All-Pair Shortest Path Problem

- Harisam Sharma

Problem Statement

Given a weighted Graph G(V,E), we need to find the length of the shortest path between each pair of vertices. I.e, for all pairs of nodes (u,v), we need to compute dist[u][v] = minimum sum of weights of edges of any path going from u to v.

Floyd Warshall's Algorithm

- This is the standard algorithm which can solve the APSP Problem.
- Again, this Algorithm is based on **Dynamic Programming**.
- The algorithm works in O(N^3) time (independent of M), can you try and guess the states of the DP?
- Dp[i][j][k] = shortest distance between nodes i and j such that the intermediate path contains only the nodes {1,2,....k}
- Dp[i][j][k] = min(dp[i][j][k-1], dp[i][k][k-1] + dp[k][j][k-1])

```cpp
vector<vector<ll>>floyd_warshall()
{
    vector<vll>dist(n+2,vll(n+1,inf));
    // base cases, when i = j, and when we are using direct edges.
    fo(i,1,n)
    {
        dist[i][i] = 0;
    }
    for(auto x : edges)
    {
        dist[x.first][x.second] = min(dist[x.first][x.second], x.weight);
    }

    fo(k,1,n)
    {
        fo(i,1,n)
        {
            fo(j,1,n)
            {
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
            }
        }
    }
    return dist;
}
```

Time Complexity: O(N^3)

Sample Problem

Link: https://codeforces.com/contest/295/problem/B