1. All Pairs Shortest Path algorithm using matrix arithmetic

Nomenclature

- A : Adjacency Matrix
- D : Distance matrix
- A ∗ B : Modified_Matrix_Multiplication(A,B)
- A^n : Power(A,n) using modified matrix multiplication

Algorithm

- Define modified matrix multiplication function
- We initialize the result matrix C with very large numbers which represent there is not path initially between the nodes
- Modification in that function is that instead of summing over the values of A[i][k] * B[k][j] we take the minimum(A[i][k] + B[i][k]) for k from 1 to n

- Now in main code, we initialise a matrix DIST by assigning it Values from Adj_mat
- We keep squaring the matrix DIST using modified matrix multiplication until the power of DIST is more than n-1
- Now, the final DIST will be our distance matrix

Pseudo Code

```
Modified_Matrix_Multiplication(A,B):
        C ← inf  // matrix of len(A) × len(B[0])
        LOOPS (i, 0, len(A)):
                LOOPS (j, 0, len(B)):
                        LOOPS (k, 0, len(B[0])):
                                C[i][j] ← min(C[i][j], A[i][k] + B[k][j])
                        END
                END
        END
        RETURN C


All_Pairs_Distance(A):
      N ← len(A)
      DIST ←  A
      L ← 1

      WHILE( L < n - 1 ) :
          DIST ← Modified_Matrix_Multiplication(DIST,DIST)
          L ← L + L
      END
```

2. Proof of Correctness

D(k) : The distance matrix such that paths of utmost **k edges** are covered

Lemma 1. D(k) = A^k

Proof: We will prove this by induction

1. D(1) = A, The adjacency matrix will contain the 1 for directly connected nodes, hence the D(1) will have stored the paths of utmost 1 length

2. Now suppose we have D(k) then, for paths of utmost k+1 length
    a. Dij (k+1) = MIN { Dij (k) , MIN $1 \le p \le n$ { Dip (k) + Apj }}
    = MIN $1 \le p \le n$ { Dip (k-1) + Apj }

3. The above formula is the same as calculated in the modified matrix multiplication, hence D(k+1) = D(k) $*$ A
4. Now since
    a. D(1) = A
    b. D(2) = D(1) $*$ A = A $*$ A = A^2
    c. D(3) = D(2) $*$ A = A^2 $*$ A = A^3
5. Using recursion, D(n) = A^n will be our distance matrix

Lemma 2. D(2*k) = D(k) $*$ D(k)

Proof: We will prove this by contradiction

1. Let us assume for some i, j : Dij (2*k)= k1
2. Actual distance between i and j = k2
3. Assumption: k1 $\neq$ k2

4. By definition of distance: k2 = D**im** (k) + D**m**j(k) such that k2 is minimum, $1 \le$ **m** $\le n$

5. Formula for Dij (2*k) in modified matrix multiplication will be:
    a. Dij (2*k) = MIN { Dij (2*k) , MIN $1 \le p \le n$ { Dip (k) + Dpj (k) }}
    b. k1 = MIN $1 \le p \le n$ { Dip (k) + Dpj (k) }

6. Comparing the results from (4) and (5) k1 will be the minimum value among all combination of sums between Dip(k) + Dpj(k), and hence it will be equal to k2
7. Hence by contradiction we proved that, D(2*k) = D(k) $*$ D(k)

Lemma 3. We can compute D(n) in log(n) iterations

Proof: We will prove this by observation

1. D(1) = A … 1st iteration
2. D(2) = D(1) $*$ D(1) = A*A = A^2 … 2nd iteration

3. $D(4) = D(2) * D(2) = A^4$ … 3rd iteration
4. $D(8) = D(4) * D(4) = A^8$ ... 4th iteration

……..
……..
……..

5. $D(n) = A^n$ … log(n) th iteration

3. Time Complexity

Time complexity for Modified Matrix multiplication = N*N*N = N^3

Time complexity for iterating till D(n) : log(n)

Overall time complexity : N^3 * log(n)