Name: Loc Nguyen

Last 4 of ID: 2624

CS3310 Homework 4

1. Multiplication order is A x ((B x C) x D). Work as shown below

C

	1	2	3	4
1	0	600	2700	1650
2		0	900	1350
3			0	1500
4				0

k	1	2	3	4
1		1	1	1
2			2	3
3				3
4				

Chained Matrix Multiplication

$$C[i,j] = \min_{i \le k < j} \{ C[i,k] + C[k+1,j] + d_i \times d_k \times d_j$$

$$\underset{20\ \ 3}{A} \times \underset{3\ \ 10}{B} \times \underset{10\ \ 30}{C} \times \underset{30\ \ 5}{D}$$

$$d_0 = 20,\ d_1 = 3,\ d_2 = 10,\ d_3 = 30,\ d_4 = 5$$

$$C[1,2] = \min_{1 \le k < 2} \ k=1 \{ C[1,1] + C[2,2] + d_0 \times d_1 \times d_2 \}$$
$$0 + 0 + 20 \times 3 \times 10 = 600$$

$$C[2,3] = \min_{2 \le k < 3} \ k=2 \{ C[2,2] + C[3,3] + d_1 \times d_2 \times d_3 \}$$
$$0 + 0 + 3 \times 10 \times 30 = 900$$

$$C[3,4] = \min_{3 \le k < 4} \ k=3 \{ C[3,3] + C[4,4] + d_2 \times d_3 \times d_4 \}$$
$$0 + 0 + 10 \times 30 \times 5 = 1500$$

$$C[1,3] = \min_{1 \le k < 3} \begin{cases} k=1 & C[1,1] + C[2,3] + d_0 \times d_1 \times d_3 \\ k=2 & C[1,2] + C[3,3] + d_0 \times d_2 \times d_3 \end{cases}$$

$$\Rightarrow \begin{array}{l} k=1 \rightarrow 0 + 900 + 20 \times 3 \times 30 = \boxed{2700} \\ k=2 \rightarrow 600 + 0 + 20 \times 10 \times 30 = 6600 \end{array}$$

$$C[2,4] = \min_{2 \le k < 4} \begin{cases} k=2 & C[2,2] + C[3,4] + d_1 \times d_2 \times d_4 \\ k=3 & C[2,3] + C[4,4] + d_1 \times d_3 \times d_4 \end{cases}$$

$$\Rightarrow \begin{array}{l} k=2 \rightarrow 0 + 1500 + 3 \times 10 \times 5 = 1650 \\ k=3 \rightarrow 900 + 0 + 3 \times 30 \times 5 = \boxed{1350} \end{array}$$

$$C[1,4] = \min_{1 \le k < 4} \begin{cases} k=1 & C[1,1] + C[2,4] + d_0 \times d_1 \times d_4 \\ k=2 & C[1,2] + C[3,4] + d_0 \times d_2 \times d_4 \\ k=3 & C[1,3] + C[4,4] + d_0 \times d_3 \times d_4 \end{cases}$$

$$\Rightarrow \begin{array}{l} k=1 \rightarrow 0 + 1350 + 20 \times 3 \times 5 = 1350 + 300 = \boxed{1650} \\ k=2 \rightarrow 600 + 1500 + 20 \times 10 \times 5 = 2100 + 1000 = 3100 \\ k=3 \rightarrow 2700 + 0 + 20 \times 30 \times 5 = 2700 + 3000 = 5700 \end{array}$$

Order $\Rightarrow (A) \times ((B \times C) \times D) = A \times ((B \times C) \times D)$

Complexity $\Rightarrow O(r^3)$

2. (M+1)x(n+1) Table

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	2	2	2	2	2	2	2	2
2	0	3	3	5	5	5	5	5	5	5	5
3	0	3	3	5	5	5	6	9	9	11	11
4	0	3	3	5	5	.6	6	9	9	11	11
5	0	3	4	7	7	9	9	11	11	13	13
6	0	3	4	7	7	9	9	11	12	13	14

The solution of 0/1 knapsack is M = [1,1,0,0,1,1] which includes the first, second, fifth, and sixth object and excludes the third and fourth object.

3. 0/1/2 Knapsack
   a. Let j = knapsack capacity, k is number of copies such that $0 \le k \le 2$, and i = object number so M[i][j][k] = Vi + M[i-1][j-ki][max(0, k-1)] when we include a certain object i. We have to add the value of object i and deduct its weight ki from the bag capacity of j for remaining capacity j-ki. I-1 represents the number of objects left and k is amount of copies allowed when object I is already included. It dictates the upper limit of how many more copies can be added.
   However, if we don't include object i, M[i][j][k] = M[i-1][j][k] such that the number of object we have to consider decreases by 1, and since we didn't include it, j and k remain the same.
   0 copies => M[i-1][j][k]
   1 copy => Vi + M[i-1][j-ki][max(0, k-1)]
   2 copies => 2Vi + M[i-1][j-2ki][max(0, k-2)]

   For $0 \le i \le n$ where n is the total number of items, $0 \le j \le W$ where W is the total capacity of the knapsack, and $0 \le k \le 2$

   We can derive the final equation to be M[n][W][0] where 0 represents how there will be 0 copies of any item remaining to be compared.

   b. Algorithm implementation
   M[i][j][k] = 0 for all i,j, and k
   For i from 1 to n:
   For j from 0 to W:
   For k from 0 to 2:
   If k = 0: then => M[i][j][k] = M[i-1][j][k]
   If k > 0 and ki>j: then => M[i][j][k] = M[i][j][k-1]
   If k > 0 and ki ≤ j: then => M[i][j][k] = max{M[i-1][j][k], Vi + M[i-1][j-ki][max(0, k-1)], 2Vi + M[i-1][j-2ki][max(0, k-2)]}
   Return M[n][W][0]

This algorithm first initialize variables weight and value. Then, for each item I and each knapsack weight from 0 to W, and for each k from 0 to 2, we try to find the maximum value that can be obtained by not including an object, include it once, or include it twice. Lastly, we return the max value for knapsack with weight capacity W and no copies of any item left to check.

c. Time complexity is O(nW) because the algorithm iterate over n, W, and 0,1,2 for values of k. Since each iteration of k takes constant time, hence the TC = O(nW)

4. True or False
    a. False. If P = NP, then problems that are NP-complete can be solved in polynomial time deterministically, which is not all problems in NP because one, not all problems are NP-complete and two, each problem requires a different algorithm.
    b. True. Because 5SAT is an NP-complete problem and there exist an algorithm solvable in polynomial time of n^{2021}, then P=NP.
    c. False. If X is NP-complete and it's reducible to Y in polynomial time, then Y is NP-hard. However, inversely, if X is reducible to Y and Y is NP-complete, then X is in NP and a problem both in NP and is in NP-hard is NP-complete.