matrix representing the shortest path between all pairs of vertices

Solution

Defining Subproblem

Lemma: The same vertex does not appear twice in a shortest path.

Proof : A path that has the same vertex twice will have a cycle. Removing the cycle results in a shorter path.

Let $d(i, j, k)$ be the length of the shortest path from i to j such that all intermediate vertices on the path (if any) are in set {1,2,3 ....,k}

Let D_k be the $n \times n$ matrix $[d(i, j, k)]$.

Here Our goal is to find D_n

Getting recursion relation

There are two possibilities for a shortest path from I to j in which any intermediate vertices on the path are chosen from the set{1,2,3 ... k}:

1. k is not a vertex on the path then shortest path is $d(i, j, k-1)$
2. k is a vertex on the path then shortest path is $d(i, k, k-1) + d(k, j, k-1)$

Consider the shortest path between I and j that includes the vertex k. It is made up of two subpaths, one from I to k and the other from k to j.

Path Length = $d(i, k, k-1) + d(k, j, k-1)$

Combining both the cases we can get the following recursion relation

$$d(i, j, k) = \min d(i, j, k-1), d(i, k, k-1) + d(k, j, k-1)$$

Bottom-up Computation

- Bottommost(smallest problem) D_0 this will same as given transition matrix as D_0 is the shortest path between each pair of vertices using no intermediate states.
- Then we can calculate D_1 using above relation as $d(i, j, 1) = \min d(i, j, 0), d(i, 1, 0) + d(1, j, 0)$
- Since we don't need to keep track D_{k-1} after evaluating D_{k}. . We only maintain one matrix and keep overwriting it. This overwriting dont mess-up the older matrix before getting the values of newer matrix because
    - To calculate any $d(i, j, k)$ we only need the value $d(i, j. k-1)$, $d(i, k, k-1)$, $d(k, j, k-1)$

- From the definition of d(i,j,k). we can say $d(i,k,k-1) = d(i,k,k)$ and $d(k,j,k-1) = d(k,j,k)$
- Since we just reached $d(i,j.k-1)$ That will also be unchanged before we reach there.
- Since all three required values remains unchanged till we reach the point . This method should work.

Time Analysis

Since we are filling n^3 elements and each filling takes O(1).

Final complexity = $O(n^3)$

Space complexity

From the disscussion we conclude that we only need to keep track of 1 2-d table so the

Final space Complexity = $O(n^2)$