COL351: Analysis and Design of Algorithms
Instructor: Ragesh Jaiswal

1. *Matrix Chain Multiplication Problem*: You are given a sequence of n matrices $M_1, ..., M_n$ of size $(m_1 \times m_2), (m_2 \times m_3), ..., (m_n \times m_{n+1})$. Determine in what order these matrices should be multiplied (using naïve method) so as to reduce the total number of multiplication operations.

   (Note that the total number of multiplication operations involved in multiplying two matrices of size $p \times q$ and $q \times r$ using the naïve method is pqr.)

   Let us consider an example to understand the problem better. Consider 4 matrices M_1, M_2, M_3, M_4 of size $(50 \times 20), (20 \times 1), (1 \times 10), (10 \times 100)$ respectively.

   The number of multiplication operations for different ways of multiplying these matrices are given below. The total cost of multiplying in different ways is different. We want to figure out a way of multiplying that uses the least number of operations.

   - $M_1 \times ((M_2 \times M_3) \times M_4)$
     - $20 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100 = 120,200$
   - $(M_1 \times (M_2 \times M_3)) \times M_4$
     - $20 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100 = 60,200$
   - $(M_1 \times M_2) \times (M_3 \times M_4)$
     - $50 \cdot 20 + 10 \cdot 100 + 50 \cdot 100 = 7000$

   For solving the problem using dynamic programming, we first need to define the subproblems. Let us define subproblems for computing the minimum number of operations required for matrix chain multiplication and then later we will figure out a way to output the "bracketing" that minimizes the total number of operations.

   For any $i < j$, let $C(i, j)$ denote the minimum cost of multiplying matrices $M_i \cdot M_{i+1} \cdot ... \cdot M_j$.

   Question 1: What is the value of $C(i, i)$ for any i?

   Question 2: Give a recursive formulation for $C(i, j)$ for any $i < j$.

   (Try writing $C(i, j)$ in terms of $C(i', j')$ where $j' - i' < j - i$.)

   We will now try to compute the value of $C(i, j)$ for all $i < j$. The minimum number of operations will be given by $C(1, n)$.

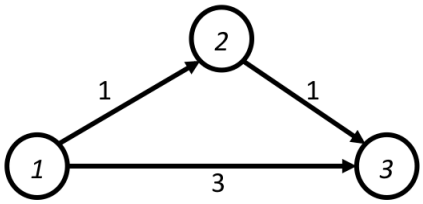
   Question 3: In what order should we compute the $C(i, j)'s$?

   Question 4: Write an algorithm that returns the minimum number of operations required.

   Question 5: What is the running time of your algorithm?

Question 6: How do we find the "bracketing" that minimizes the number of operations? Give an algorithm that outputs the bracketing.

2. *All pairs shortest paths problem*: Given a weighted, directed graph $G = (V, E)$, you are supposed to design an algorithm that outputs an $|V| \times |V|$ matrix A such that $A[i, j]$ contains the length of the shortest path in G from vertex i to vertex j.
(*For example, the shortest path matrix for the graph given below is on the right.*)



G

	1	2	3
1	0	1	2
2	∞	0	1
3	∞	∞	0

Figure 7.0.1: ∞ in a table entry $A[i, j]$ means that there is no path in the graph from vertex i to vertex j.

You can solve this problem using Dijkstra's algorithm repeatedly on the same graph and different starting vertices.

Question 1: What is the running time of the above algorithm?

We will design an algorithm with better running time using Dynamic Programming idea. For any i, j, k, let $D_k(i, j)$ denote the length of the shortest path from vertex i to vertex j when all the intermediate vertices in the path is from the set $\{1, ..., k\}$.

Given the above definition, we can say that that for all i, j, $D_0(i, j)$ = weight of edge (i, j) in case it exists, otherwise ∞.

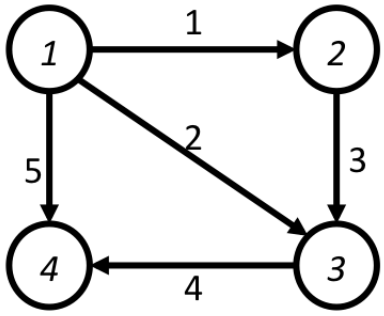
Question 2: Write $D_1(.,.)$ in terms of $D_0(.,.)$.

Question 3: Write $D_i(.,.)$ in terms of $D_{i-1}(.,.)$.

Note that for the output matrix A, $A[i, j] = D_n(i, j)$ for all i, j. So, all we need to do is to figure out a way to compute $D_n(i, j)$ for all i, j. As evident from the recursive formulation in the previous question, we should compute $D_i(.,.)$ before computing $D_{i-1}(.,.)$. So, the algorithm runs in n passes and in the i^{th} pass it computes $D_i(j, k)$ for all j, k.

Question 4: What is the running time of the above algorithm? Is this better than running Dijkstra's repeatedly?

Question 5: Consider the graph given below and simulate this algorithm on this graph. That is, fill the tables $D_0(.,.), D_1(.,.), ..., D_4(.,.)$.



	1	2	3	4
1	0	1	2	5
2	∞	0	3	∞
3	∞	∞	0	4
4	∞	∞	∞	0

D_0

	1	2	3	4
1				
2				
3				
4				

D_1

	1	2	3	4
1				
2				
3				
4				

D_2

	1	2	3	4
1				
2				
3				
4				

D_3

	1	2	3	4
1				
2				
3				
4				

D_4